

TM-S9000/S2000

API Reference Guide for Win32/64

Overview

Descriptions of the features and a development environment.

Install and Uninstall

Describes how to install and uninstall a driver.

Programming Guide

Descriptions of a programming method for application development using the TM-S9000/S2000 API.

TM-S9000/S2000 API Reference

Describes the TM-S9000/S2000 API.

Compatible Information

Describes the information on compatibility of the TM-J9000 API/TM-S1000 API.

Setting File

Descriptions of a configuration file for the TM-S9000/S2000 driver.

Log Function

Describes how to generate a log file.

Appendix

Describes the Software License.

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.

Copyright notice

EPSON is a registered trademark of Seiko Epson Corporation.

Exceed Your Vision is a registered trademarks or trademarks of Seiko Epson Corporation.

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

QR Code® is a registered trademark of DENSO Wave Incorporated.

Intel Core® and Pentium® are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.



All other trademarks are the property of their respective owners and used for identification purpose only.

©Seiko Epson Corporation 2012–2022.

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.

	Provides information that must be observed to avoid damage to your equipment or a malfunction.
	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 TM-S9000II, TM-S2000II, TM-S9000 and TM-S2000.

TM-S9000II and TM-S2000II have the same specifications as USB Mode for TM-S9000II-NW and TM-S2000II-NW. Descriptions of TM-S9000II and TM-S2000II in this manual also include USB Mode for TM-S9000II-NW and TM-S2000II-NW.

Manual Content

The manual is made up of the following sections:

Chapter 1	Overview
Chapter 2	Install and Uninstall
Chapter 3	Programming Guide
Chapter 4	TM-S9000/S2000 API Reference
Chapter 5	Compatible Information
Chapter 6	Setting File
Chapter 7	Log Function
Appendix	Software License

Contents

■ For Safety	3
Key to Symbols	3
■ Restriction of Use	3
■ About this Manual	4
Aim of the Manual	4
Manual Content	4
■ Contents	5

Overview..... 12

■ Features	12
Structure	13
■ Features of the TM-S9000/S2000 API	16
■ Operating Environment	18
OS	18
Computer	18
Interface	18
■ Development Environment	19
■ Technical Support Web Site	20

Install and Uninstall21

■ Installation	21
■ Uninstallation	22
■ Silent Install	23
Creating a Silent Install Setting File	24
Setting a Silent Install Setting File	24
Executing the silent install	25
Executing the silent uninstall	26
Confirming the result of the Silent Install	26

Programming Guide.....27

■ Application Processing Steps 27

Step 1. Preparing for the use of TM-S9000/S2000 API	28
Step 2. Opening the device	28
Step 3. Registering CALLBACKs.....	28
Step 4. Setting the scan processing.....	30
Step 5. Performing scan processing	32
Step 6. Starting print data transmission	33
Step 7. Successfully completing scan processing	34
Step 8. Occurrence of a paper jam error.....	35

■ Programming flow 38

Check scan processing	38
Card scan processing	40
Electronic endorsement printing	42
Physical endorsement printing.....	44
Roll paper printing (TM-S9000II and TM-S9000).....	47
Template printing	49
Cut sheet paper printing	51
CTS2010 compliant operation.....	54

■ Priority for error detection 56

■ How to Use the Scanner Advanced Functions..... 58

When not using the scanner advanced functions	58
Using the scanner advanced functions.....	58
Editing scanned-in images	59
Cropping	59

■ Actions to be taken in the event of errors 60

Device status.....	60
Return value	61

■ Sample Program..... 64

TM-S9000/S2000 API Reference65

■ API Index..... 65

Basic APIs.....	65
API List (Alphabetical Order)	66

■ BiOpenMonPrinter	68
■ BiCloseMonPrinter	70
■ BiSCNMICRSetStatusBackFunction	71
■ BiSCNMICRSetStatusBackWnd	73
■ BiSCNMICRSetStatusBackWndEx	75
■ BiSCNMICRCancelStatusBack	77
■ BiStartEndorsementSetStatusBackFunction	78
■ BiStartEndorsementSetStatusBackWnd	79
■ BiStartEndorsementCancelStatusBack	80
■ BiEndEndorsementSetStatusBackFunction	81
■ BiEndEndorsementSetStatusBackWnd	82
■ BiEndEndorsementCancelStatusBack	84
■ BiGetStatus	85
■ BiGetRealStatus	86
■ BiSetStatusBackFunction	87
■ BiSetStatusBackFunctionEx	88
■ BiSetStatusBackWnd	89
■ BiSetStatusBackWndEx	90
■ BiCancelStatusBack	91
■ BiGetInkStatus	92
■ BiSetInkStatusBackFunction	93
■ BiSetInkStatusBackFunctionEx	94
■ BiSetInkStatusBackWnd	95
■ BiSetInkStatusBackWndEx	96
■ BiCancelInkStatusBack	97
■ BiMICRSelectDataHandling	98
■ BiSCNSelectScanUnit	100
■ BiSCNSetImageTypeOption	101
■ BiSCNGetImageTypeOption	105
■ BiSCNMICRFunctionContinuously	106
■ BiSCNMICRFunctionPostPrint	115
■ BiSCNMICRFunction	124
■ BiSCNMICRCancelFunction	131
■ BiSCNDeleteCroppingArea	133
■ BiSCNSelectScanFace	134
■ BiGetPrnCapability	135
■ BiSCNGetImageFormat	136

■ BiSCNSetImageFormat.....	137
■ BiSCNGetScanArea	139
■ BiSCNSetScanArea	140
■ BiSCNGetImageQuality.....	142
■ BiSCNSetImageQuality	144
■ BiSCNSelectScanImage.....	146
■ BiSCNGetCroppingArea	147
■ BiSCNSetCroppingArea	148
■ BiSCNGetImageAdjustment	150
■ BiSCNSetImageAdjustment	152
■ BiMICRClearSpaces	154
■ BiSetOcrABAreaOrigin	155
■ BiGetMicrText	157
■ BiGetOcrABText	159
■ BiGetScanImage	161
■ BiGetScanImageSize.....	166
■ BiGetBarcodeData	167
■ BiDecodeBarcode.....	169
■ BiDecodeBarcodeMemory	170
■ BiSetMonInterval	172
■ BiGetPrintStation	173
■ BiSetPrintStation.....	174
■ BiGetPrintPosition	175
■ BiSetPrintPosition	176
■ BiSetEndorseDirection	178
■ BiBufferedPrint	180
■ BiGetPrintSize	181
■ BiGetPrintControl.....	182
■ BiSetPrintControl	183
■ BiSetPrintSize.....	184
■ BiGetPrintImageMethod	186
■ BiSetPrintImageMethod	187
■ BiPrintBarCode	188
■ BiPrintImage	192
■ BiGetPrintAlignment	194
■ BiSetPrintAlignment.....	195
■ BiPrintMemoryImage	196

■ BiSCNPrintMemoryImage	198
■ BiPrintText	200
■ BiSCNPrintText.....	202
■ BiPrintMultipleToneImage	204
■ BiUpdateEndorseText.....	205
■ BiInsertValidation	207
■ BiRemoveValidation	208
■ BiGetPrintCutSheetSettings	209
■ BiSetPrintCutSheetSettings	210
■ BiPrintCutSheet	211
■ BiAutoCutRollPaper	216
■ BiLoadTemplatePrintArea	217
■ BiTemplatePrint.....	218
■ BiSetTemplatePrintArea	220
■ BiClearTemplatePrintData	223
■ BiGetTransactionNumber.....	224
■ BiSetTransactionNumber	225
■ BiSetTransactionNumberWithIncremental.....	227
■ BiSetWaterfallMode.....	228
■ BiGetIQAResult	230
■ BiGetVersion	231
■ BiESCNEnable	233
■ BiESCNGetAutoSize	234
■ BiESCNSetAutoSize.....	235
■ BiESCNGetCutSize.....	236
■ BiESCNSetCutSize	237
■ BiESCNGetRotate	238
■ BiESCNSetRotate.....	239
■ BiESCNGetDeSkew.....	240
■ BiESCNSetDeSkew	241
■ BiESCNGetDocumentSize.....	242
■ BiESCNSetDocumentSize	243
■ BiESCNDefineCropArea	244
■ BiESCNGetMaxCropAreas.....	246
■ BiESCNSStoreImage.....	247
■ BiESCNClearImage	249
■ BiESCNRtrieveImage	251

■ BiESCNGetRemainingImages	253
■ BiSetBehaviorToScnResult	254
■ BiSetNumberOfDocuments	255
■ BiSelectErrorEjectAtContinuously	256
■ BiSelectJamDetect	258
■ BiMICRGetStatus	259
■ BiConfirmBufferedData	260
■ BiMICRCleaning	261
■ BiInkHeadCleaning	262
■ BiOpenDrawer	263
■ BiRingBuzzer	264
■ BiGetCounter	265
■ BiResetCounter	266
■ BiLoadAPISettings	267
■ BiSetConfigure	268
■ BiResetPrinter	270
■ BiCancelError	271
■ BiGetType	272
■ BiGetOfflineCode	273
■ BiGetOfflineCodeByIndex	274
■ BiSCNGetClumpStatus	275
■ BiSetPaperThickness	276
■ Structures	277
MF_BASE01	277
MF_SCAN	282
MF_MICR01	286
MF_OCR_AB	290
MF_PRINT01	294
MF_PROCESS01	298
MF_IQA	312
MF_IQA01	321
MF_IQA_RESULT	330
MF_IQA_RESULT01	341
MF_BARCODE	353
MF_DECORATE	360
MF_MICR	362
MF_PROCESS	363

■ Device Information	364
Device Status	364
Ink Status	367
Maintenance Counter	368
MICR Status	369
Offline Code	370
Device ID	381
Type ID	384

Compatible Information 385

■ API	385
■ Structures	390

Setting File 391

■ Method for setting the setting file.....	391
■ API settings file	391
Settings for API settings files.....	392
■ Template printing setting file	403
Settings for the template print setting file	403
■ Setting for the IQA function.....	405

Log Function..... 406

■ Overview	406
■ Settings for the log function.....	406
Log setting file settings.....	407
■ Log file output	408
Output destination for the log file	408
Log file name	408
Output example	409

Appendix..... 410

Overview

This chapter describes the features and specifications of the product.

Features

TM-S9000II, TM-S2000II, TM-S9000 and TM-S2000 provides the following features through the TM-S9000/S2000 API:

Image Scanning

- Capability to retrieve color image data (check sheet/ID card).
- The TM-S9000II, TM-S2000II, TM-S9000 and TM-S2000 has two light sources, an RGB one and an infrared one, and can use both simultaneously to retrieve the image.
For the TM-S2000II UV model and the TM-S2000 UV model, the UV light source can be also selected for front side scanning.
- Capability to scan both the front and the back of a sheet at the same time.
- Capability to save image data with the resolution and file format specified.
- Capability to process scanned image data (auto size, skew correction).
- Capability to recognize OCR-A/B font.
- Capability to retrieve barcode data.
- Capability to perform IQA (Image Quality Assurance) analysis of image data. IQA conforms to the recommendations of FSTC (Financial Services Technology Consortium) and CTS2010 in India. (IQA cannot be used in a card scan.)

Printing

- Endorsement printing on the check sheet as well as Cut sheet paper printing can be performed by the ink-jet head.
- Electronic endorsement is supported, by which text data and image data can be added to pictorial data.
- The style can be specified for the characters to be printed.
- Roll paper printing is supported. (TM-S9000II and TM-S9000)
- Template printing is supported.

MICR

- Capable of reading E13B/CMC7 font.

Device operation

- The device provides notification of processing by means of a buzzer.
- The drawer can be opened.
- Sorting documents into 2 pockets depending on the scan result.
(Using two pockets is an optional specification.)

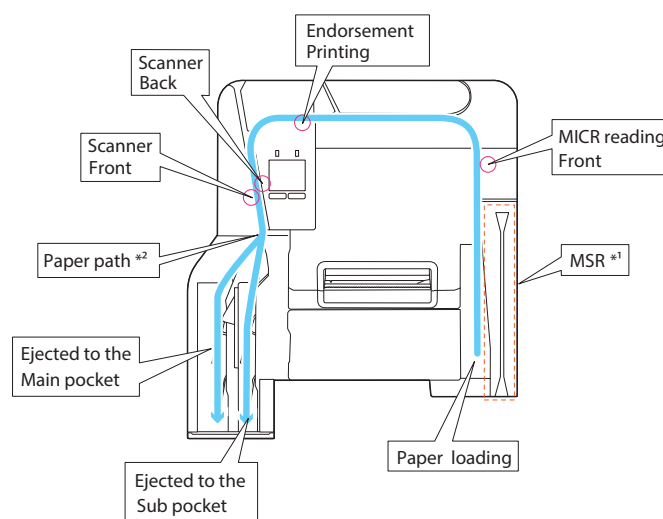
Structure

The following describes the locations of the main functions and the flow of processing.

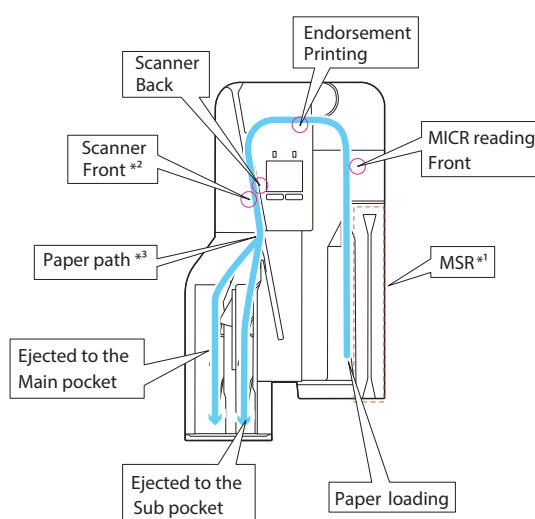
- Check scan ([p.13](#))
- Card scan ([p.14](#))
- Cut sheet printing ([p.15](#))

Check scan

TM-S9000II and TM-S9000



TM-S2000II and TM-S2000



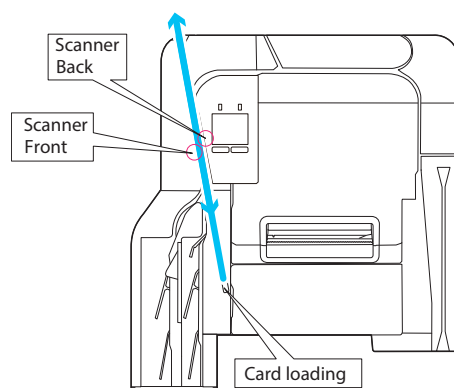
*1: The MSR is an option. When you want to use it, use utilities for processing. The TM-S9000/S2000 API doesn't support it.

*2: The TM-S2000II UV Model and the TM-S2000 UV Model can also use the UV light source for scanning.

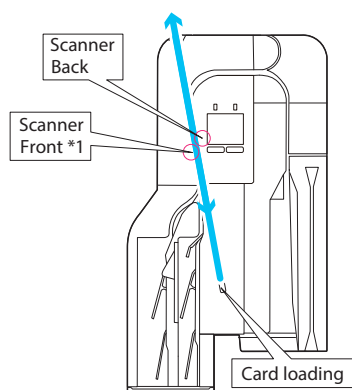
*3: Pocket sorting is possible with the two pocket specification. Two pocket specification is an option.

Card scan

TM-S9000II and TM-S9000



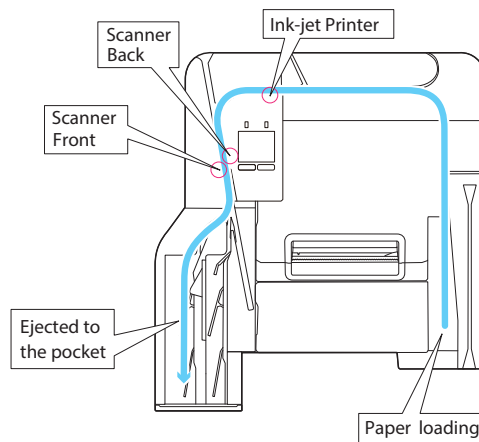
TM-S2000II and TM-S2000



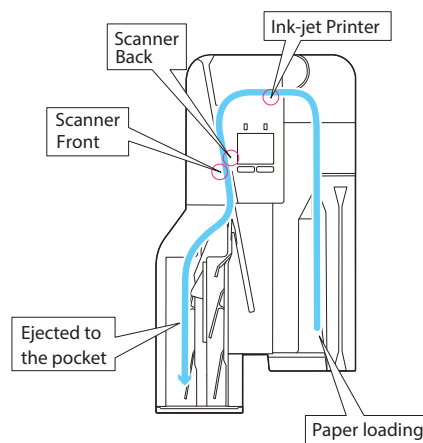
*1: The TM-S2000II UV Model and the TM-S2000 UV Model can also use the UV light source for scanning.

Cut sheet printing

TM-S9000II and TM-S9000



TM-S2000II and TM-S2000

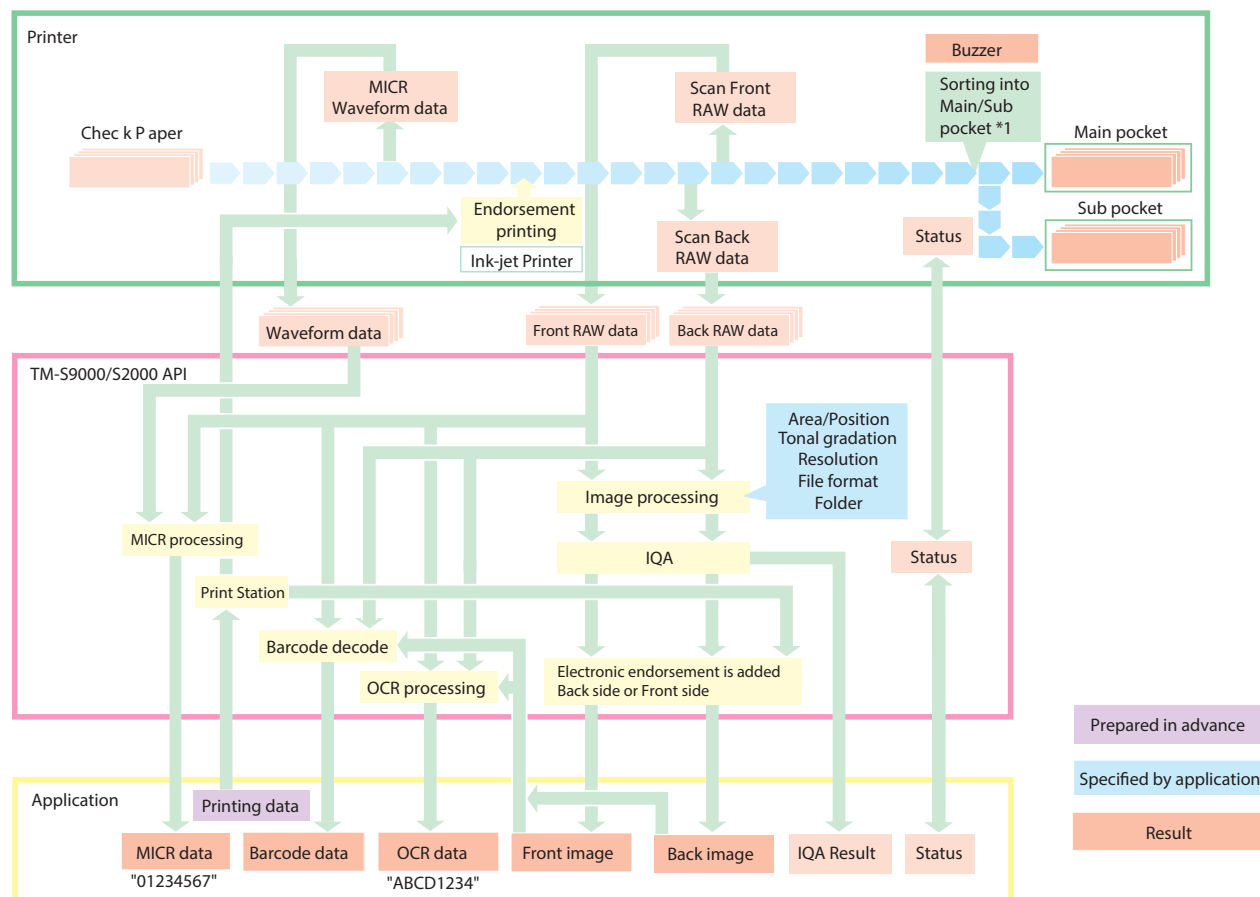


<Regarding the front and back of a sheet of paper>

If processing cut sheets, the back side of the sheet will be reversed when doing a check scan. This is because the ink-jet printer is on the inside of the paper's path. When you set for cut sheets, please set the side to print to as the inner side (back side), and the side not printed to as the outer side (front side).

Features of the TM-S9000/S2000 API

The figure below shows the flow of document reading, data processing, and functions. Specifying on an application can obtain various results.



*1: Pocket sorting is possible with the two pocket specification.

Functions	Side		Specify/Prepare	Result
	Front	Back		
Front image Scanning	✓	-	<ul style="list-style-type: none"> Specify the station. Specify the light source (RGB/ Infrared/UV). The UV light source is available only on the TM-S2000II UV model and TM-S2000 UV model. Specify the resolution. Specify the read area. Specify the file format. 	<ul style="list-style-type: none"> Image data Barcode data OCR data
Back image Scanning	-	✓	<ul style="list-style-type: none"> Specify the station. Specify the light source (RGB/ Infrared). Specify the resolution. Specify the read area. Specify the file format. 	<ul style="list-style-type: none"> Image data Barcode data OCR data
MICR reading	✓	-	Font to be read	MICR text data

Functions	Side		Specify/Prepare	Result
	Front	Back		
Physical endorsement	-	✓	Prepare print data (text/image).	Data is printed on the back side of a check sheet.
Cut sheet paper printing	-	✓	<ul style="list-style-type: none"> • Prepare print data. • Set the Cut sheet paper with the print side facing inward. 	Data is printed on the print side.
Electronic endorsement	✓	✓	<ul style="list-style-type: none"> • Turn ON or OFF data appending. • Prepare electronic endorsement data. 	Image file with electronic endorsement appended
Check sheet sorting	-		Specify the sorting conditions.	Check sheets are sorted into the main or sub pockets.
Buzzer	-		Specify the beeping conditions for the buzzer.	Buzzer beeps.
Roll paper printing (TM-S9000II and TM-S9000)	-		Prepare print data (text/image/barcode). Turn ON or OFF auto cut.	Roll paper printing is performed, followed by the execution of an auto cut.

Operating Environment

OS

- Microsoft Windows 11 (64 bit)
- Microsoft Windows 10 (32/64 bit)
- Microsoft Windows 8.1 (32/64 bit)
- Microsoft Windows 8 (32/64 bit)
- Microsoft Windows 7 SP1 (32/64 bit)
- Microsoft Windows Server 2022
- Microsoft Windows Server 2019
- Microsoft Windows Server 2016
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2012
- Microsoft Windows Embedded POS Ready 7 (32/64bit)
- Microsoft Windows Embedded 8 Standard (32/64bit)
- Microsoft Windows 10 IoT Enterprise (2019 LTSC, CBB) (32/64bit) *Not support ARM

Computer

225 dpm model / 200 dpm model

CPU:	At least Intel Celeron 3205U 1.5 GHz or the equivalent
Memory:	At least 1 GB or above the minimum operating system requirement
HDD:	At least 30 MB Free space. (Before installing the driver)

130 dpm model / 110 dpm model

CPU:	At least Pentium 4 2.0 GHz or the equivalent
Memory:	At least 512 MB or above the minimum operating system requirement
HDD:	At least 30 MB Free space. (Before installing the driver)

Interface

USB 2.0

Development Environment

Development Tool	Development Language
Visual Studio 6.0	Visual C++ 6.0
Visual Studio .NET 2003	Visual Basic .NET 2003
	Visual C++ .NET 2003
	Visual C# .NET 2003
Visual Studio 2005 or later	Visual Basic 2005 or later
	Visual C++ 2005 or later
	Visual C# 2005 or later

Technical Support Web Site

You can obtain software and manuals from one of the following URLs.

For customers in North America, go to the following web site.

<https://www.epson.com/support/>

For customers in other countries and regions, go to the following web site.

<https://download.epson-biz.com/?service=pos>

Install and Uninstall

This chapter describes how to install and uninstall the TM-S9000/S2000 driver.

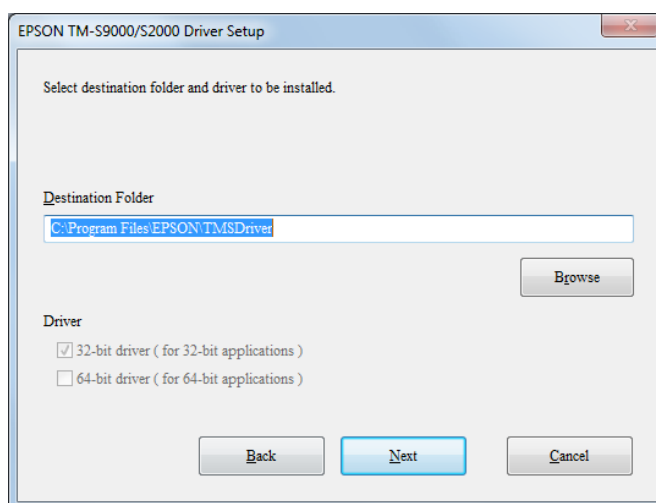
Installation

Install the TM-S9000/S2000 driver by using the following steps:



- When installing, please log on to the computer as the Administrator.
- When updating TM-S9000/S2000 driver, install the new driver without uninstalling the existing old driver. The old driver is automatically uninstalled.

- 1 Open the compressed file you procured.
- 2 Double-click the "Setup.exe" icon.
- 3 If the "User Account Control" screen appears, click [Yes].
- 4 The "Welcome" screen appears. Click [Next].
- 5 The "License Agreement" screen appears. After reviewing the License Agreement, check [I accept the terms and conditions of this Agreement.], and click [Next].
- 6 The "Installation Settings" screen appears. Specify the installation destination and the driver to be installed, then click [Next].

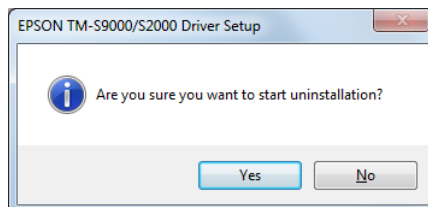


- 7 The "Confirmation" screen appears. Click [Next]. Installation begins.
- 8 The "Complete" screen appears. Click [Finish].
- 9 Connect the TM-S9000II, the TM-S2000II, the TM-S9000, and the TM-S2000 to the computer.

Uninstallation

Uninstall the TM-S9000/S2000 driver by using the following steps:

- 1** Finish other operations on the computer.
- 2** Open [Uninstall a program] or [Add or Remove Programs].
 - On Windows 10:
[Start]-[Settings]-[Apps] (or [System])-[Apps & features]
 - On Windows 8.1/ Windows 8:
[Desktop]-[Settings]-[Control Panel]-[Uninstall a program]
 - 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]
- 3** Select [EPSON TM-S9000/S2000 Driver Version x.xx [xx-bit]], and click [Uninstall/Change].
- 4** The "Welcome" screen appears. Click [Next].
- 5** The following screen appears. Click [Yes].



- 6** The "Complete" screen appears. Click [Finish].



The Windows restart screen is displayed. If you will restart your computer later, uncheck "I want to restart my computer now."

- 7** If you unchecked "I want to restart my computer now" in step 6 when the Windows restart screen was displayed, please restart your computer.

Silent Install

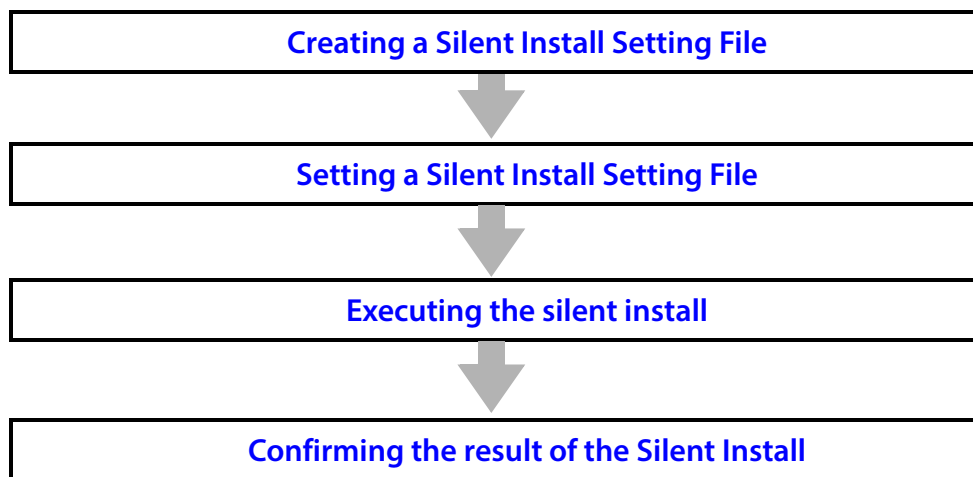
The Silent Install refers to the automatic installation of the TM-S9000/S2000 driver when the user installs an application.

This is done by executing a command and incorporating setup.exe of the TM-S9000/S2000 Driver and the Silent Install Setting file (SInst.ini) recorded during the TM-S9000/S2000 driver installation procedure into the installer of the application.

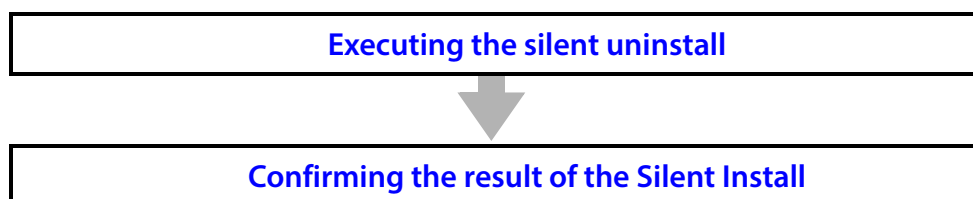


When performing the silent install on a client computer, you must have logged on to the computer as an Administrator. The installation cannot be made if you have logged on as a user.

Flow of Silent Install



Flow of Silent Uninstall



Creating a Silent Install Setting File

You can create a Silent Install Setting file (SInst.ini) attaching startup parameters to Setup.exe and performing the installation. After the installation is completed, it is created in the same folder as Setup.exe.



If there is already a Silent Install Setting file in the destination folder, it is overwritten.

Startup parameters

`"[Setup.exe (Full Path)]" [/r]`

Example

`"%USERPROFILE%\Desktop\Temp\Setup.exe" /r`

Setting a Silent Install Setting File

The created Silent Install Setting file (SInst.ini) can perform the following settings.

Section name	
SilentInstall	
Key name	Details
InstallDir	Sets the destination for the TM-S9000/S2000 driver. < Setting value > Please specify any folder.
Install32	Specifies if installing the 32-bit driver is necessary. If nothing is set, it operates the same as if set to 1. Or, if installed on a 32-bit OS, the 32-bit driver will be installed even if 0 is specified. The initial value is 1. < Setting value > 0: Do not install the 32-bit driver. 1: Install the 32-bit driver.
Install64	Specifies if installing the 64-bit driver is necessary. If nothing is set, it operates the same as if set to 1. If installed on a 64-bit OS, even if 0 is specified, the 64-bit driver is installed. Or, if installed on a 32-bit OS, the 64-bit driver will not be installed even if 1 is specified. The initial value is 1. < Setting value > 0: Do not install the 64-bit driver. 1: Install the 64-bit driver.

Executing the silent install

Execute Setup.exe with following startup parameter. Please put the Silent Install Setting file (SInst.ini) into the same folder as Setup.exe beforehand, and then execute the command.

Startup parameters

"[Setup.exe (specified with the full path)]" [/s]

Example

"%USERPROFILE%\Desktop\Temp\Setup.exe" /s



If you want to put the Silent Install Setting file (SInst.ini) into a folder of your choosing and perform a Silent Install, execute it using this startup parameter.

< Setting value >

"[Setup.exe (Full Path)]" [/s "(Full Path)"]

<Example>

"%USERPROFILE%\Desktop\Temp\Setup.exe" /s "%USERPROFILE%\Desktop\Temp\SInst.ini"

Optional parameters

<Setting non-display in [Add or Remove Programs]>

When the EPSON TM-S9000/S2000 driver is installed by means of silent install, "TM-S9000/S2000 API" will appear in [Add or Remove Programs] ([Uninstall a Program] in Windows Vista/ Windows 7/ Windows 8), in the same way as when it is installed normally. If you want to prevent it from appearing in [Add or Remove Programs], execute the installer with the following command. (Setting non-display can prevent the user from accidentally deleting the driver.)

"[Setup.exe (specified with the full path)]" [/s] [/z"Invisible"]

Example

"%USERPROFILE%\Desktop\Temp\Setup.exe" /s /z"Invisible"

Executing the silent uninstall

Execute Setup.exe with following startup parameter.



- When executing a Silent Uninstall, confirm that the TM-S9000/S2000 driver is installed beforehand.
- Check beforehand that the Setup.exe is the same version as the installed TM-S9000/S2000 driver. If Setup.exe is a different version, instead of being uninstalled, the TM-S9000II, the TM-S2000II, the TM-S9000 and the TM-S2000 is upgraded.

Startup parameters

"[Setup.exe (specified with the full path)]" [/s /u]

Example

"%USERPROFILE%\Desktop\Temp\Setup.exe" /s /u

Optional parameters

<Uninstalling the driver forcibly>

To uninstall the EPSON TM-S9000/S2000 Driver forcibly, add the following command to the installer.

"[Setup.exe (specified with the full path)]" [/z"uninstall"]

Example

"%USERPROFILE%\Desktop\Temp\Setup.exe" /z"uninstall"

Confirming the result of the Silent Install

The result of executing the Silent Install (including the creation of the Silent Install Setting file / Silent Uninstall) is output in the Silent Install log (SInstLog.txt). If there was a failure in execution, you can check the cause in the Silent Install log.

The Silent Install log is output to the folder where the Setup.exe which executed the install is kept.



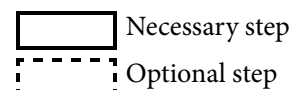
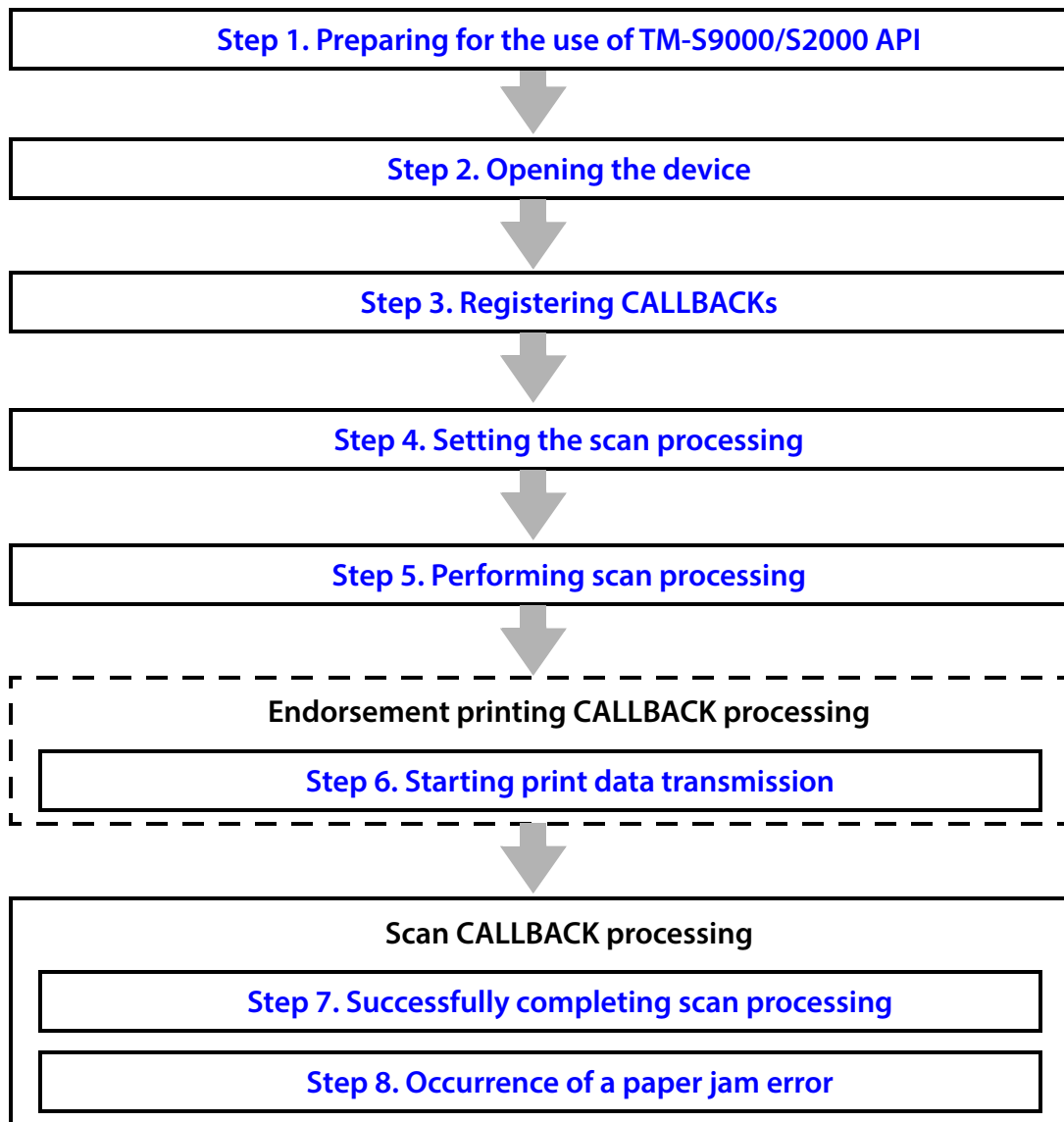
The Silent Install log is not created in cases where the file cannot be created in the same folder as Setup.exe, for example, if Setup.exe is kept on a CD-ROM,

Programming Guide

This chapter describes a programming method for the development of application using TM-S9000/S2000 API.

Application Processing Steps

This section describes basic processing steps of applications using the TM-S9000/S2000 API.



Step 1. Preparing for the use of TM-S9000/S2000 API

Load the following DLL (dynamic link library):

- EpsStmApiWrapper.dll



Header file (EpsStmApiInterface.h/MultiFunction.h) included in the sample program.

Step 2. Opening the device

Call BiOpenMonPrinter ([page 68](#)). Retrieve the connected TM-S9000II, TM-S2000II, TM-S9000 and TM-S2000 to enable communication with the application.

Step 3. Registering CALLBACKs

The TM-S9000/S2000 API has five types of callback features.

CALLBACK for scan processing

This CALLBACK function is invoked when a scan processing is started/completed, when data reception is started/completed, or when a paper jam error occurs.

Register the CALLBACK with BiSCNMICRSetStatusBackFunction ([page 71](#)).

You can cancel this CALLBACK function by using BiSCNMICRCancelStatusBack ([page 77](#)).

CALLBACK for starting endorsement printing

This CALLBACK function is invoked when the device is ready to receive endorsement print data.

This CALLBACK function passes endorsement print data. Register the CALLBACK function with BiStartEndorsementSetStatusBackFunction ([page 78](#)). You can cancel this CALLBACK function by using BiStartEndorsementCancelStatusBack ([page 80](#)).

CALLBACK for completion of endorsement printing

This CALLBACK function is invoked when the device is no longer capable of receiving endorsement print data.

Register the CALLBACK function with BiEndEndorsementSetStatusBackFunction ([page 81](#)).

You can cancel this CALLBACK function by using BiEndEndorsementCancelStatusBack ([page 84](#)).

CALLBACK for device status

This CALLBACK function is invoked when the product sensor status changes.

Register the CALLBACK function with BiSetStatusBackFunctionEx ([page 88](#)). You can cancel this CALLBACK function by using BiCancelStatusBack ([page 91](#)).

CALLBACK for ink status

This CALLBACK function is invoked at the time of ink cartridge replacement, to provide notification of presence/absence of an ink cartridge, and at the time of head cleaning.

Register the CALLBACK function with `BiSetInkStatusBackFunctionEx` ([page 94](#)). You can cancel this CALLBACK function by using `BiCancelInkStatusBack` ([page 97](#)).

Step 4. Setting the scan processing

This step is used to set scan processing operation.

Setting the scan unit

Call BiSCNSelectScanUnit ([page 100](#)) to set the scan unit (check sheet/ID card).

Setting the structure

Set values for each structure. The structures to be set are as follows:

Structure	Setting
MF_BASE01	Operation setting for scan processing
MF_SCAN	Scan setting (resolution, the size of a string to be appended, etc.)
MF_MICR01	<ul style="list-style-type: none"> MICR setting (E13B/CMC7, etc.) OCR setting
MF_PROCESS01	<ul style="list-style-type: none"> Selection of processing mode (High speed/Confirmation) Setting the operation assignment to be performed in the event of error detection (double feed/insertion orientation error/noise/bad data) Setting the operation to be performed in the event of pocket near-full

Structures that can be set for each procedure

- ✓: Structure that requires settings. If you execute processing without setting it properly, an ERR_PARAM is returned.
- ✗: Based on conditions, a ERR_PARAM might come back.
- ✕: Structure that does not need settings. Processing is performed regularly without settings.
- : Structure that ignores settings. Processing is performed regularly whether there are settings or not.

Structure	Check Scan (page 100)	Card Scan (page 100)	Print Cutsheet (page 211)
MF_BASE01	✓	✓	✓
MF_SCAN (Front)	✗ ^{*1}	✗ ^{*1}	When execute scan: ✗ ^{*3} When not scan: -
MF_SCAN (Back)	✗ ^{*2}	✗ ^{*2}	When execute scan: ✗ ^{*3} When not scan: -
MF_PRINT01	✕	-	(Not setting)
MF_MICR01	✕	-	(Not setting)
MF_PROCESS01	✕	✗ ^{*4}	✗ ^{*5}

*1: If a bit from MF_MICR_USE_OCR is in the bMicOcrSelect of the MF_MICR01, an ERR_PARAM is returned.

*2: An ERR_PARAM is returned if the settings for dwEndorseType of MF_PRINT01 are different from those below.

- MF_PRINT_TYPE_ELECTRIC_ENDORSE_ONLY
- MF_PRINT_TYPE_ELECTRIC_ENDORSE_EXTEND
- MF_PRINT_TYPE_ENDORSE_NORMAL
- MF_PRINT_TYPE_ENDORSE_EXTEND

*3: An ERR_PARAM is returned if both the front and the back are not set.

*4: Settings other than `dwStartWaitTime` and `bResultPartialData` are ignored.

*5: Some members can be set.

For more information, refer to `BiPrintCutSheet` ([page 211](#)).

Setting procedure

Set each structure using the following functions:

- `BiSCNMICRFunctionContinuously` ([page 106](#))
- `BiSCNMICRFunctionPostPrint` ([page 115](#))

Setting the light source

Call `BiSCNSetImageTypeOption` ([page 101](#)) to set the light source to be used for scan processing.

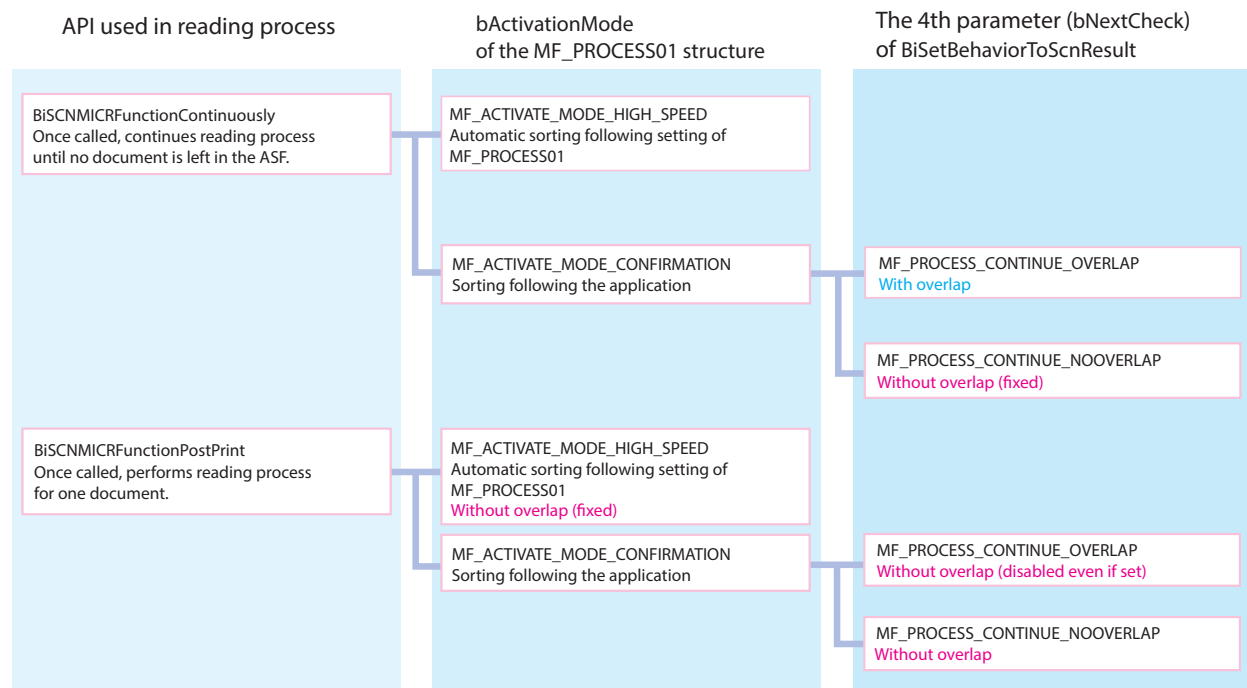
Step 5. Performing scan processing

After the retrieval of the transaction ID, scan processing is initiated by the API that is appropriate to processing mode.

Processing mode	API	Description
High speed mode	BiSCNMICRFunctionContinuously	The check sheets set in the ASF are automatically sorted into pockets without being stopped.
Confirmation mode	<ul style="list-style-type: none"> BiSCNMICRFunctionContinuously BiSCNMICRFunctionPostPrint 	Each one of the check sheets is stopped in mid-course, and the application decides which pocket they are to be sorted into.

Once the scan processing starts, the reading of both sides of the check sheet as well as MICR reading is performed simultaneously.

Basic configuration of the scan processing API



During processing with overlap, two documents may be in the paper path.
The transaction number tells you which document is jammed when a paper jam occurs.



Overlap will not be realized even if `BiSCNMICRFunctionPostPrint` is executed repeatedly.

Step 6. Starting print data transmission

Execute physical endorsement.

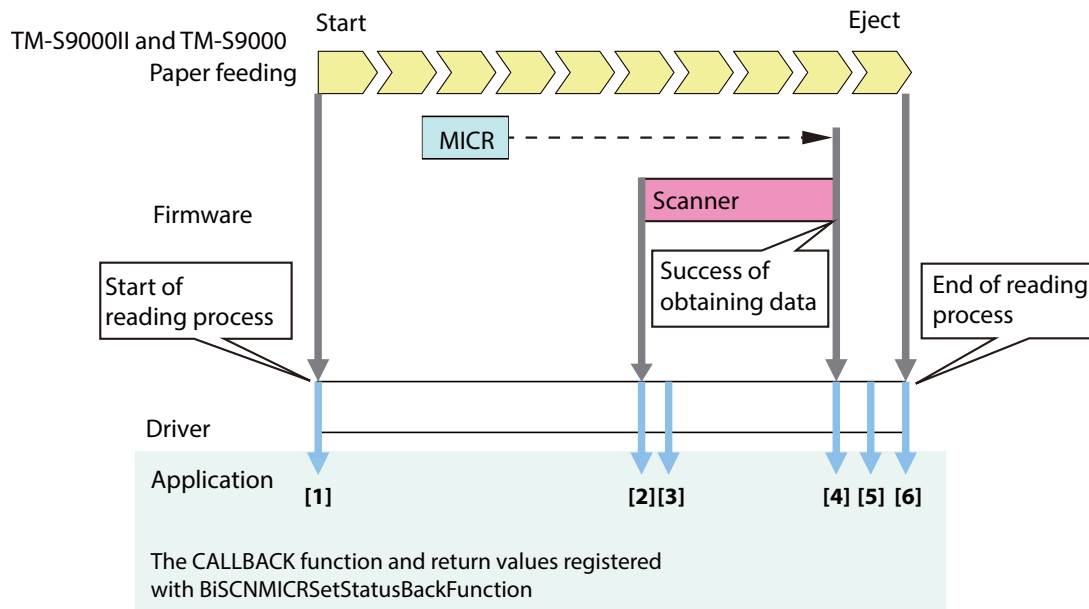
By using `BiSetPrintStation` ([page 174](#)), set the print station for physical endorsement.

Create print data, and then execute printing.

Step 7. Successfully completing scan processing

Event notification and return values

If scan processing completes successfully, events (processing status) and return values are returned to the CALLBACK in the following order:



Event	Processing status	Return value
[1]. Start of scan processing	MF_FUNCTION_START	SUCCESS
[2]. Start of paper feeding	MF_CHECKPAPER_PROCESS_START	SUCCESS
[3]. Start of data reception	MF_DATARECEIVE_START	SUCCESS
[4]. Completion of data reception	MF_DATARECEIVE_DONE	SUCCESS
[5]. Completion of paper feeding	MF_CHECKPAPER_PROCESS_DONE	SUCCESS
[6]. Completion of scan processing	MF_FUNCTION_DONE	SUCCESS

Processing inside the CALLBACK function

After the data reception completion event (MF_DATARECEIVE_DONE) is confirmed, necessary processing will be executed inside the CALLBACK function in the following order:

- 1** MICR data is retrieved.
- 2** Image data is retrieved with the image data format specified.
- 3** In case of Confirmation mode, the ejection destination pocket as well as the presence/absence of overlap is specified.

Step 8. Occurrence of a paper jam error

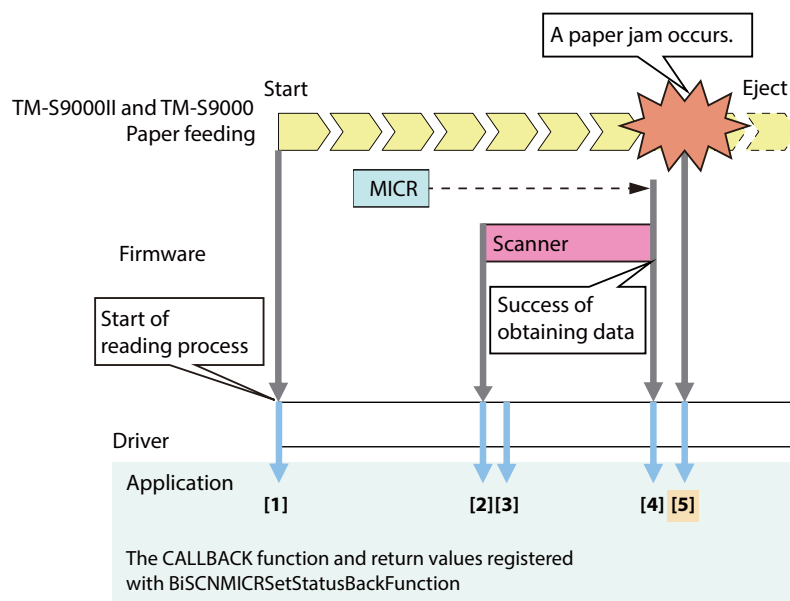
If the paper position detected is inappropriate with respect to the paper feed amount, a paper jam error occurs. The event (processing status) and the return value differ depending on the timing of the occurrence of the paper jam error and as well as on whether the read data is to be discarded or retrieved.

Three patterns of processing are described below. A description of the solution for fixing a paper jam error is also provided.

- "If a paper jam error occurs after the completion of data reception:" on page 35
- "If a paper jam error occurs during the process of retrieving data, and if the read data is to be discarded:" on page 36
- "If a paper jam error occurs during the process of retrieving data, and if the read data is to be retrieved:" on page 37
- "If a paper jam error occurs after the completion of data reception:" on page 35

If a paper jam error occurs after the completion of data reception:

Timing at which a paper jam error occurs:



Event notification and return values

Event	Processing status	Return value
[1]. Start of scan processing	MF_FUNCTION_START	SUCCESS
[2]. Start of paper feeding	MF_CHECKPAPER_PROCESS_START	SUCCESS
[3]. Start of data reception	MF_DATARECEIVE_START	SUCCESS
[4]. Completion of data reception	MF_DATARECEIVE_DONE	SUCCESS
[5]. Completion of scan processing	MF_FUNCTION_DONE	ERR_PAPER_JAM

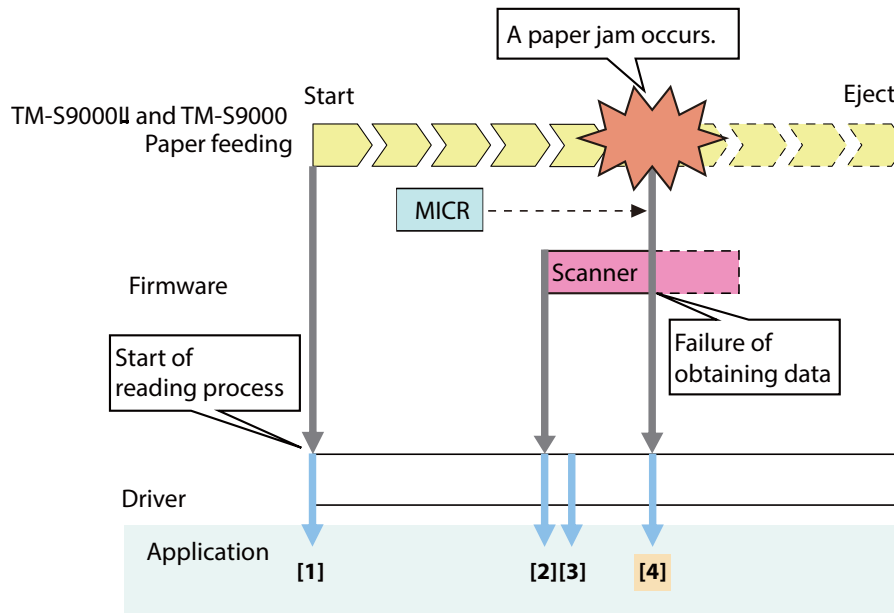


The following event will not be notified if a paper jam error occurs after the completion of data reception:

- Completion of paper feeding (MF_CHECKPAPER_PROCESS_DONE)

If a paper jam error occurs during the process of retrieving data,
and if the read data is to be discarded:

Timing at which a paper jam error occurs:



Event notification and return values

Event	Processing status	Return value
[1]. Start of scan processing	MF_FUNCTION_START	SUCCESS
[2]. Start of paper feeding	MF_CHECKPAPER_PROCESS_START	SUCCESS
[3]. Start of data reception	MF_DATARECEIVE_START	SUCCESS
[4]. Completion of scan processing	MF_FUNCTION_DONE	ERR_PAPER_JAM



If a paper jam error occurs after the completion of data reception, the following events will not be notified:

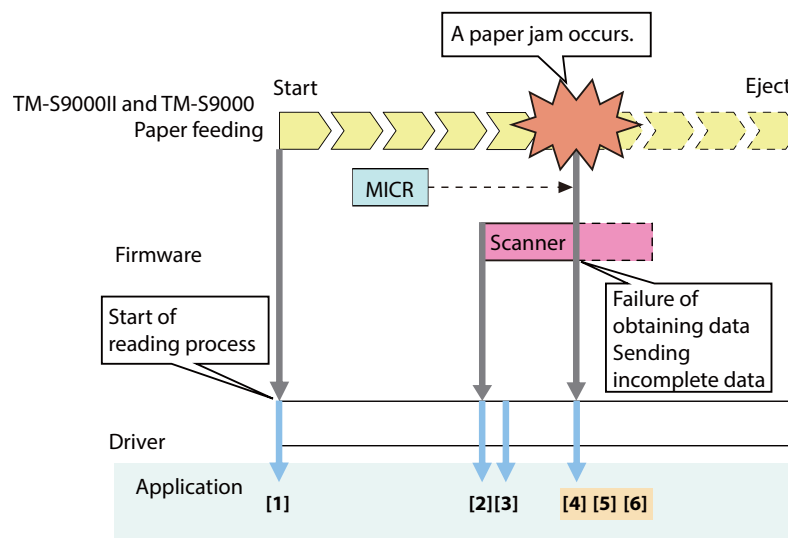
- Completion of data reception (MF_DATARECEIVE_DONE)
- Completion of paper feeding (MF_CHECKPAPER_PROCESS_DONE)

If a paper jam error occurs during the process of retrieving data, and if the read data is to be retrieved:

Setting to be specified before scan processing

To retrieve the data read up to the point of a paper jam error, MF_RESULT_PARTIAL must be specified in advance for bResultPartialData of the MF_PROCESS01 structure.

Timing at which a paper jam error occurs:



Event notification and return values

An error occurrence event (MF_ERROR_OCCURRED) is notified when a paper jam error occurs, before the notification of a data reception event (MF_DATARECEIVE_DONE) and the retrieval of data.

Event	Processing status	Return value
[1]. Start of scan processing	MF_FUNCTION_START	SUCCESS
[2]. Start of paper feeding	MF_CHECKPAPER_PROCESS_START	SUCCESS
[3]. Start of data reception	MF_DATARECEIVE_START	SUCCESS
[4]. Occurrence of an error	MF_ERROR_OCCURRED	ERR_PAPER_JAM
[5]. Completion of data reception	MF_DATARECEIVE_DONE	SUCCESS
[6]. Completion of scan processing	MF_FUNCTION_DONE	ERR_PAPER_JAM



The following event will not be notified if a paper jam error occurs after the completion of data reception:

- Completion of paper feeding (MF_CHECKPAPER_PROCESS_DONE)

How to recover from a paper jam error

Firstly, open a cover and remove the check sheet remaining in the path. Then, call BiCancelError from the application to recover from the paper jam error.

You can view a message prompting the removal of jammed paper through the CALLBACK function that indicates the device status (See "[CALLBACK for device status](#)" on page 28) or by checking the sensor status using BiGetStatus.

Please refer to the troubleshooting section of the user's manual for details about dealing with a paper jam error.

Programming flow

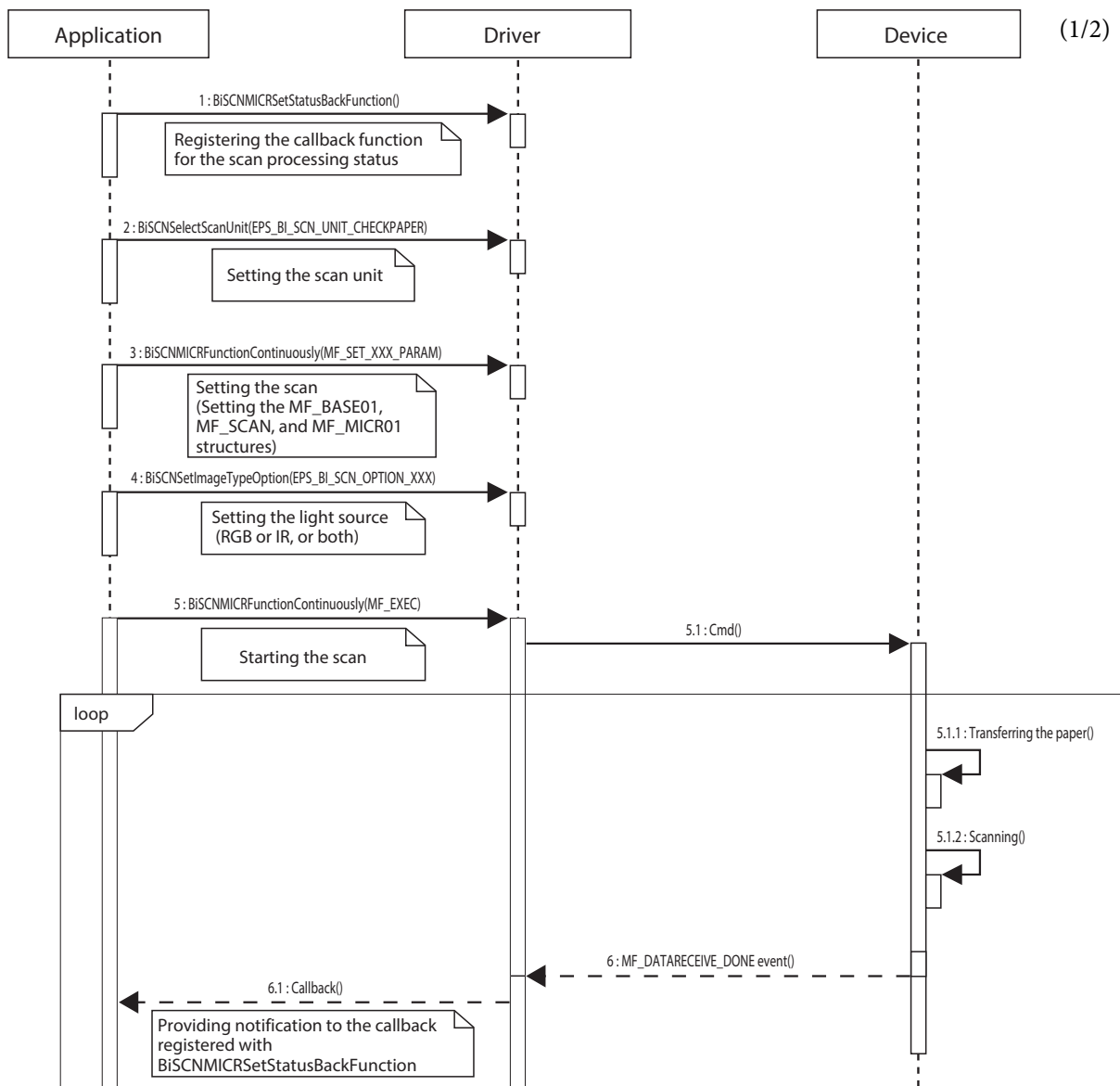
This section describes how to program the features offered by the TM-S9000/S2000 API by using sequence diagrams.

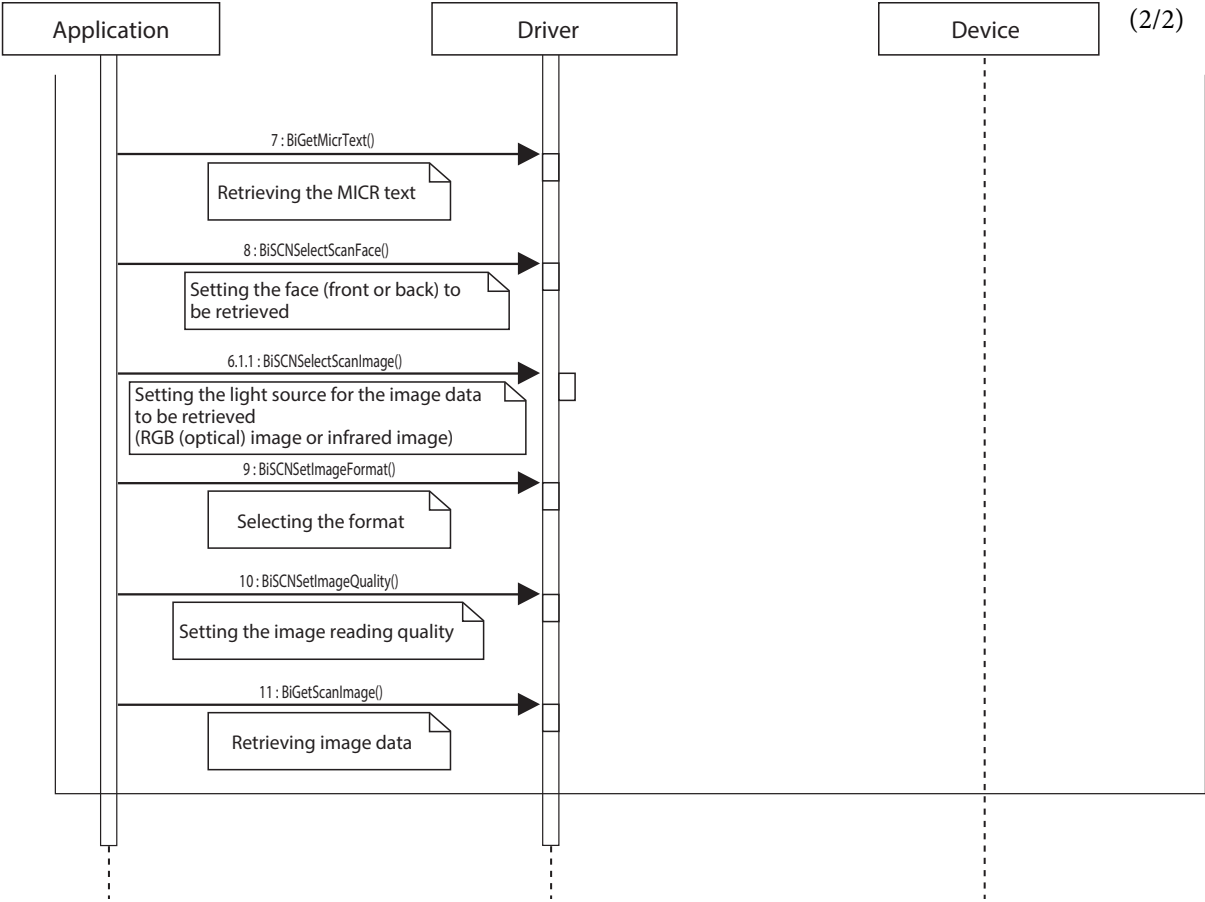


Information on DLL loading, BiOpenMonPrinter and BiCloseMonPrinter are not provided here.

Check scan processing

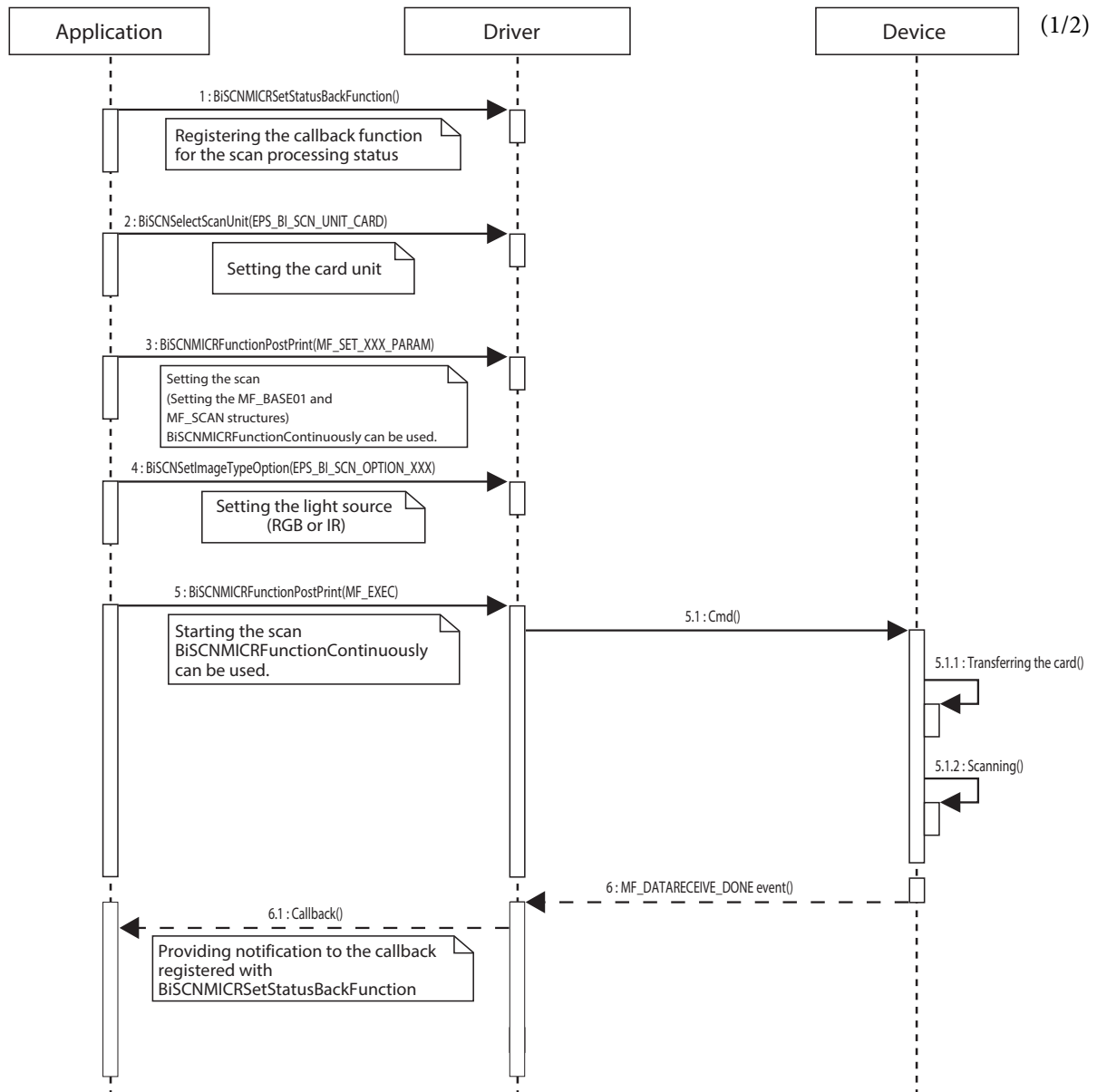
This is the process for scanning the check sheet and retrieving the scan results (MICR data, image data).



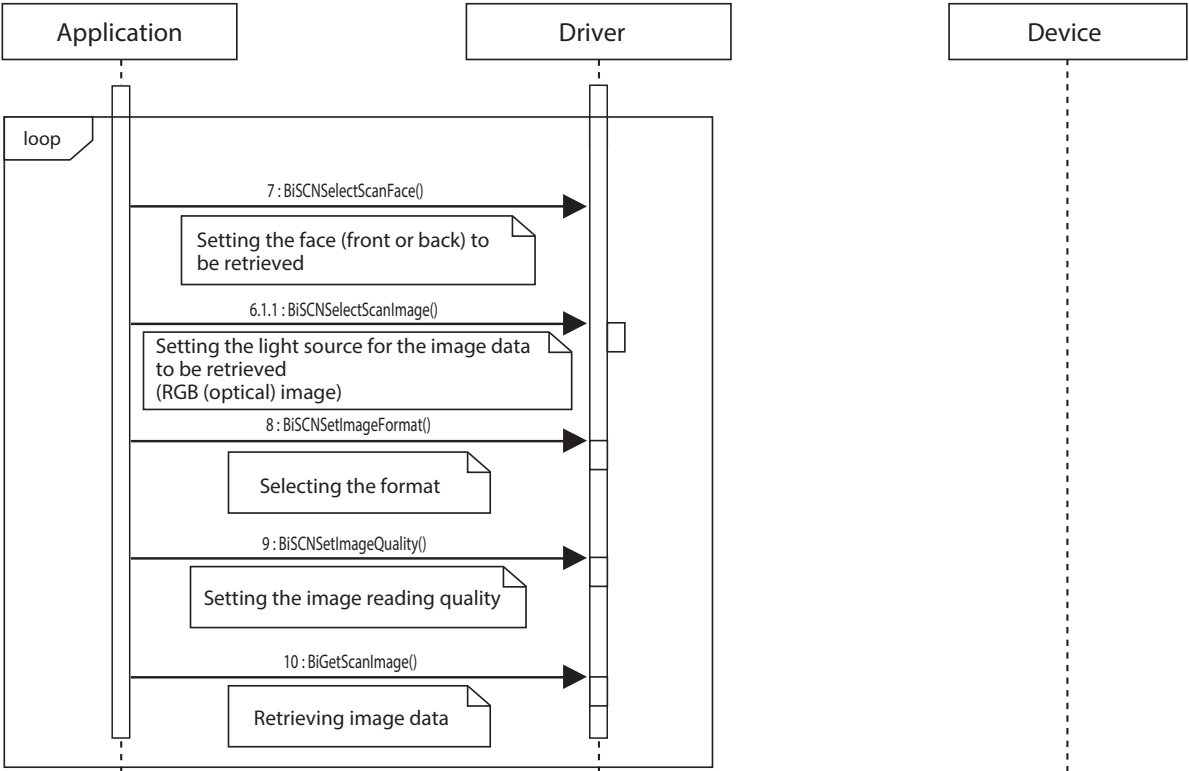


Card scan processing

This is the process for scanning a card and retrieving the scan results (image data).

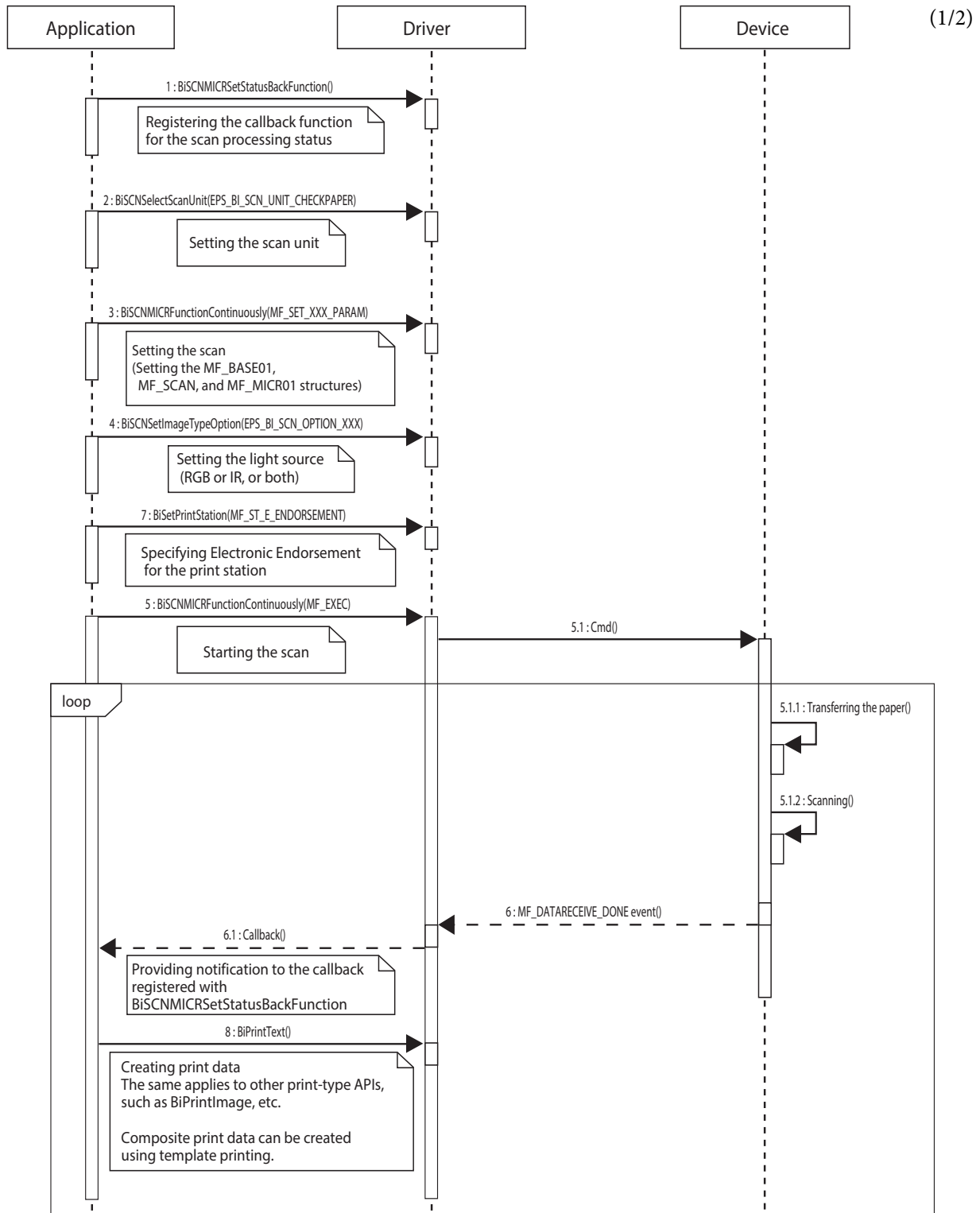


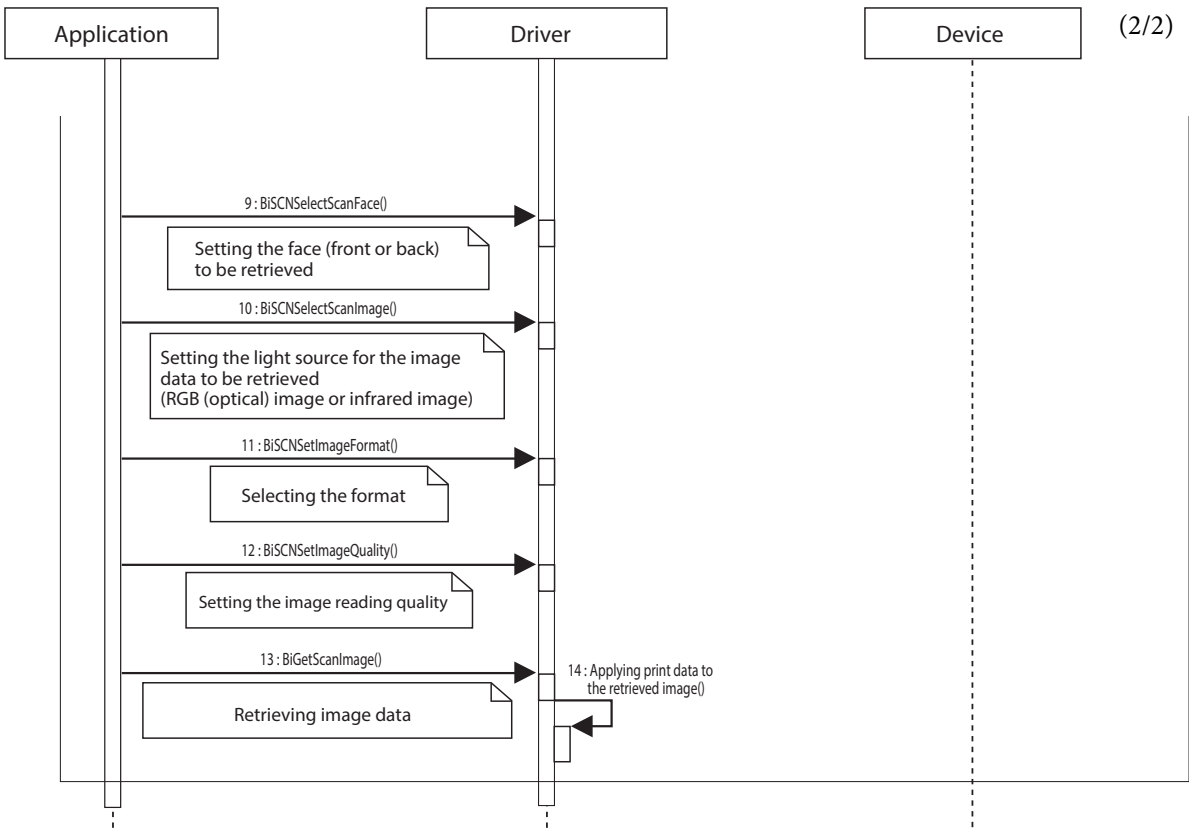
(2/2)



Electronic endorsement printing

This is the process for scanning the check sheet and adding an electronic endorsement to retrieve the scan results (image data).

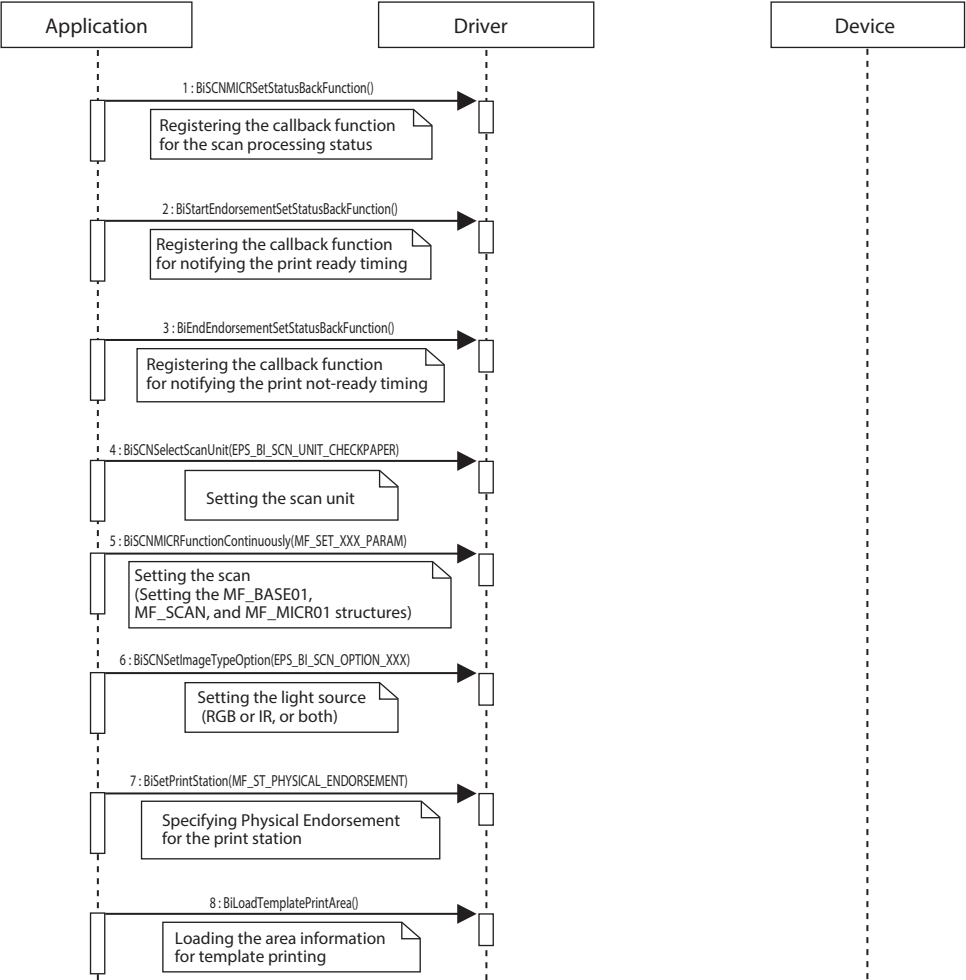


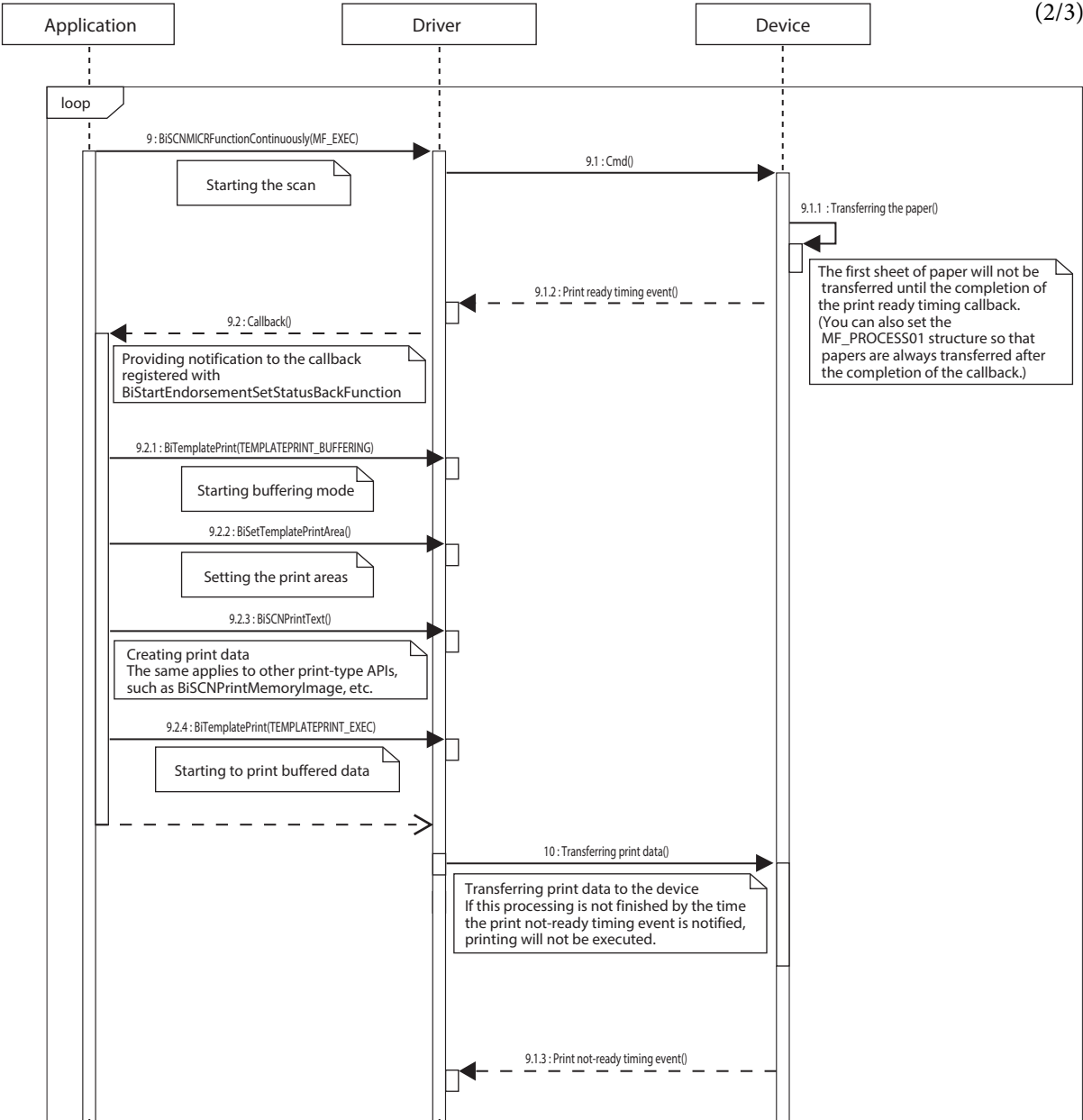


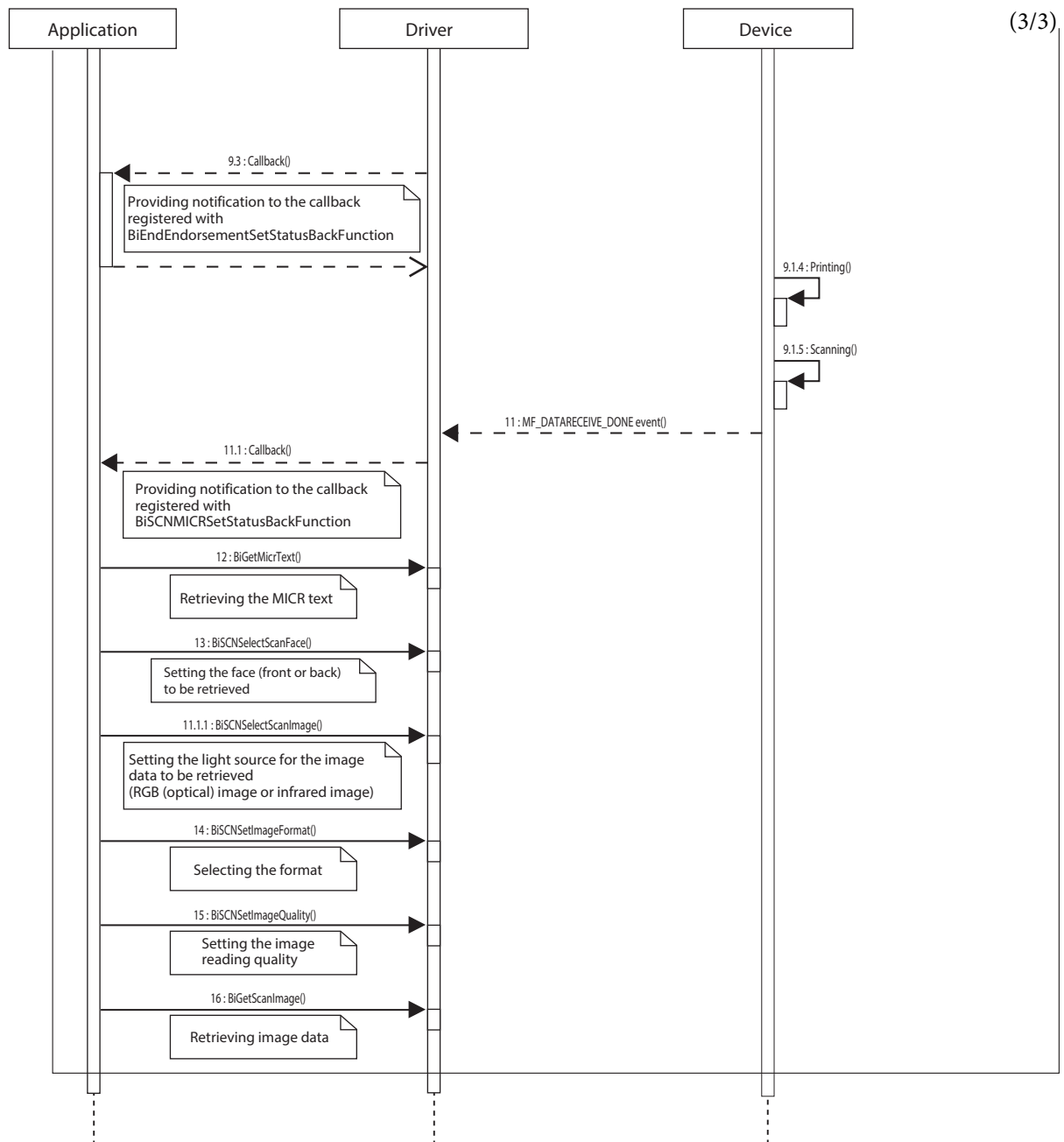
Physical endorsement printing

This is the process for applying a physical endorsement and then scanning the check sheet, and then retrieving the scan results (image data).

(1/3)

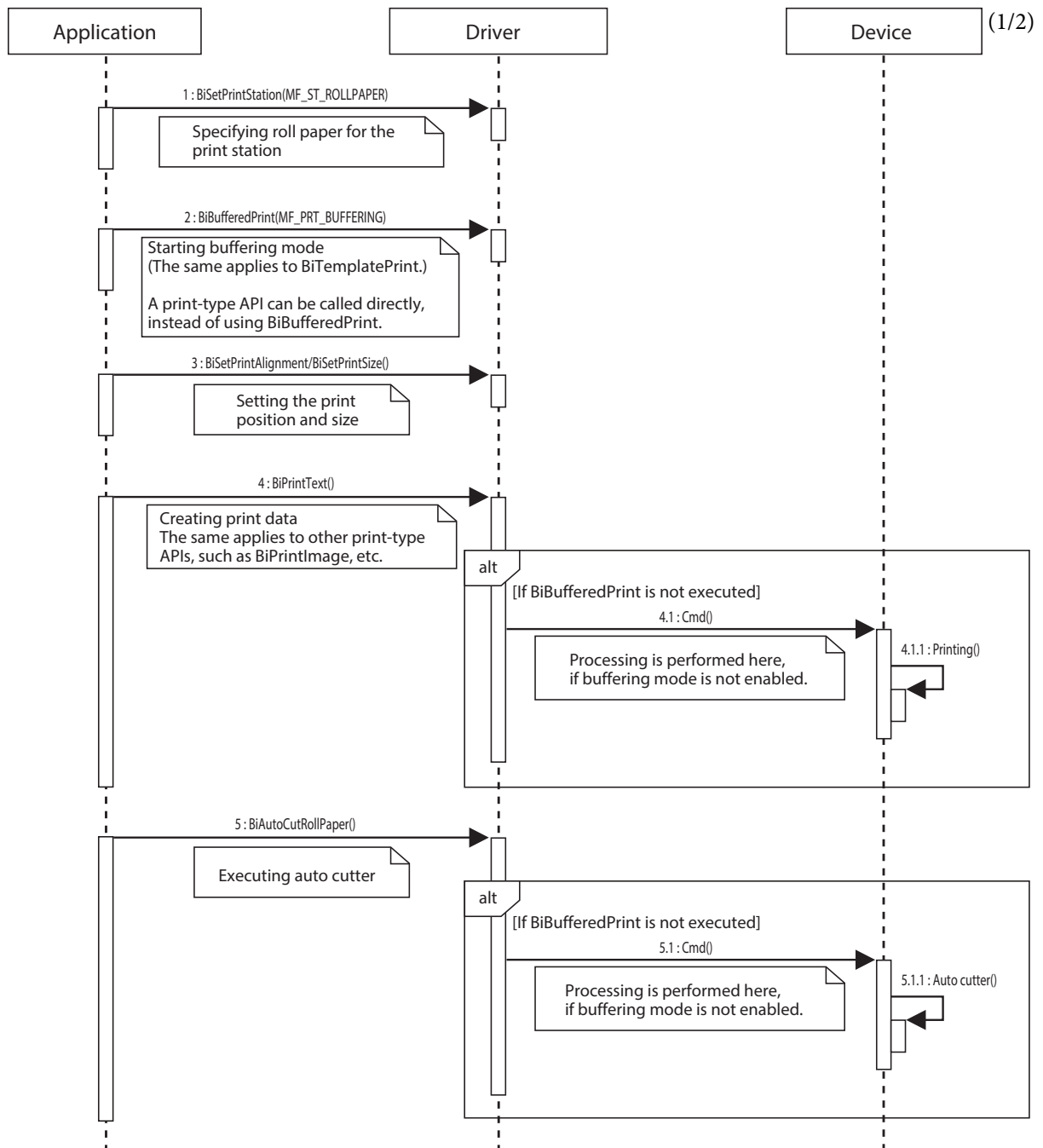


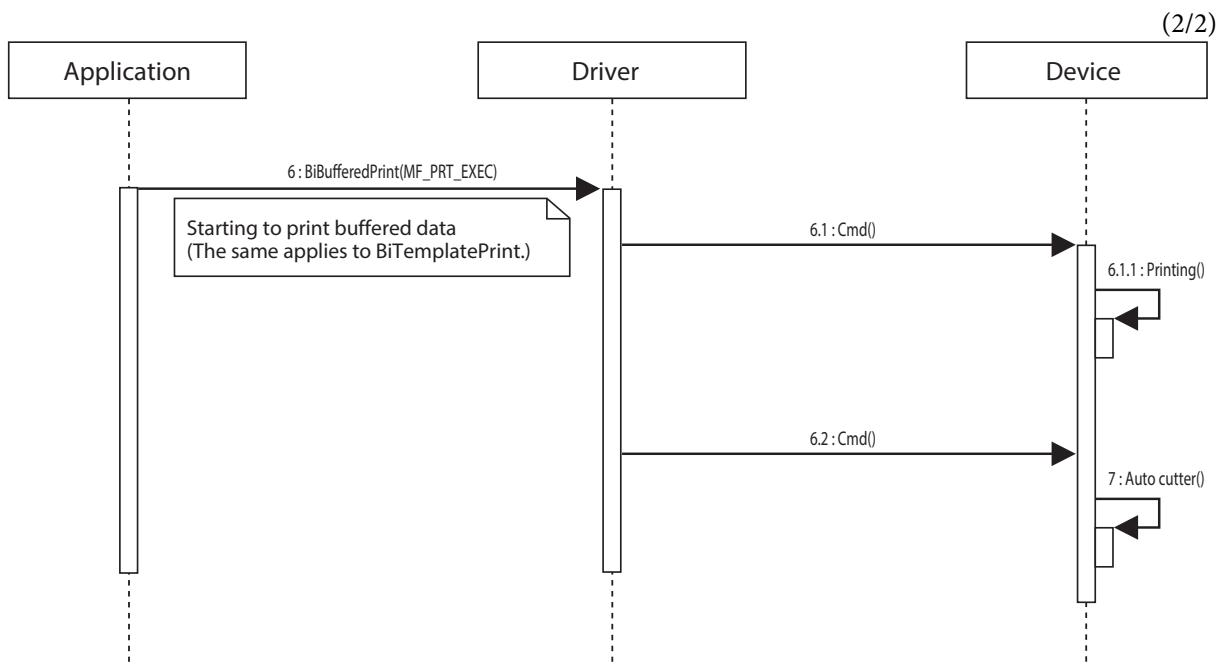




Roll paper printing (TM-S9000II and TM-S9000)

This is the process for printing roll paper.



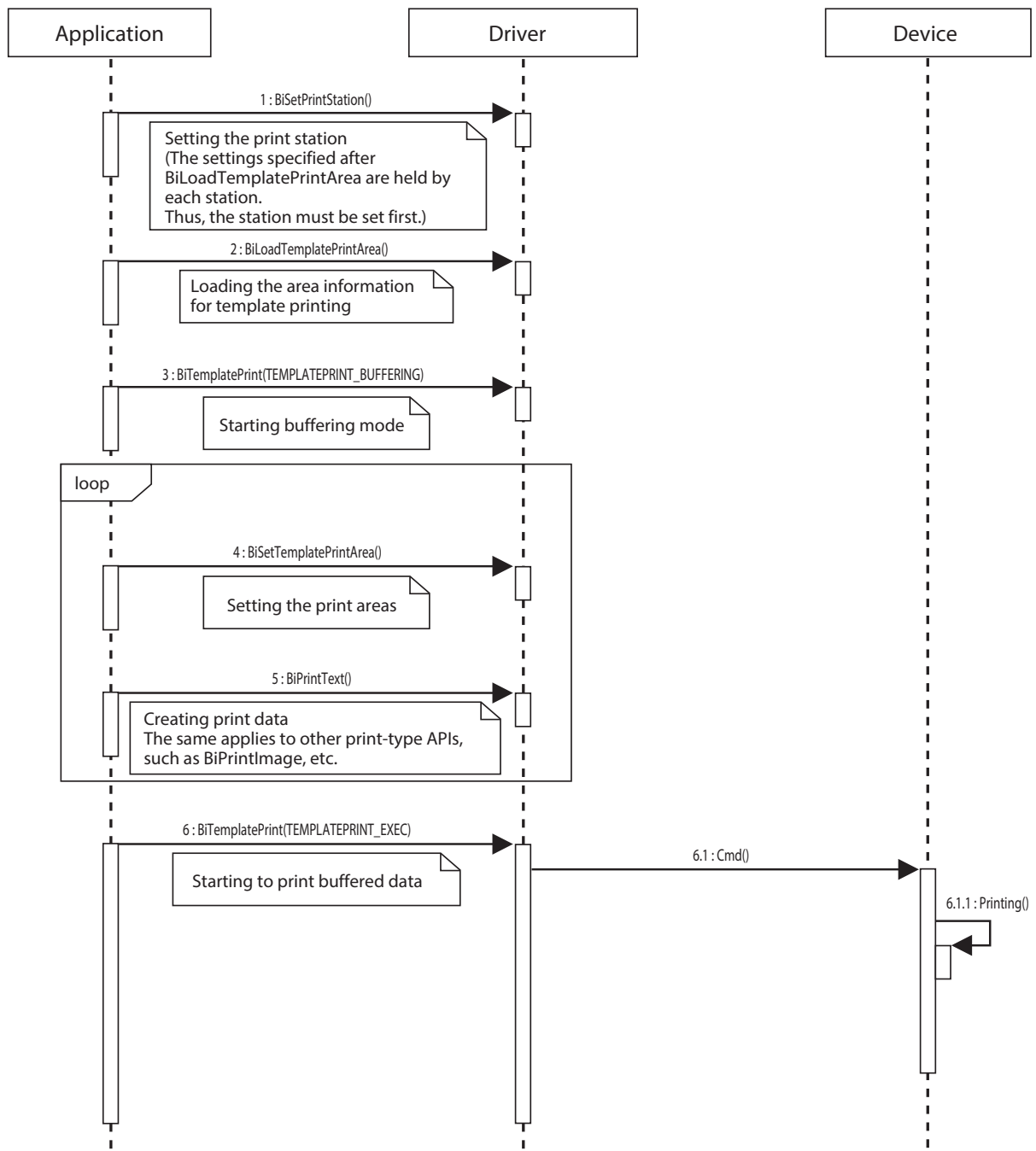


Template printing

This is the process for specifying an area for printing.

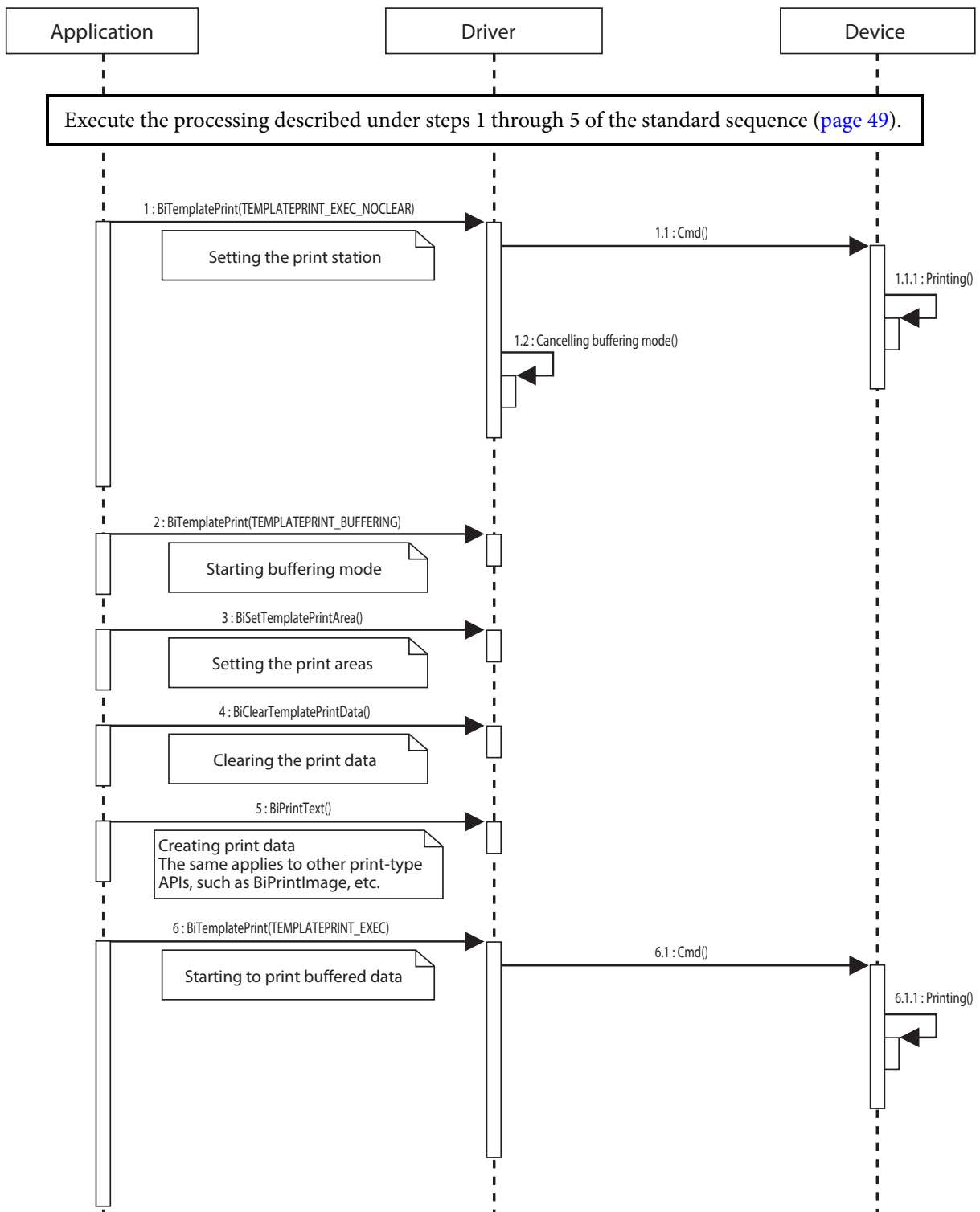
Standard sequence

This sequence is a basic sequence for template printing.



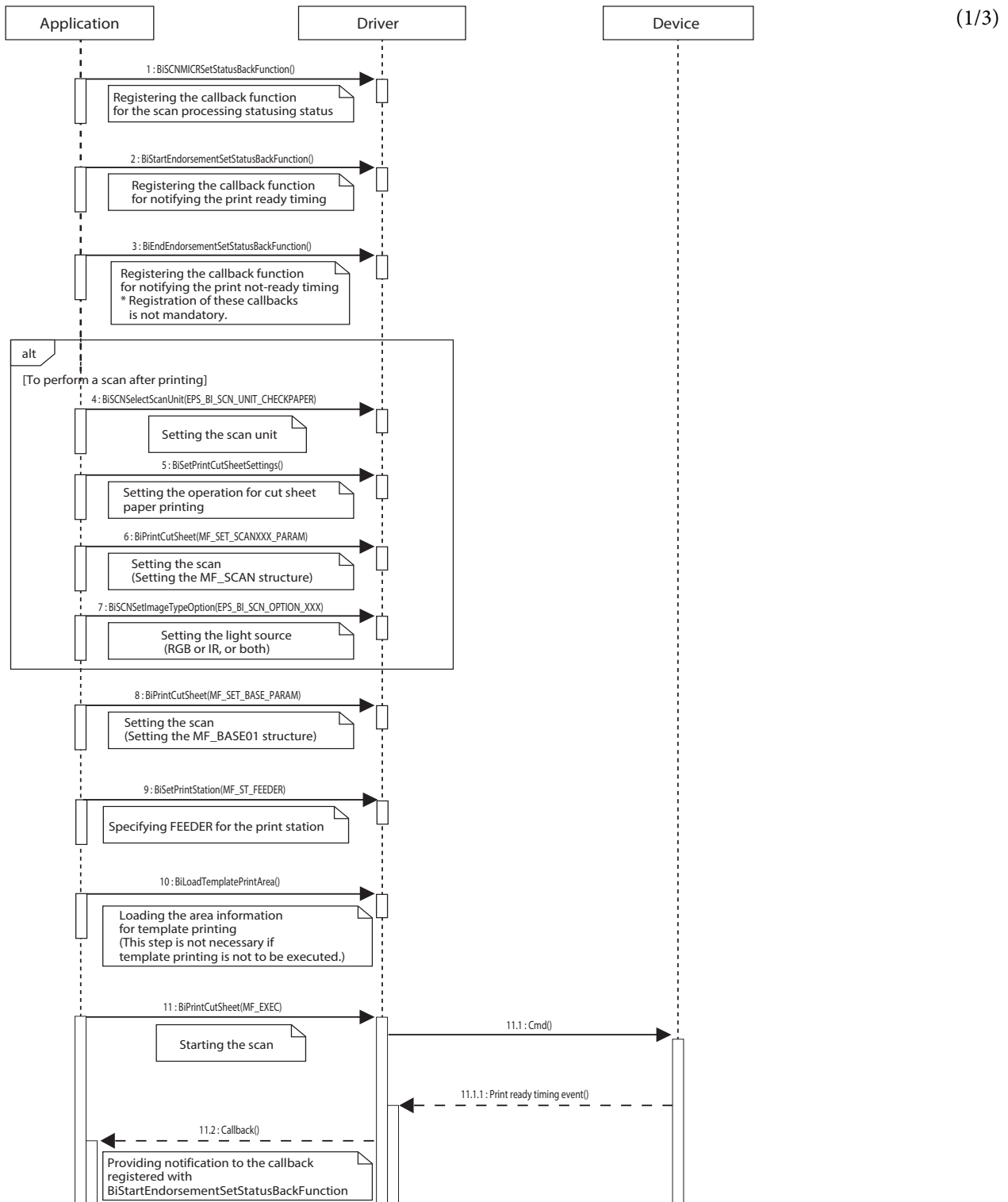
Continuous printing sequence

In this sequence, the same template is used for printing, with only the content changed for each sheet printed.

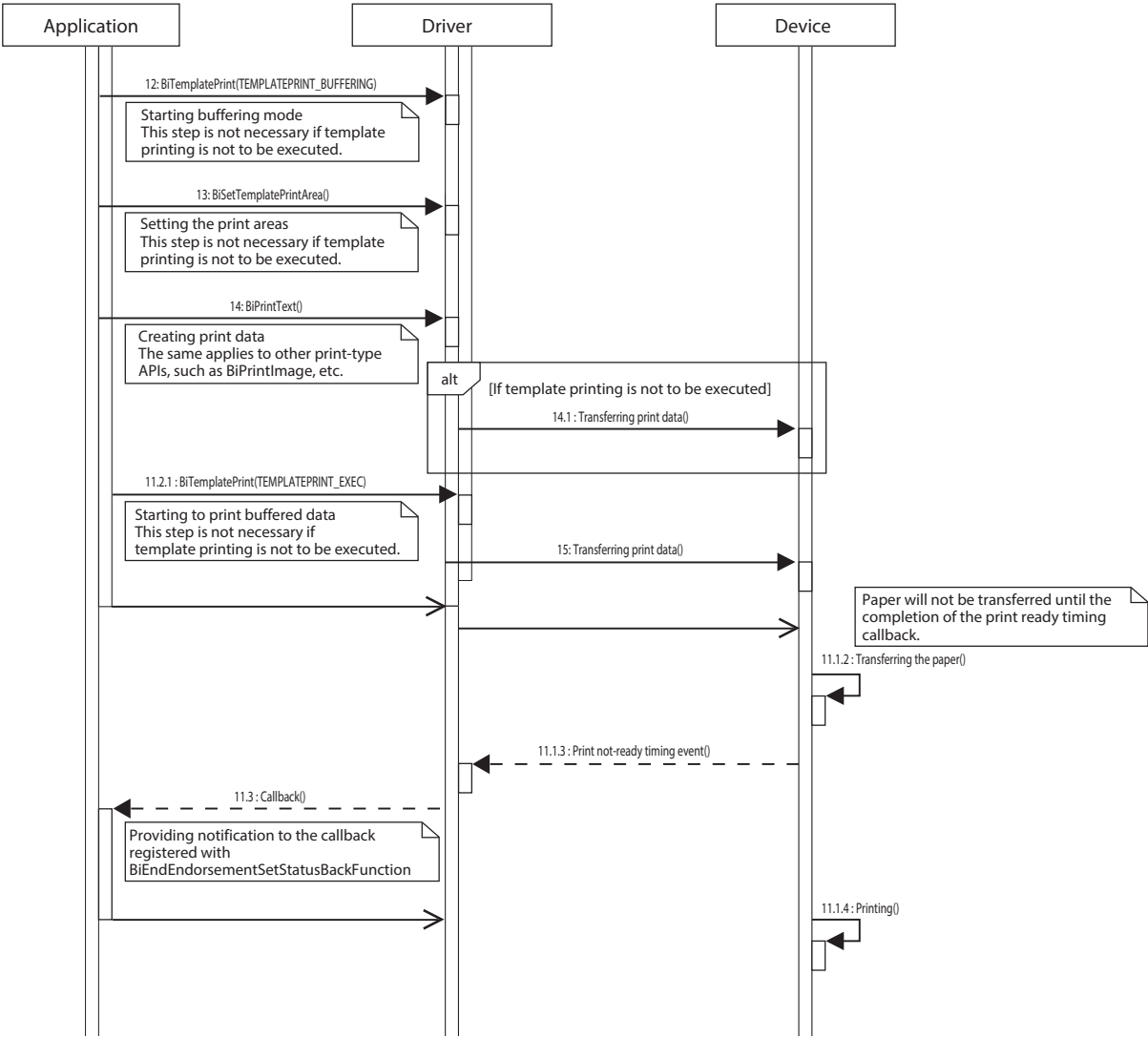


Cut sheet paper printing

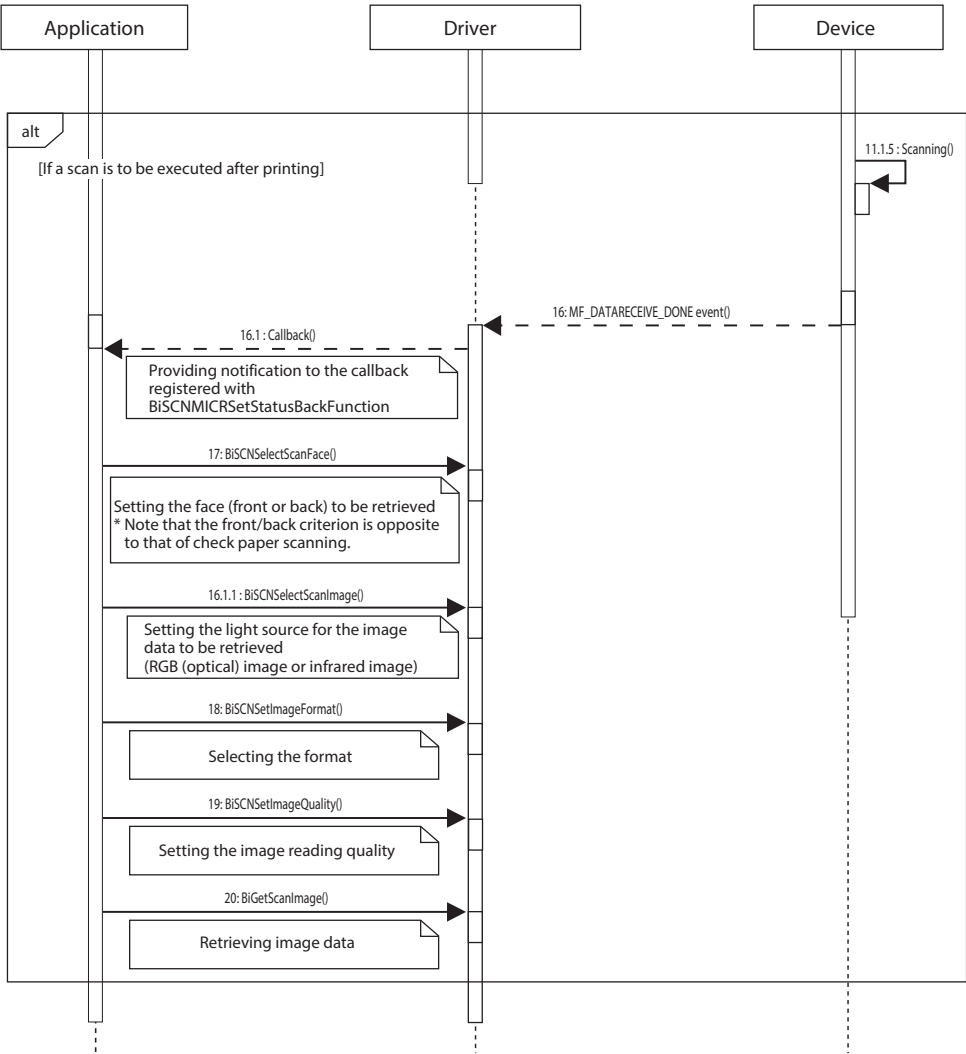
This is the process for printing and scanning a cut sheet, and then retrieving the scan results (image data).



(2/3)



(3/3)



CTS2010 compliant operation

This is the process for scanning the check sheets and retrieving the scan results listed below:

Front side:

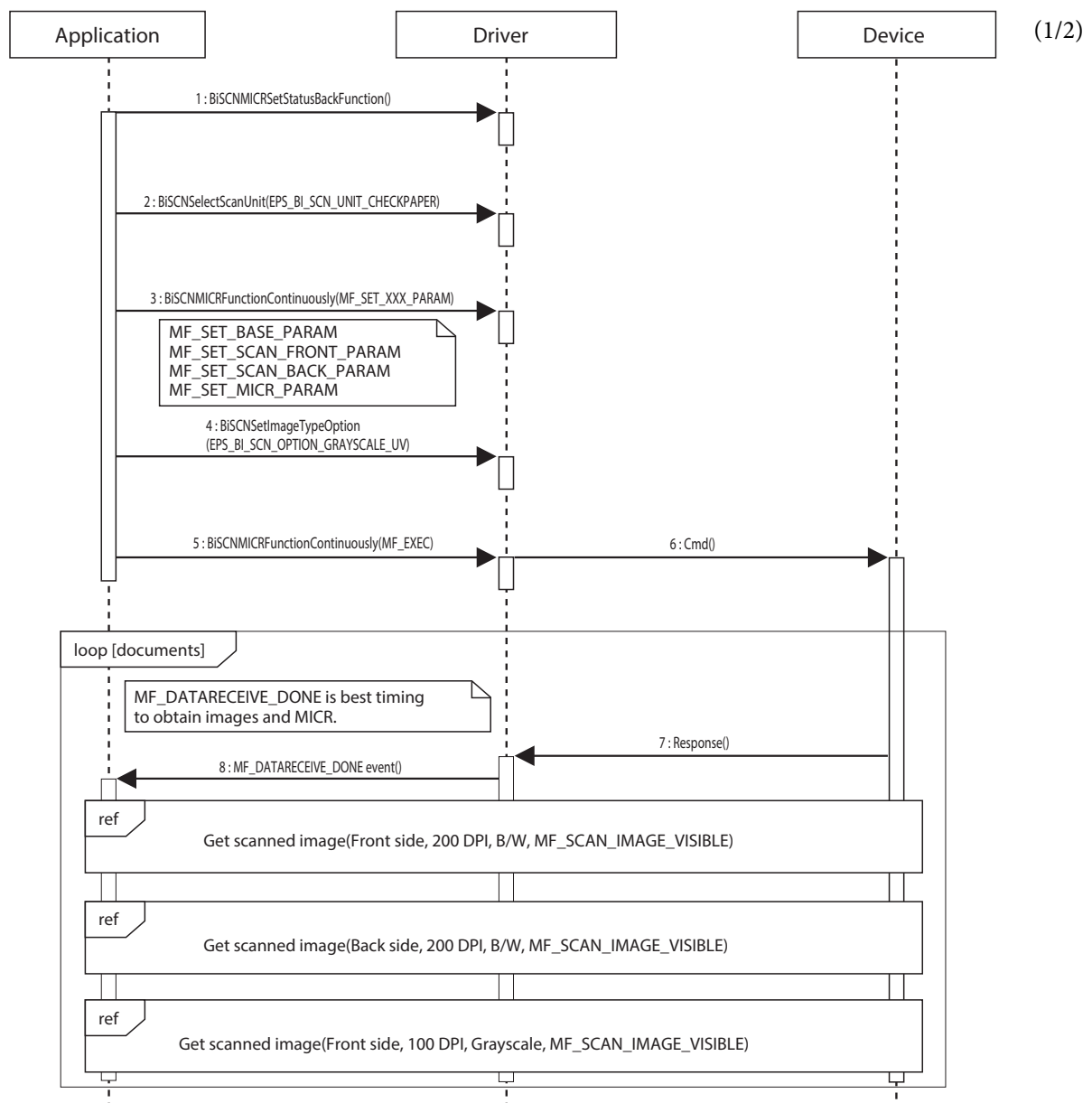
1. MICR
2. Visible B/W 200 dpi image, TIFF G4 format
3. Visible Grayscale 100 dpi image, JPEG(JFIF) format
(UV Grayscale 100 dpi image, JPEG(JFIF) format)

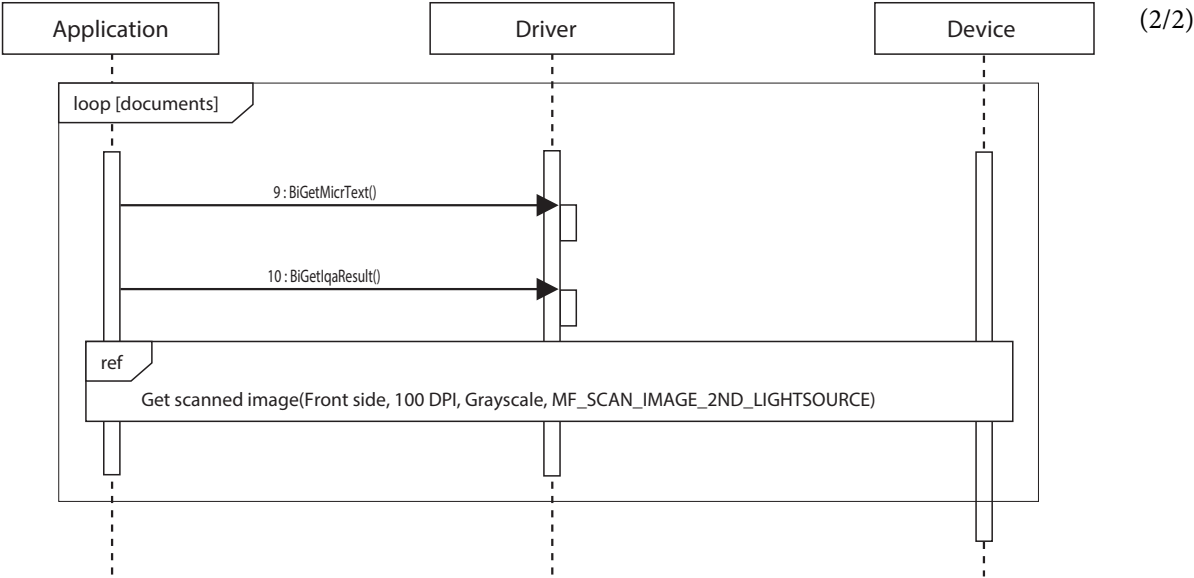
Back side:

4. Visible B/W 200 dpi image, TIFF G4 format

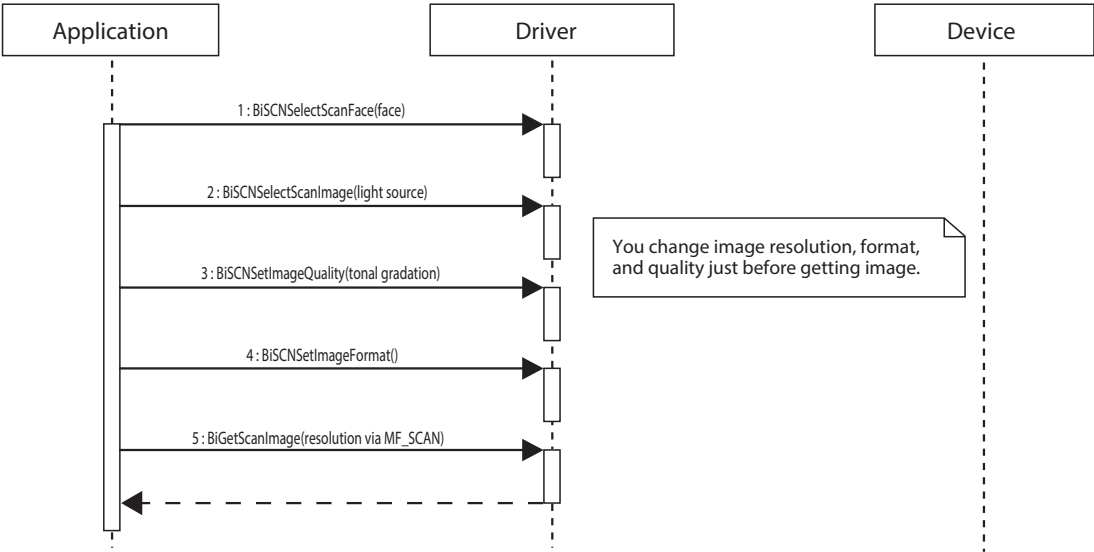


Regarding 3. Visible Grayscale 100 dpi image, UV image can be merged using "BiSCNSelectScanImage" on page 146.





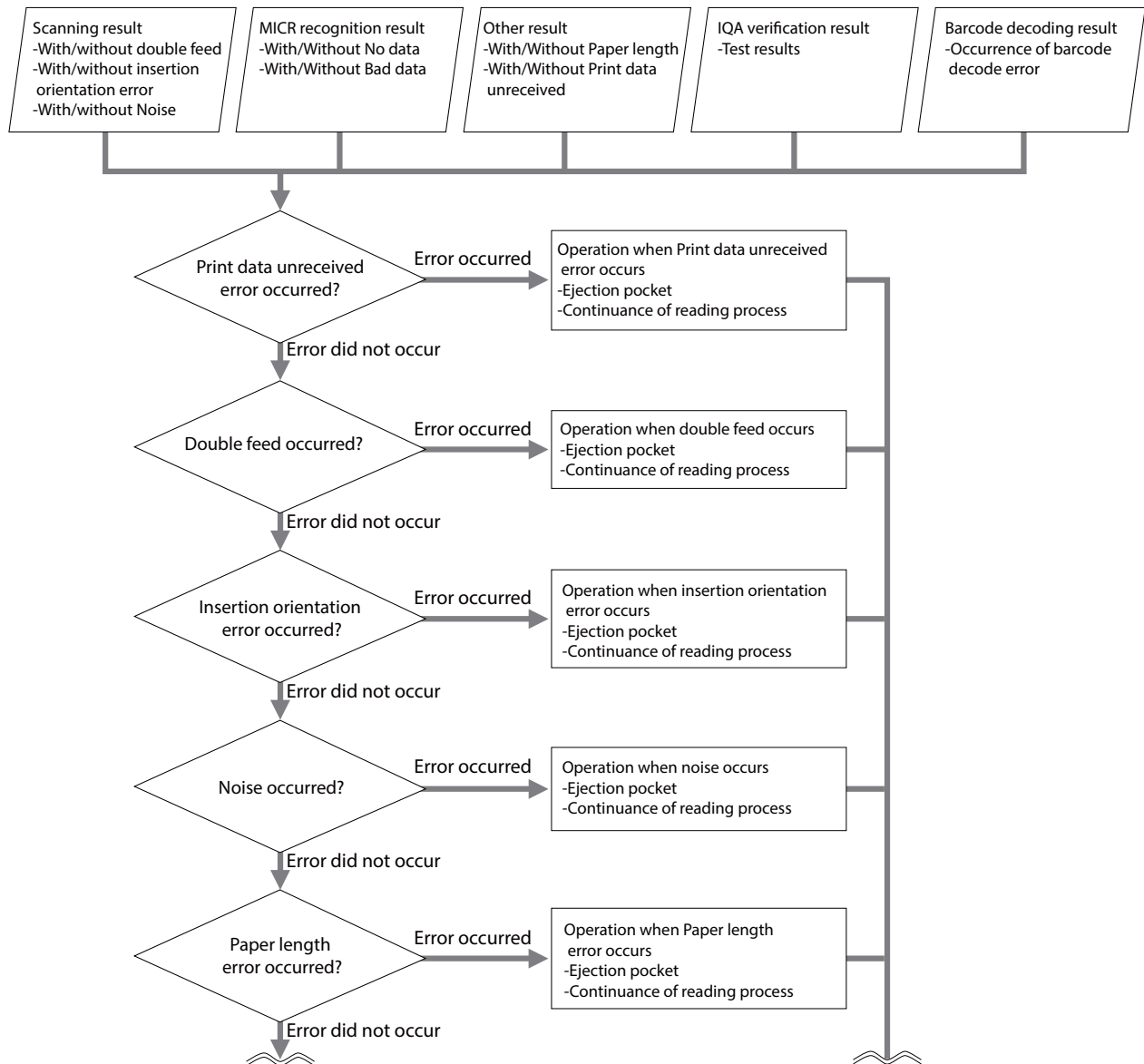
Get scanned image



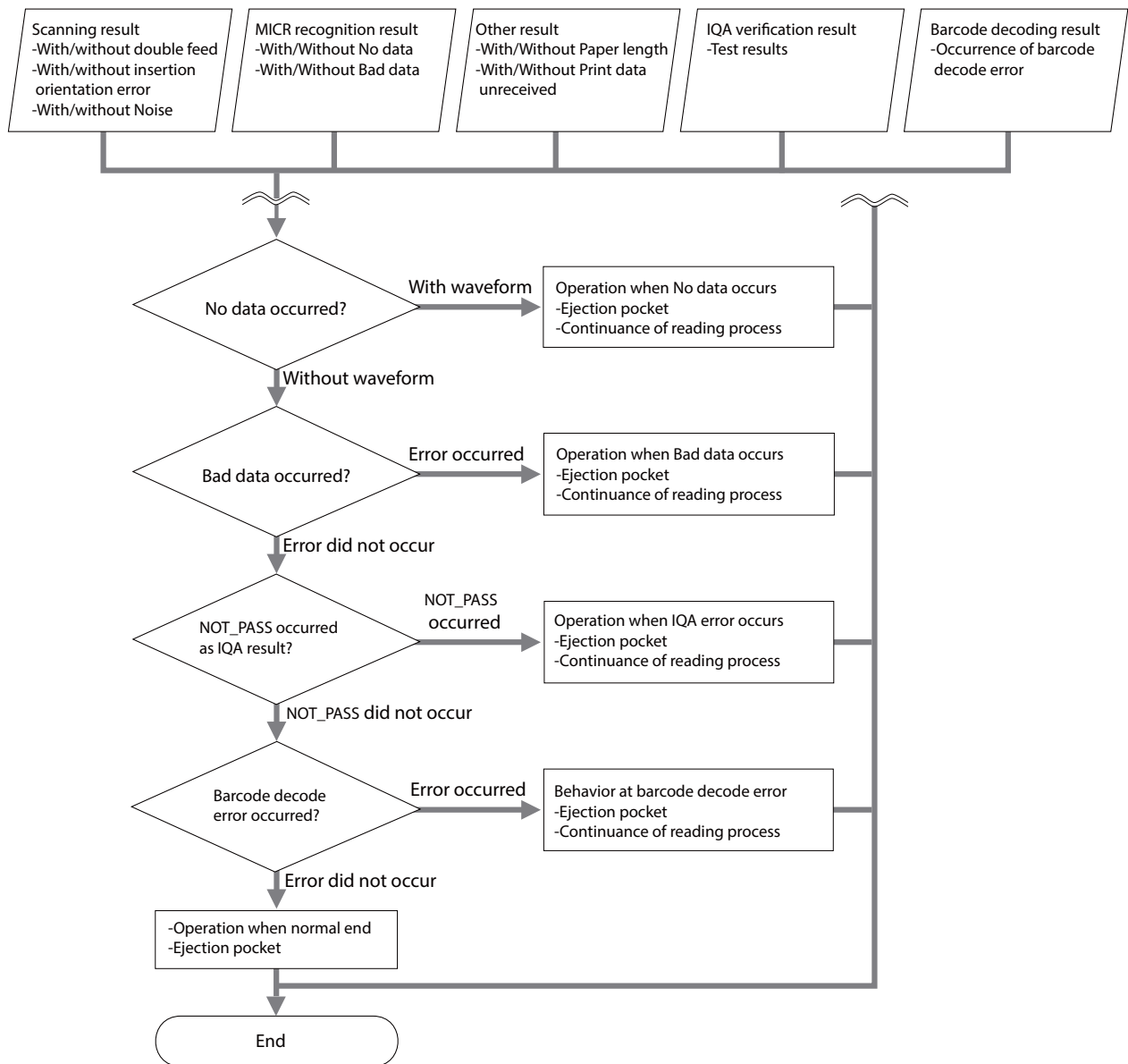
Priority for error detection

Error detection (Double feed/Insertion orientation error/Noise/No data/Bad data/Paper length/Print data unreceived/IQA/ Barcode decode error) and operation priorities when an error occurs are shown below.

(1/2)



(2/2)

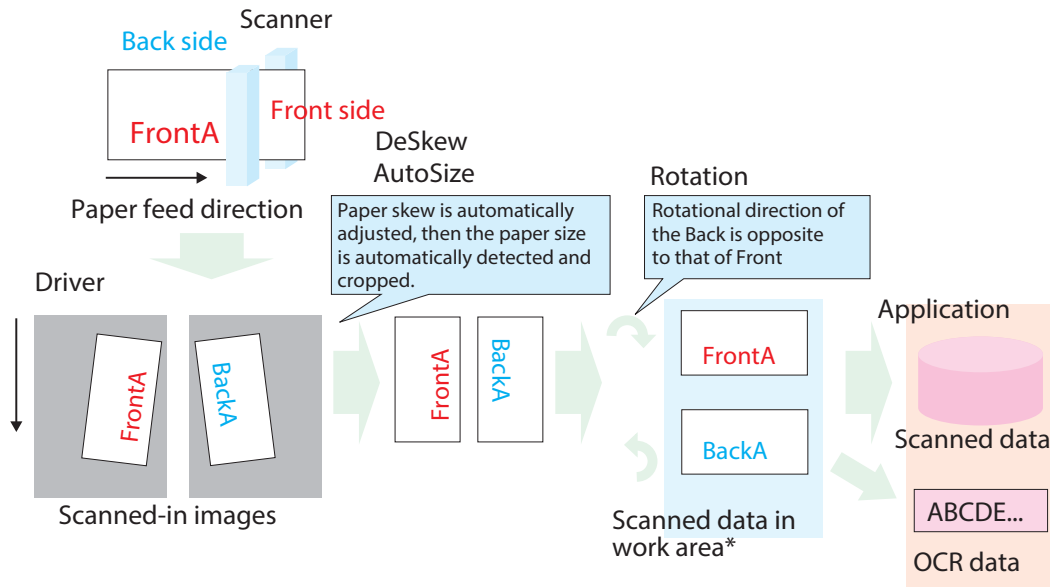


When executing heavy processing such as IQA, the print data unreceived error may occur frequently.

How to Use the Scanner Advanced Functions

When not using the scanner advanced functions

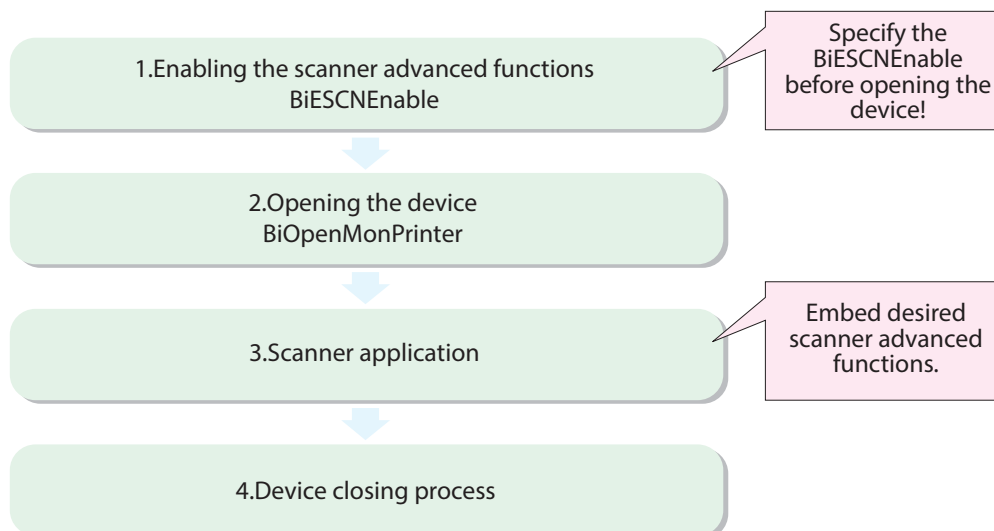
Scanned-in images are automatically edited and saved to the work area as shown below.



*OCR fonts in the scanned data saved in the work area can be read separately by specifying the area in which the fonts are included.
This can be carried out without using the scanner advanced functions.

Using the scanner advanced functions

Application process flow



Editing scanned-in images

Images scanned can be edited using customer's application and saved to the work area.
The following settings are available with the scanner advanced functions.

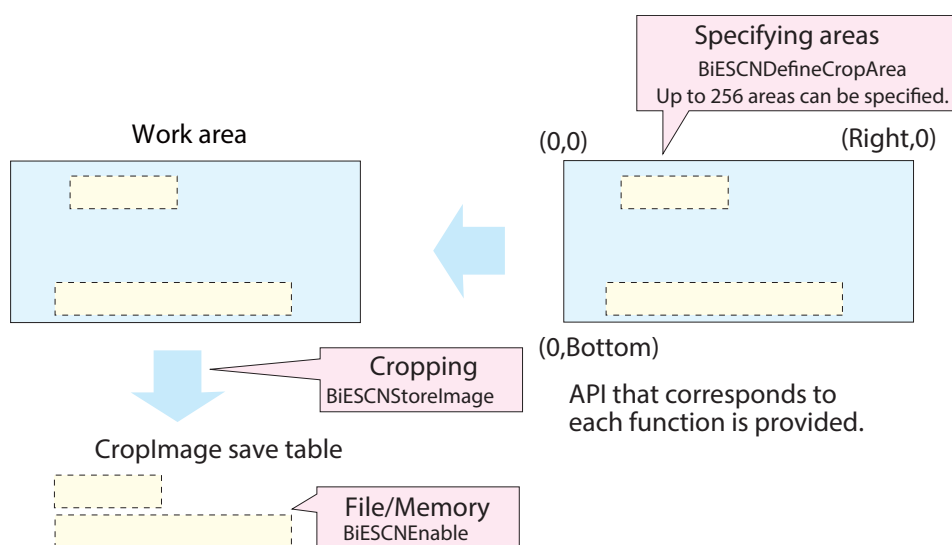
Item	API	Settings	When not using the advanced functions	Functions
Destination of the edited data	BiESCNEable	Memory/File	-	Specifies the destination of edited data. BiESCNEable have to be called before BiOpenMon-Printer if you demand to use the scanner advanced functions.
AutoSize	BiESCNSetAutoSize	Enable/Disabled	Enable	Crops scanned-in image into a check sheet size.
Skew adjustment	BiESCNSetDeSkew	Enable/Disable /Enable when skew by specified or more degrees is detected	Skew by 1.5 degrees or more is adjusted	Skew of the scanned check sheet is detected and corrected.
Rotation	BiESCNSetRotate	Enable/Disabled	Enable	Front side is rotated clockwise and back side is rotated counterclockwise.



There is no difference in scanning speed between using or not using the scanner advanced functions.

Cropping

Areas to be cropped from scanned data in work area can be specified in advance using coordinates.



Actions to be taken in the event of errors

There are two types of errors that may occur in the TM-S9000/S2000 API: the error notified by the device status and the error that occurs when the TM-S9000/S2000 API are called. A description of errors as well as solutions for fixing these errors is provided below. Please solve these errors in your own applications.

Device status

The errors below occur when the device status is retrieved.

* Not Supported in TM-S2000II and TM-S2000.

Macro Definition (Constant)	Cause
ASB_NO_RESPONSE	No response.
ASB_PRINT_SUCCESS	Printing completed.
ASB_DRAWER_KICK*	Status of drawer kick connector 3rd pin became high.
ASB_OFF_LINE	The device is in an off-line state.
ASB_MAIN_POCKET_NEAR_FULL	The main pocket is near full.
ASB_COVER_OPEN	The device cover is open.
ASB_PAPER_FEED*	Roll paper is being fed with the printer Feed Switch or Button 2.
ASB_SUB_POCKET_NEAR_FULL	The sub pocket is near full.
ASB_WAIT_PEPRT_EJECT	Waiting for removal of a slip paper.
ASB_PANEL_SWITCH*	Paper feed switch or button 2 is on.
ASB_MECHANICAL_ERR	A recoverable error occurred.
ASB_AUTOCUTTER_ERR*	An auto cutter error occurred.
ASB_UNRECOVER_ERR	An unrecoverable error occurred.
ASB_AUTORECOVER_ERR	An automatic recovery error occurred.
ASB_NOT_CARD_INSERT	No card is inserted in the PhotoID detector.
ASB_RECEIPT_NEAR_END*	No paper in the roll paper near end detector
ASB_MAIN_NEAR_FULL	The main pocket is not near full.
ASB_EJECT_SENSOR_NO_PAPER	No paper in the ejection sensor.
ASB_RECEIPT_END*	No paper in the roll paper end detector
ASB_SUB_NEAR_FULL	The sub pocket is not near full.
ASB_PAPER_INTERMEDIATE	No paper in the intermediate detector
ASB_SLIP_TOF	No paper in the TOF detector
ASB_ASF_PAPER	No paper in the ASF detector
ASB_SLIP_SELECTED	The cut sheet paper is not selected.
ASB_PRINT_SLIP	The cut sheet paper cannot be printed.
ASB_STAMP_EXIST	No stamp.
ASB_VALIDATION_SELECTED	The validation is not selected.
ASB_PRINT_VALIDATION	The validation cannot be printed.
ASB_WAIT_INSERT	Waiting for insertion of cut sheet paper.
ASB_SLIP_PAPER_SIZE	No paper in the paper length detector.
ASB_VALIDATION_NO_PAPER	No paper in the validation detector.
ASB_FRANKING_SENSOR	No paper in the franking detector.

Return value

These errors (return values of the TM-S9000/S2000 API) occur when the TM-S9000/S2000 API are called. The content of the errors differ depending on the TM-S9000/S2000 API.

Macro Definition (Constant)	Value	Cause	Solution
SUCCESS	0	Processing was completed successfully.	-
ERR_TYPE	-10	The nType parameter is incorrect.	Specify a correct constant that is defined in the header file.
ERR_OPENED	-20	The port is being used by other applications.	Terminate other applications.
ERR_NO_PRINTER	-30	The specified device cannot be found.	Check the connected status of the device. (Power status of the device, cable connection, etc.)
ERR_NO_TARGET	-40	An unsupported USB driver is used.	Reinstall the TM-S9000/S2000 driver.
ERR_NO_MEMORY	-50	Insufficient memory.	Terminate other applications or upgrade the memory.
ERR_HANDLE	-60	The handle value specifying the device is invalid.	Check if the handle value has been retrieved with BiOpenMonPrinter.
ERR_TIMEOUT	-70	A timeout error occurred.	If a timeout error occurs repeatedly, check the device status.
ERR_ACCESS	-80	The API did not complete successfully.	Check the device status. (Power status of the device, cable connection, etc.)
ERR_PARAM	-90	A parameter error occurred.	Check if the API parameter value or the value of the member of the structure is correct.
ERR_NOT_SUPPORT	-100	The API feature called is not available.	<ul style="list-style-type: none"> Check the API or the parameter called. If you are using the advanced scanner feature API, check if BiESCNEnable has been called.
ERR_OFFLINE	-110	The device is in an off-line state.	Check the device status.
ERR_NOT_EPSON	-120	A device other than an Epson device is being used.	Check if an Epson device is being used.
ERR_WITHOUT_CB	-130	The CALLBACK function is not registered.	Call either BiSCNMICRSetStatusBackFunction or BiSCNMICRSetStatusBackWnd.
ERR_BUFFER_OVER_FLOW	-140	There is not enough memory available.	Increase the available memory size.
ERR_ENABLE	-160	BiOpenMonPrinter has already been called.	Execute BiCloseMonPrinter and then call BiOpenMonPrinter again.
ERR_DISK_FULL	-170	There is not enough free disk space.	Review the operating environment.
ERR_NO_IMAGE	-180	The image file cannot be found.	Check if the image file exists at the specified path.
ERR_ENTRY_OVER	-190	The maximum number of entries that can be registered has already been reached.	-

Macro Definition (Constant)	Value	Cause	Solution
ERR_CROPEAREID	-200	The specified CropArea does not exist.	Set DefineCropArea and then call DefineCropArea again.
ERR_EXIST	-210	The specified data already exists.	Change the parameter or call BiESCNClearImage.
ERR_NOT_FOUND	-220	The specified data or module cannot be found.	<ul style="list-style-type: none"> If the data cannot be found, check the parameter of the executed API. If the module cannot be found, reinstall the TM-S9000/S2000 driver.
ERR_IMAGE_FILEOPEN	-230	An attempt to open a file failed.	Check if the file is not being used by other applications.
ERR_IMAGE_UNKNOWNFORMAT	-240	The file format is either invalid or not supported.	Check if the file can be opened successfully with other applications.
ERR_IMAGE_FAILED	-250	An attempt to save an image file failed.	Review the operating environment.
ERR_IMAGE_FILEREAD	-290	An attempt to load an image file failed.	Check if the file is located at the save destination.
ERR_PAPERINSERT_TIMEOUT	-300	An attempt to insert paper failed.	Check if the paper is set correctly in the device.
ERR_EXEC_FUNCTION	-310	The processing invoked by other API is being executed.	Wait until the processing completes and then call the API again.
ERR_RESET	-400	The device is unavailable because it is currently being reset.	Wait for a while and then call the API again.
ERR_ABORT	-430	The scan processing was cancelled.	Determine whether or not to execute scan processing again.
ERR_MICR	-440	An attempt to read MICR data failed.	Execute scan processing again.
ERR_SCAN	-450	An attempt to read image data failed.	Execute scan processing again.
ERR_NOT_EXEC	-470	The processing specified with the API was not executed.	<ul style="list-style-type: none"> Call the API again. Check if APIs were called in correct order.
ERR_DATA_INVALID	-480	The area information either did not exist or contained invalid data.	Check the area information file.
ERR_SIZE	-1000	The size is not set, or the value is invalid.	Check the value specified for the size and call the API again.
ERR_PAPER_PILED	-1010	A double feed error occurred.	Execute scan processing again.
ERR_PAPER_JAM	-1020	A paper jam error occurred.	Remove the paper.
ERR_COVER_OPEN	-1030	Cover is open.	Close the cover.
ERR_MICR_NODATA	-1040	An attempt to detect magnetic waveform data failed.	Execute scan processing again.
ERR_MICR_BADDATA	-1050	An unanalyzable character was detected.	Execute scan processing again.
ERR_MICR_PARSE	-1060	An attempt to parse MICR data failed.	Execute scan processing again.
ERR_MICR_NOISE	-1070	A noise error was detected.	Execute scan processing again.
ERR_PAPER_EXIST	-1090	Paper exists in the path.	Remove the paper.
ERR_PAPER_INSERT	-1100	An attempt to insert paper failed.	Set the paper correctly in the device.
ERR_LESS_CHECKS	-1110	An attempt to scan a specified number of sheets failed.	Either change the specified number of sheets or increase the number of sheets to be loaded.

Macro Definition (Constant)	Value	Cause	Solution
ERR_SCN_IQA	-1120	NOT_PASS is detected at the IQA validation.	Load BiGetIQAResult, check the test results, and determine whether to scan again.
ERR_BARCODE_NODATA	-1130	An attempt to detect a barcode failed.	-
ERR_IMAGE_FILEREMOVE	-1170	An attempt to delete an image file failed.	Check if the file is not being used by other applications.
ERR_PRINT_DATA_LENGTH_EXCEED	-1180	The size of the printing content exceeds printable area	The size of the printing content exceeds printable area. Use an appropriate paper size.
ERR_PRINT_DATA_UNRECEIVE	-1190	Print data unreceived error occurred.	Review the operating environment, or set bEndorsePrintMode for the MF_PROCCES structure to MF_ENDORSEPRINT_MODE_DATAWAITING and scan again.
ERR_INK_STATUS *	-1200	No ink cartridge or ink is low.	Install a new ink cartridge.
ERR_UNKNOWN	-9999	A module cannot be loaded.	Reinstall the driver.

* Supported in Ver.2.10 or later

Sample Program

Programming with the TM-S9000/S2000 API is described using 8 functional level sample programs.

Sample Program	Description
Step1	This is a sample program for opening and closing devices, and scanning.
Step2	In addition to Step 1, this sample program allows you to select to scan either a check sheet or a card, and then retrieve and display the scanned images.
Step3	In addition to Step 2, this sample program retrieves and displays MICR data. It also cleans the MICR.
Step4	In addition to Step 3, this sample program performs physical endorsement/electronic endorsement.
Step5	In addition to Step 3, this sample program processes cashier's checks. It also prints cut sheets/roll paper (TM-S9000II and TM-S9000).
Step6	In addition to Step 3, this sample program retrieves and displays OCR-AB recognition results. It also sounds an alarm.
Step7	In addition to Step 3, this sample program processes IQA and displays the results. It also allows the selection of scanning mode (High Speed Mode/Confirmation Mode/Waterfall).
Step8	In addition to Step 3, this sample program retrieves and displays barcode decode results, and detects errors. It also retrieves device status and device information.
UV	This sample program checks whether the product is equipped with a UV light source and displays images scanned with the visible light source and UV light source. IQA conforms to India CTS2010. (TM-S2000II UV model and TM-S2000 UV model)



Sample programs are designed so that the screens and components are separate, and therefore they can easily be copied and added to customers' programs.

TM-S9000/S2000 API Reference

This chapter provides a description of the TM-S9000/S2000 API and their syntax.

API Index

Basic APIs

API	Page	API	Page
BiOpenMonPrinter	page 68	BiCloseMonPrinter	page 70
BiSCNMICRSetStatusBackFunction	page 71	BiSCNMICRCancelStatusBack	page 77
BiStartEndorsementSetStatusBackFunction	page 78	BiEndEndorsementSetStatusBackFunction	page 81
BiStartEndorsementCancelStatusBack	page 80	BiEndEndorsementCancelStatusBack	page 84
BiSCNMICRFunctionContinuously	page 106	BiSCNMICRFunctionPostPrint	page 115
BiPrintImage	page 192	BiPrintText	page 200
BiSCNPrintText	page 202	BiTemplatePrint	page 218
BiPrintCutSheet	page 211	BiClearTemplatePrintData	page 223
BiSCNSelectScanUnit	page 100	BiSCNSelectScanFace	page 134
BiSCNSetImageTypeOption	page 101	BiSetPrintStation	page 176
BiSetPrintCutSheetSettings	page 209	BiLoadTemplatePrintArea	page 217
BiSetTemplatePrintArea	page 220	BiSCNSetImageFormat	page 137
BiSCNSetImageQuality	page 144	BiSCNSelectScanImage	page 146
BiGetScanImage	page 161	BiGetMicrText	page 224
BiSetStatusBackFunctionEx	page 88	BiCancelStatusBack	page 91
BiSetInkStatusBackFunctionEx	page 94	BiCancelInkStatusBack	page 97
BiCancelError	page 271	BiResetPrinter	page 270
BiSCNGetImageTypeOption	page 105	BiGetPrintStation	page 173
BiGetPrintCutSheetSettings	page 209	BiSCNGetImageFormat	page 136
BiSCNGetImageQuality	page 142	BiESCNEEnable	page 233
BiBufferedPrint	page 180	BiSetPrintPosition	page 176
BiSetPrintSize	page 184		

API List (Alphabetical Order)

API	Page	API	Page
A			
BiAutoCutRollPaper	page 216		
B			
BiBufferedPrint	page 180		
C			
BiCancelError	page 271	BiCancelInkStatusBack	page 91
BiCancelStatusBack	page 91	BiClearTemplatePrintData	page 223
BiCloseMonPrinter	page 70	BiConfirmBufferedData	page 260
D			
BiDecodeBarcode	page 169	BiDecodeBarcodeMemory	page 170
E			
BiEndEndorsementCancelStatusBack	page 82	BiEndEndorsementSetStatusBackFunction	page 81
BiEndEndorsementSetStatusBackWnd	page 82	BiESCNClearImage	page 249
BiESCNDeriveCropArea	page 244	BiESCNEnable	page 233
BiESCNGetAutoSize	page 234	BiESCNGetCutSize	page 237
BiESCNGetDeSkew	page 240	BiESCNGetDocumentSize	page 242
BiESCNGetMaxCropAreas	page 246	BiESCNGetRemainingImages	page 253
BiESCNGetRotate	page 238	BiESCNRetrieveImage	page 251
BiESCNSetAutoSize	page 235	BiESCNSetCutSize	page 237
BiESCNSetDeSkew	page 241	BiESCNSetDocumentSize	page 243
BiESCNSetRotate	page 239	BiESCNSoreImage	page 247
G			
BiGetBarcodeData	page 167	BiGetCounter	page 265
BiGetInkStatus	page 92	BiGetIQAResult	page 230
BiGetMicrText	page 157	BiGetOcrABText	page 159
BiGetOfflineCode	page 273	BiGetOfflineCodeByIndex	page 274
BiGetPrintAlignment	page 194	BiGetPrintControl	page 182
BiGetPrintCutSheetSettings	page 209	BiGetPrintImageMethod	page 186
BiGetPrintPosition	page 173	BiGetPrintSize	page 181
BiGetPrintStation	page 173	BiGetPrnCapability	page 135
BiGetRealStatus	page 86	BiGetScanImage	page 161
BiGetScanImageSize	page 166	BiGetStatus	page 85
BiGetTransactionNumber	page 224	BiGetType	page 272
BiGetVersion	page 231		
I			
BiInkHeadCleaning	page 262	BiInsertValidation	page 207
L			
BiLoadAPISettings	page 267	BiLoadTemplatePrintArea	page 217
M			
BiMICRCleaning	page 261	BiMICRClearSpaces	page 154
BiMICRGetStatus	page 259	BiMICRSelectDataHandling	page 98

API	Page	API	Page
O			
BiOpenDrawer	page 263	BiOpenMonPrinter	page 68
P			
BiPrintBarCode	page 188	BiPrintCutSheet	page 211
BiPrintImage	page 192	BiPrintMemoryImage	page 196
BiPrintMultipleToneImage	page 204	BiPrintText	page 200
R			
BiRemoveValidation	page 208	BiResetCounter	page 266
BiResetPrinter	page 270	BiRingBuzzer	page 264
S			
BiSCNDeleteCroppingArea	page 133	BiSCNGetClumpStatus	page 275
BiSCNGetCroppingArea	page 147	BiSCNGetImageAdjustment	page 150
BiSCNGetImageFormat	page 136	BiSCNGetImageQuality	page 142
BiSCNGetImageTypeOption	page 105	BiSCNGetScanArea	page 139
BiSCNMICRCancelFunction	page 131	BiSCNMICRCancelStatusBack	page 77
BiSCNMICRFunction	page 124	BiSCNMICRFunctionContinuously	page 106
BiSCNMICRFunctionPostPrint	page 115	BiSCNMICRSetStatusBackFunction	page 71
BiSCNMICRSetStatusBackWnd	page 73	BiSCNMICRSetStatusBackWndEx	page 75
BiSCNPrintMemoryImage	page 198	BiSCNPrintText	page 202
BiSCNSelectScanFace	page 134	BiSCNSelectScanImage	page 146
BiSCNSelectScanUnit	page 100	BiSCNSetCroppingArea	page 148
BiSCNSetImageAdjustment	page 152	BiSCNSetImageFormat	page 137
BiSCNSetImageQuality	page 144	BiSCNSetImageTypeOption	page 101
BiSCNSetScanArea	page 140	BiSelectErrorEjectAtContinuously	page 256
BiSelectJamDetect	page 258	BiSetBehaviorToScnResult	page 254
BiSetConfigure	page 268	BiSetEndorseDirection	page 178
BiSetInkStatusBackFunction	page 93	BiSetInkStatusBackFunctionEx	page 94
BiSetInkStatusBackWnd	page 95	BiSetInkStatusBackWndEx	page 96
BiSetMonInterval	page 172	BiSetNumberOfDocuments	page 255
BiSetOcrABAAreaOrigin	page 155	BiSetPaperThickness	page 276
BiSetPrintAlignment	page 195	BiSetPrintControl	page 183
BiSetPrintCutSheetSettings	page 210	BiSetPrintImageMethod	page 187
BiSetPrintPosition	page 176	BiSetPrintSize	page 184
BiSetPrintStation	page 174	BiSetStatusBackFunction	page 87
BiSetStatusBackFunctionEx	page 88	BiSetStatusBackWnd	page 89
BiSetStatusBackWndEx	page 90	BiSetTemplatePrintArea	page 220
BiSetTransactionNumber	page 225	BiSetTransactionNumberWithIncremental	page 227
BiSetWaterfallMode	page 228	BiStartEndorsementCancelStatusBack	page 80
BiStartEndorsementSetStatusBackFunction	page 78	BiStartEndorsementSetStatusBackWnd	page 79
T			
BiTemplatePrint	page 218		
U			
BiUpdateEndorseText	page 205		

BiOpenMonPrinter

Communication initialization API.

This API enables communication with a device connected to local computers.



Handle acquired by this API are used as nHandle for other APIs.

Syntax

```
int BiOpenMonPrinter(int nType, LPSTR pName)
```

Argument

nType: This specifies the type of name specified in pName. One of the following two types is specified.

Constant	Value	Description
TYPE_PORT	1	Specify the port name in pName.
TYPE_PRINTER	2	Specify the device name in pName.

pName: This specifies the device that is opened. The specification is as follows, depending on the nType value.

- If the nType is TYPE_PORT, it specifies the port name to which the device is connected.

<TM-S9000II/TM-S9000>

```
iRet = BiOpenMonPrinter(TYPE_PORT, "USB3");
```

<TM-S2000II/TM-S2000>

```
iRet = BiOpenMonPrinter(TYPE_PORT, "USB4");
```

- If the nType is TYPE_PRINTER, the device name is specified.

<TM-S9000II/TM-S9000>

```
iRet = BiOpenMonPrinter(TYPE_PRINTER, "TM-S9000U");
```

<TM-S2000II/TM-S2000>

```
iRet = BiOpenMonPrinter(TYPE_PRINTER, "TM-S2000U");
```



If the nType is TYPE_PRINTER, you can automatically detect and control devices connected to the computer. In that case, set pName as "TM-S9000U, TM-S2000U".

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_TARGET	-40	An unsupported device was specified (The device's power is not On or the cable connections are faulty, etc.)
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_UNKNOWN	-9999	Invalid install configuration




- For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).
- If the common memory has failed to be secured because of the user authority, ERR_NO_PRINTER or ERR_NO_MEMORY is returned.

Description

Before using an API function other than this function, it is necessary that this function be executed first. The handle value obtained is valid only in the same thread.

BiCloseMonPrinter

Communication termination API.
The communication enabled status specified using the communication initialization API is disabled.

Handle acquired by this API are used as nHandle for other APIs.

Syntax


```
int BiCloseMonPrinter(int nHandle)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

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

For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSCNMICRSetStatusBackFunction

Registers callback function for notification of the reading status.



This is unavailable when the development environment is VB 6.0.

Syntax

```
int BiSCNMICRSetStatusBackFunction
(int nHandle, int (CALLBACK EXPORT* pScnMicrCB)
(DWORD dwTransactionNumber, WORD wMainStatus
, WORD wSubStatus, LPSTR lpcPortName))
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
 pScnMicrCB: Specifies the address of the callback function for notification of the BiSCNMICRFunctionContinuously scanning processing status.

Callback function parameters

dwTransactionNumber: The transaction number (ID) corresponding to the check sheet of the processing status source.
 wMainStatus: The main status of the processing status.
 wSubStatus: The sub status of the processing status.
 lpcPortName: The memory address where the port name of the callback invoker is saved.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Registers the address of the callback function for notification of the BiSCNMICRFunctionContinuously / BiSCNMICRFunctionPostPrint scanning processing status. When the processing status of the check sheet changes, the values are saved in dwTransactionNumber, wMainStatus, and wSubStatus, and the registered callback function is invoked. In this case, the port name is saved in lpcPortName in order to distinguish the callback invoker.



For details of the processing status, refer to processing status in ["Processing status list" on page 114](#).

BiSCNMICRSetStatusBackWnd

In order to provide a notification of the scan processing status, this records the handle of the button that sends a click event and the memory address that stores the notified value.



This API is compatible with the TM-J9000/TM-S1000 API.



- This API may not work properly with the 64-bit driver. Thus, BiSCNMICRSetStatusBackWndEx ([page 75](#)) should be used instead with the 64-bit driver.
- When a processing status notification event is no longer needed, the BiSCNMICRCancelStatusBack API must be called to deregister the event notification handle.
- If a processing status callback is registered with the API BiSCNMICRSetStatusBackFunction, notification to the callback function registered with the BiSCNMICRSetStatusBackFunction API will be prioritized, and the event will not be notified. But even in this case, the event notification handle is registered successfully. This handle should be deregistered if not needed.

Syntax

```
int BiSCNMICRSetStatusBackWnd
    (int nHandle, long hWnd, LPDWORD lpdwTransactionNumber
    , LPWORD lpwMainStatus, LPWORD lpwSubStatus)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- hWnd: Specifies the window handle of a button that sends a click event to issue notification of the status of BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint processing.
- lpdwTransactionNumber: A memory address at which the transaction number (ID) is stored is specified.
- lpwMainStatus: A memory address at which the main processing status is stored is specified.
- lpwSubStatus: A memory address at which the sub processing status is stored is specified.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Registers the handle of the button and the memory address. The handle of the button receives a click event to indicate the processing status of BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint. The memory address receives the detail information of the processing status. Do not discard the memory address registered with this API until the registration made with BiSCNMICRCancelStatusBack has been released. Whenever there is a change in the check sheet processing status, a value is saved to lpdwTransactionNumber, lpwMainStatus, and lpwSubStatus.



For details of the processing status, refer to processing status in ["Processing status list" on page 114](#).

BiSCNMICRSetStatusBackWndEx

In order to provide a notification of the scan processing status, this records the handle of the button that sends a click event and the memory address that stores the notified value.



- When a processing status notification event is no longer needed, the BiSCNMICRCancelStatusBack API must be called to deregister the event notification handle.
- If a processing status callback is registered with the API BiSCNMICRSetStatusBackFunction, notification to the callback function registered with the BiSCNMICRSetStatusBackFunction API will be prioritized, and the event will not be notified. But even in this case, the event notification handle is registered successfully. This handle should be deregistered if not needed.

Syntax

```
int BiSCNMICRSetStatusBackWndEx
(int nHandle, HWND hWnd, LPDWORD pdwTransactionNumber
, LPWORD pwMainStatus, LPWORD pwSubStatus)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- hWnd: Specifies the window handle of a button that sends a click event to issue notification of the status of BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint processing.
- pdwTransactionNumber: A memory address at which the transaction number (ID) is stored is specified.
- pwMainStatus: A memory address at which the main processing status is stored is specified.
- pwSubStatus: A memory address at which the sub processing status is stored is specified.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

To notify the processing status of BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint, this API registers the handle of the button to which a click event is sent as well as the memory address at which the value to be notified is stored. The memory address registered with this API must not be discarded until it is deregistered using BiSCNMICRCancelStatusBack. If the processing status of the check sheet changes, values are saved in pdwTransactionNumber, pwMainStatus, and pwSubStatus.



For details of the processing status, refer to processing status in ["Processing status list" on page 114](#).

BiSCNMICRCancelStatusBack

Cancels the reading status notification request registered using either of BiSCNMICRSetStatusBackFunction.

Syntax

```
int BiSCNMICRCancelStatusBack(int nHandle)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

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



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiStartEndorsementSetStatusBackFunction

When endorsement printing is to be executed using BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint, this API registers the address of a callback function that is invoked upon transmission of endorsement print data.

If this API is called several times, the last registered callback function will be used.



- When notification of a processing status callback is no longer needed, the BiStartEndorsementCancelStatusBack API must be called to deregister the callback function.
- Even if this API is not executed, BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint can still be executed, but an endorsement print data transmission ready callback will not be notified.

Syntax

```
int BiStartEndorsementSetStatusBackFunction
(int nHandle, int (CALLBACK EXPORT* pEndorseCB)
(unsigned long ulTransactionNumber, char* pszPortName ))
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pEndorseCB: The address of a callback function is specified.

Callback function parameters

ulTransactionNumber: Transaction number (ID) corresponding to the check sheet from which the processing status has been issued.
pszPortName: Memory address at which the name of the port from which the callback has originated is stored.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiStartEndorsementSetStatusBackWnd

When endorsement printing is to be executed using BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint, this API registers the handle of the button to which a click event invoked upon transmission of endorsement print data is sent as well as the memory address at which the value to be notified is stored.

If this API is called several times, the last registered callback function will be used.



- When a processing status notification event is no longer needed, the BiStartEndorsementCancelStatusBack API must be called to deregister the event notification handle.
- Even if this API is not executed, BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint can still be executed, but an endorsement print data transmission ready callback will not be notified.

Syntax

```
int BiStartEndorsementSetStatusBackWnd
(int nHandle, HWND hWnd, unsigned long* pulTransactionNumber)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- hWnd: When endorsement printing is to be executed using BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint, this API specifies the window handle of the button to which a click event is sent to notify the timing at which to send endorsement print data.
- pulTransactionNumber: A memory address at which the transaction number (ID) is stored is specified.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

If the processing status of the check paper changes, a value is saved in pulTransactionNumber.



- The memory address registered with this API must not be discarded until it is deregistered using BiStartEndorsementCancelStatusBack.
- If a processing status callback is registered with the BiStartEndorsementSetStatusBackFunction API, notification to the callback function registered with the BiStartEndorsementSetStatusBackFunction API will be prioritized, and the event will not be notified. But even in this case, the event notification handle is registered successfully. This handle should be deregistered if not needed.

BiStartEndorsementCancelStatusBack

Cancels the processing status information notification request registered using either `BiStartEndorsementSetStatusBackFunction` or `BiStartEndorsementSetStatusBackWnd`.

Syntax

```
int BiStartEndorsementCancelStatusBack(int nHandle)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

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



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiEndEndorsementSetStatusBackFunction

When endorsement printing is to be executed using BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint, this API registers the address of a callback function that is invoked when the reception of endorsement print data is completed.

If this API is called several times, the last registered callback function will be used.



- When notification of a processing status callback is no longer needed, the BiEndEndorsementCancelStatusBack API must be called to deregister the callback function.
- Even if this API is not executed, BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint can still be executed, but an endorsement print data reception complete callback will not be notified.

Syntax

```
int BiEndEndorsementSetStatusBackFunction
    (int nHandle, int (CALLBACK EXPORT* pEndorseCB)
    (unsigned long ulTransactionNumber, char* pszPortName ))
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pEndorseCB: The address of a callback function is specified.

Callback function parameters

ulTransactionNumber: Transaction number (ID) corresponding to the check sheet from which the processing status has been issued.
pszPortName: Memory address at which the name of the port from which the callback has originated is stored.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiEndEndorsementSetStatusBackWnd

Registers the handle of the button as well as the memory address at which the value to be notified is stored. When endorsement printing is to be executed using BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint, a click event is invoked to the button via registered handle upon completion of reception of endorsement print data being sent.

If this API is called 2 or more times, the last registered callback function will be used.



- When a processing status notification event is no longer needed, the BiEndEndorsementCancelStatusBack API must be called to deregister the event notification handle.
- Even if this API is not executed, BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint can still be executed, but an endorsement print data reception complete callback will not be notified.

Syntax

```
int BiEndEndorsementSetStatusBackWnd
    (int nHandle, HWND hWnd, unsigned long* pulTransactionNumber)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- hWnd: When endorsement printing is to be executed using BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint, this API specifies the window handle of the button to which a click event is sent to notify the timing at which to complete the reception of endorsement print data.
- pulTransactionNumber: A memory address at which the transaction number (ID) is stored is specified.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

If the processing status of the check paper changes, a value is saved in `pulTransactionNumber`.



- The memory address registered with this API must not be discarded until it is deregistered using `BiEndEndorsementCancelStatusBack`.
- If a processing status callback is registered with the `BiEndEndorsementSetStatusBackFunction` API, notification to the callback function registered with the API `BiEndEndorsementSetStatusBackFunction` will be prioritized, and the event will not be notified. But even in this case, the event notification handle is registered successfully. This handle should be deregistered if not needed.

BiEndEndorsementCancelStatusBack

Cancels the processing status information notification request registered using either `BiEndEndorsementSetStatusBackFunction` or `BiEndEndorsementSetStatusBackWnd`.

Syntax

int ***BiEndEndorsementCancelStatusBack***(int nHandle)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

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



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiGetStatus

Acquires the current device status. The device status is set to the lpStatus.



The acquired device status is converted specific model's definition based on the status section of the API settings file. For more information, refer to ["API settings file" on page 391](#).

Syntax

```
int BiGetStatus (int nHandle, LPDWORD lpStatus)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
lpStatus: The current status of the device is set.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Refer to ["Device Status" on page 364](#), for the device statuses that you can acquire.

BiGetRealStatus

Acquires the real-time device status. The device status is set to the lpStatus.

Syntax

```
int BiGetRealStatus (int nHandle, LPDWORD lpStatus )
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
lpStatus: The real-time status of the device is set.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Refer to ["Device Status" on page 364](#), for the device statuses that you can acquire.

BiSetStatusBackFunction

Registers the callback function on the occasion of device status is changed.



This API is compatible with the TM-J9000/TM-S1000 API.
For the TM-S9000/S2000 API, using ["BiSetStatusBackFunctionEx" on page 88](#) is recommended.

Syntax

```
int BiSetStatusBackFunction
(int nHandle, int (CALLBACK EXPORT *pStatusCB)(DWORD dwStatus))
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pStatusCB: Address of a callback function invoked at the time of status notification.

Callback function parameters

dwStatus: The Device status held by the TM-S9000/S2000 API is set.

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	Another API is running
ERR_RESET	-400	Device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

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 364](#).

BiSetStatusBackFunctionEx

Registers the callback function on the occasion of device status is changed.

Identifies the port originating the CALLBACK, in addition to the functions of BiSetStatusBackFunction.

Syntax

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

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pStatusCB: Address of a callback function invoked at the time of status notification.

Callback function parameters

dwStatus: The Device status held by the TM-S9000/S2000 API is set.
lpcPortName: Memory address at which the name of the port from which the callback has originated is stored.

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	Another API is running
ERR_RESET	-400	Device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

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 364](#).

BiSetStatusBackWnd

Upon a status notification, this records the handle of the button that sends a button click event and the address of the memory that sets the device status.



This API is compatible with the TM-J9000/TM-S1000 API.



This API does not work properly with the 64-bit driver. Thus, ["BiSetStatusBackWndEx" on page 90](#) should be used instead with the 64-bit driver.

Syntax

int **BiSetStatusBackWnd** (int nHandle, long hWnd, LPDWORD lpStatus)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- hWnd: Upon a status notification, this records the handle of the button that sends a button click event and the address of the memory that sets the device status.
- lpStatus: The event is sent with the device status value set in lpStatus.

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	Another API is running
ERR_RESET	-400	Device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

When there has been a change in the status of the device, posts notification of a click event being sent for the specified button. BiCancelStatusBack cancels this API. For a list of device statuses that can be retrieved with this API, refer to ["Device Status" on page 364](#).

BiSetStatusBackWndEx

Registers the handle of the button to which a click event is sent at the time of status notification as well as the memory address at which the status information is set.

If device status changes, the status is notified with a click event sent to the specified button.



If a device status callback is registered with the BiSetStatusBackFunction API, notification to the callback function registered with the BiSetStatusBackFunction API will be prioritized, and the event will not be notified. But even in this case, the event notification handle is registered successfully. This handle should be deregistered if not needed.

Syntax

int **BiSetStatusBackWndEx** (int nHandle, HWND hWnd, LPDWORD pdwStatus)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
 hWnd: Upon a status notification, this records the handle of the button that sends a button click event and the address of the memory that sets the device status.
 pdwStatus: The event is sent with the device status value set in pdwStatus.

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	Another API is running
ERR_RESET	-400	Device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

When there has been a change in the status of the device, posts notification of a click event being sent for the specified button. BiCancelStatusBack cancels this API. For a list of device statuses that can be retrieved with this API, refer to ["Device Status" on page 364](#).

BiCancelStatusBack

The status notification request is deregistered.

Syntax

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

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

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



- For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).
- Returns "SUCCESS" even if you call this API by mistake while the automatic status notification request process has not been registered.

BiGetInkStatus

Acquires the current ink status.

Syntax

```
int BiGetInkStatus(int nHandle, LPWORD Status)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- Status: The current ink status is set.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the product is incorrect.
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Refer to ["Ink Status" on page 367](#) regarding the acquired Ink status.

BiSetInkStatusBackFunction

A callback function invoked at the time of ink status notification is registered.



This API is compatible with the TM-J9000 API.
For the TM-S9000/S2000 API, using ["BiSetInkStatusBackFunctionEx" on page 94](#) is recommended.

Syntax

```
int BiSetInkStatusBackFunction
(int nHandle, int (CALLBACK EXPORT *pStatusCB)(WORD wStatus))
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pStatusCB: Address of a callback function invoked at the time of ink status notification.
wStatus: The ink status held by the TM-S9000/S2000 API is set.

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	Another API is running
ERR_RESET	-400	Device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

When this API is called, the callback function is called with the ink status set in dwStatus. When the ink status changes, the callback function is called with the new information automatically set in dwStatus. BiCancelInkStatusBack cancels this API. For a list of ink statuses that can be retrieved with this API, refer to ["Ink Status" on page 367](#).

BiSetInkStatusBackFunctionEx

Registers the callback function on the occasion of ink status is changed.

Identifies the port originating the CALLBACK, in addition to the functions of BiSetInkStatusBackFunction.

Syntax

```
int BiSetInkStatusBackFunctionEx
    (int nHandle, int (CALLBACK EXPORT *pStatusCB)
    (WORD wStatus, LPSTR lpcPortName))
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pStatusCB: Address of a callback function invoked at the time of ink status notification.

Callback function parameters

wStatus: The ink status held by the TM-S9000/S2000 API is set.
lpcPortName: Memory address at which the name of the port from which the callback has originated is stored.

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	Another API is running
ERR_RESET	-400	Device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

When this API is called, the callback function is called with the ink status set in dwStatus. When the ink status changes, the callback function is called with the new information automatically set in dwStatus. BiCancelInkStatusBack cancels this API. For a list of ink statuses that can be retrieved with this API, refer to ["Ink Status" on page 367](#).

BiSetInkStatusBackWnd

Registers the handle of the button to which a click event is sent at the time of ink status notification as well as the memory address at which the status information is set.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).



- If an ink status callback is registered with the BiSetInkStatusBackFunction API, notification to the callback function registered with the BiSetInkStatusBackFunction API will be prioritized, and the event will not be notified. But even in this case, the event notification handle is registered successfully. This handle should be deregistered if not needed.
- This API does not work properly with the 64-bit driver. Thus, ["BiSetInkStatusBackWndEx" on page 96](#) should be used instead with the 64-bit driver.

Syntax

int **BiSetInkStatusBackWnd** (int nHandle, long hWnd, LPDWORD lpStatus)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- hWnd: Upon a status notification, this records the handle of the button that sends a button click event and the address of the memory that sets the ink status.
- lpStatus: The event is sent with the ink status value set in lpStatus.

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	Another API is running
ERR_RESET	-400	Device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

If the ink status changes, the status information is set at the address specified in lpStatus, and a click event is sent to the button specified in hWnd. BiCancelInkStatusBack cancels this API. For a list of ink statuses that can be retrieved with this API, refer to ["Ink Status" on page 367](#).

BiSetInkStatusBackWndEx

Registers the handle of the button to which a click event is sent at the time of ink status notification as well as the memory address at which the status information is set.



- When an ink status notification event is no longer needed, the BiCancelInkStatusBack API must be called to deregister the event notification handle.
- If an ink status callback is registered with the BiSetInkStatusBackFunction API, notification to the callback function registered with the BiSetInkStatusBackFunction API will be prioritized, and the event will not be notified. But even in this case, the event notification handle is registered successfully. This handle should be deregistered if not needed.

Syntax

```
int BiSetInkStatusBackWndEx
    (int nHandle, HWND hWnd, LPDWORD pdwStatus)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

hWnd: Upon a status notification, this records the handle of the button that sends a button click event and the address of the memory that sets the ink status.

pdwStatus: The event is sent with the ink status value set in pdwStatus.

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	Another API is running
ERR_RESET	-400	Device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

If the ink status changes, the status information is set at the address specified in lpStatus, and a click event is sent to the button specified in hWnd. BiCancelInkStatusBack cancels this API. For a list of ink statuses that can be retrieved with this API, refer to ["Ink Status" on page 367](#).

BiCancelInkStatusBack

The ink status notification request is deregistered.

Syntax

int ***BiCancelInkStatusBack***(int nHandle)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

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



- For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).
- Returns "SUCCESS" even if you call this API by mistake while the ink status notification request process has not been registered.

BiMICRSelectDataHandling

Selects the check reading operation.



This API is compatible with the TM-J9000/TM-S1000 API.

Syntax

int **BiMICRSelectDataHandling**
(int nHandle, BYTE charSelect, BYTE detailSelect, BYTE errorSelect)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

charSelect: Specifies handling of characters that cannot be analyzed.

Value	Description
0	Interrupts analysis processing at the point when characters that cannot be analyzed are detected and does not add the reading data.
1	Replaces characters which cannot be analyzed with a '?' and continues analysis processing, then the reading data are added.

detailSelect: Not used.

errorSelect: Specifies whether to finish or to continue the reading process when an error occurs. However, the reading process continues when it finishes normally or when an error that adds reading results occurs, irrespective of this setting. The following values can be set individually or as a logical sum.

Constant	Value	Description
ES_STOP_ALL	0	Finishes the reading process when an error occurs.
ES_CONTINUE_ALL	1	Continues the reading process when an error occurs if continuing is possible. The 4 errors where continuing is possible are double feed, magnetic waveform detection error, unrecognized character detection error, and noise error.
ES_CONTINUE_DOUBLEFEED	2	Continues the reading process when double feed is detected.
ES_CONTINUE_NODATA	4	Continues the reading process when a magnetic waveform detection error occurs.
ES_CONTINUE_BADDATA	8	Continues the reading process when an unrecognized character detection error occurs.
ES_CONTINUE_NOISE	16	Continues the reading process when a noise error occurs.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value specified for the product is incorrect.
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSCNSelectScanUnit

Sets the unit that operates image scanning.

Syntax

```
int BiSCNSelectScanUnit(int nHandle, BYTE bSelectUnit)
```

Argument

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

bSelectUnit: Specifies the unit to scan. The selectable value is as follows.

Constant	Value	Description
EPS_BI_SCN_UNIT_CHECKPAPER	48	Unit for check reading
EPS_BI_SCN_UNIT_CARD	49	Unit for card scanning

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The following methods are enabled for the selected facilities:

- BiSCNSetImageQuality
- BiSCNSetScanArea
- BiSCNSetImageFormat
- BiSCNSetCroppingArea
- BiSCNGetImageQuality
- BiSCNGetScanArea
- BiSCNGetImageFormat
- BiSCNGetCroppingArea
- BiESCNStoreImage

BiSCNSetImageTypeOption

The light source to be used for scan execution is selected.

Scan processing is executed using the light source specified with this API.



When scanning, the light source specified with this API is used to retrieve images.

Syntax

```
int BiSCNSetImageTypeOption(int nHandle, unsigned long ulOption)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

ulOption: Image options are specified. The following values can be specified independently:

Constant	Value	Used Light Source
EPS_BI_SCN_OPTION_COLOR (Default)	0	RGB color
EPS_BI_SCN_OPTION_IR	1	Infrared
EPS_BI_SCN_OPTION_DROPOUT_RED	2	Red of RGB color
EPS_BI_SCN_OPTION_DROPOUT_GREEN	3	Green of RGB color
EPS_BI_SCN_OPTION_DROPOUT_BLUE	4	Blue of RGB color
EPS_BI_SCN_OPTION_COLOR_IR	8	RGB color and infrared
EPS_BI_SCN_OPTION_GRAYSCALE	9	RGB grayscale
EPS_BI_SCN_OPTION_GRAYSCALE_IR	10	RGB grayscale and infrared
EPS_BI_SCN_OPTION_DROPOUT_RED_IR	11	Red of RGB color and infrared
EPS_BI_SCN_OPTION_DROPOUT_GREEN_IR	12	Green of RGB color and infrared
EPS_BI_SCN_OPTION_DROPOUT_BLUE_IR	13	Blue of RGB color and infrared
EPS_BI_SCN_OPTION_UV	14	Ultraviolet
EPS_BI_SCN_OPTION_COLOR_UV	15	RGB color and ultraviolet
EPS_BI_SCN_OPTION_GRAYSCALE_UV	16	RGB grayscale and ultraviolet
EPS_BI_SCN_OPTION_DROPOUT_RED_UV	17	Red of RGB color and ultraviolet
EPS_BI_SCN_OPTION_DROPOUT_GREEN_UV	18	Green of RGB color and ultraviolet
EPS_BI_SCN_OPTION_DROPOUT_BLUE_UV	19	Blue of RGB color and ultraviolet

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

With dropout color, a specified color for the light source is used for scanning. The specified color in the acquired image disappears, and the image is grayscale. The effects vary depending on the color.

Note

- If BiSCNSelectScanUnit has been called first, the unit setting specified with that API is updated.
- If BiSCNSelectScanFace has been called first, the scan face setting specified with that API is updated.
- Some light sources cannot be used, depending on the BiSCNSelectScanUnit settings.
 - Allowed ulOption for scan unit (1 pass Photo ID scan)

ulOption	BiSCNSelectScanUnit	
	Check Paper	Card
EPS_BI_SCN_OPTION_COLOR	✓	✓
EPS_BI_SCN_OPTION_IR	✓	✓
EPS_BI_SCN_OPTION_DROPOUT_RED	✓	✓
EPS_BI_SCN_OPTION_DROPOUT_GREEN	✓	✓
EPS_BI_SCN_OPTION_DROPOUT_BLUE	✓	✓
EPS_BI_SCN_OPTION_COLOR_IR	✓	✓(*1)
EPS_BI_SCN_OPTION_GRAYSCALE	✓	✓
EPS_BI_SCN_OPTION_GRAYSCALE_IR	✓	✓(*1)
EPS_BI_SCN_OPTION_DROPOUT_RED_IR	✓	✓(*1)
EPS_BI_SCN_OPTION_DROPOUT_GREEN_IR	✓	✓(*1)
EPS_BI_SCN_OPTION_DROPOUT_BLUE_IR	✓	✓(*1)
EPS_BI_SCN_OPTION_UV	✓	✓
EPS_BI_SCN_OPTION_COLOR_UV	✓	✓(*1)
EPS_BI_SCN_OPTION_GRAYSCALE_UV	✓	✓(*1)
EPS_BI_SCN_OPTION_DROPOUT_RED_UV	✓	✓(*1)
EPS_BI_SCN_OPTION_DROPOUT_GREEN_UV	✓	✓(*1)
EPS_BI_SCN_OPTION_DROPOUT_BLUE_UV	✓	✓(*1)

*1: The setting is allowed with the TM-S2000II UV model and the TM-S2000 UV model.

Whether 1 pass card scanning in card mode is valid can be found with 5-bit of Type ID (B) in Device ID.

Detail can be found in [page 383](#).

- Image Data Type sent from firmware to driver

2nd parameter of BiSCNSetImage- TypeOption	Color of light source					# of image plane per side	Byte per pixel in vis- ible light source image
	Red	Green	Blue	Infrared	UV		
EPS_BI_SCN_OPTION_COLOR	On	On	On	-	-	1	3
EPS_BI_SCN_OPTION_IR	-	-	-	On	-	1	n/a
EPS_BI_SCN_OPTION_DROPOUT_RED	On	-	-	-	-	1	1
EPS_BI_SCN_OPTION_DROPOUT_GREEN	-	On	-	-	-	1	1
EPS_BI_SCN_OPTION_DROPOUT_BLUE	-	-	On	-	-	1	1
EPS_BI_SCN_OPTION_COLOR_IR	On	On	On	On	-	2	3
EPS_BI_SCN_OPTION_GRAYSCALE	On	On	On	-	-	1	1
EPS_BI_SCN_OPTION_GRAYSCALE_IR	On	On	On	On	-	2	1
EPS_BI_SCN_OPTION_DROPOUT_RED_IR	On	-	-	On	-	2	1
EPS_BI_SCN_OPTION_DROPOUT_GREEN_IR	-	On	-	On	-	2	1
EPS_BI_SCN_OPTION_DROPOUT_BLUE_IR	-	-	On	On	-	2	1
EPS_BI_SCN_OPTION_UV	-	-	-	-	On	1	n/a
EPS_BI_SCN_OPTION_COLOR_UV	On	On	On	-	On	2	3
EPS_BI_SCN_OPTION_GRAYSCALE_UV	On	On	On	-	On	2	1
EPS_BI_SCN_OPTION_DROPOUT_RED_UV	On	-	-	-	On	2	1
EPS_BI_SCN_OPTION_DROPOUT_GREEN_UV	-	On	-	-	On	2	1
EPS_BI_SCN_OPTION_DROPOUT_BLUE_UV	-	-	On	-	On	2	1

- Image Data Type sent to application

* inconsistent setting

2nd parameter of BiSCNSelectScanImage	2nd parameter of BiSCNSetImage-TypeOption	2nd parameter of BiSCNSetImageQuality	Image Type	Bit per pixel
MF_SCAN_IMAGE_VISIBLE	• EPS_BI_SCN_OPTION_COLOR	EPS_BI_SCN_1BIT	Black & White	1
	• EPS_BI_SCN_OPTION_COLOR_IR	EPS_BI_SCN_8BIT	Grayscale	8
	• EPS_BI_SCN_OPTION_COLOR_UV	EPS_BI_SCN_24BIT	Color	24
	• EPS_BI_SCN_OPTION_GRAYSCALE • EPS_BI_SCN_OPTION_DROPOUT_RED • EPS_BI_SCN_OPTION_DROPOUT_GREEN • EPS_BI_SCN_OPTION_DROPOUT_BLUE	EPS_BI_SCN_1BIT	Black & White	1
	• EPS_BI_SCN_OPTION_GRAYSCALE_IR • EPS_BI_SCN_OPTION_DROPOUT_RED_IR • EPS_BI_SCN_OPTION_DROPOUT_GREEN_IR • EPS_BI_SCN_OPTION_DROPOUT_BLUE_IR	EPS_BI_SCN_8BIT	Grayscale	8
	• EPS_BI_SCN_OPTION_GRAYSCALE_UV • EPS_BI_SCN_OPTION_DROPOUT_RED_UV • EPS_BI_SCN_OPTION_DROPOUT_GREEN_UV • EPS_BI_SCN_OPTION_DROPOUT_BLUE_UV	EPS_BI_SCN_24BIT	Grayscale	8*
MF_SCAN_IMAGE_INFRA-RED	• EPS_BI_SCN_OPTION_GRAYSCALE_IR • EPS_BI_SCN_OPTION_DROPOUT_RED_IR • EPS_BI_SCN_OPTION_DROPOUT_GREEN_IR	EPS_BI_SCN_1BIT	Black & White	1
	• EPS_BI_SCN_OPTION_DROPOUT_BLUE_IR • EPS_BI_SCN_OPTION_GRAYSCALE_UV • EPS_BI_SCN_OPTION_DROPOUT_RED_UV	EPS_BI_SCN_8BIT	Grayscale	8
	• EPS_BI_SCN_OPTION_DROPOUT_GREEN_UV • EPS_BI_SCN_OPTION_DROPOUT_BLUE_UV	EPS_BI_SCN_24BIT	Grayscale	8*
	• EPS_BI_SCN_OPTION_IR • EPS_BI_SCN_OPTION_UV • EPS_BI_SCN_OPTION_COLOR_IR • EPS_BI_SCN_OPTION_COLOR_UV	EPS_BI_SCN_1BIT	Black & White	1
		EPS_BI_SCN_8BIT	Grayscale	8
		EPS_BI_SCN_24BIT	Grayscale	8*

BiSCNGetImageTypeOption

Acquires the action of the light source during scan that is retained in the driver.



A value specifiable by BiSCNSetImageTypeOption can be acquired.

Syntax

int **BiSCNGetImageTypeOption** (int nHandle, unsigned long* pulOption)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
 pulOption: Specifies the memory address where the action of the light source during scan is set.
 The set value is as shown below.

Constant	Value	Used Light Source
EPS_BI_SCN_OPTION_COLOR	0	RGB color
EPS_BI_SCN_OPTION_IR	1	Infrared
EPS_BI_SCN_OPTION_DROPOUT_RED	2	Red of RGB color
EPS_BI_SCN_OPTION_DROPOUT_GREEN	3	Green of RGB color
EPS_BI_SCN_OPTION_DROPOUT_BLUE	4	Blue of RGB color
EPS_BI_SCN_OPTION_COLOR_IR	8	RGB color and infrared
EPS_BI_SCN_OPTION_GRAYSCALE	9	RGB grayscale
EPS_BI_SCN_OPTION_GRAYSCALE_IR	10	RGB grayscale and infrared
EPS_BI_SCN_OPTION_DROPOUT_RED_IR	11	Red of RGB color and infrared
EPS_BI_SCN_OPTION_DROPOUT_GREEN_IR	12	Green of RGB color and infrared
EPS_BI_SCN_OPTION_DROPOUT_BLUE_IR	13	Blue of RGB color and infrared
EPS_BI_SCN_OPTION_UV	14	Ultraviolet
EPS_BI_SCN_OPTION_COLOR_UV	15	RGB color and ultraviolet
EPS_BI_SCN_OPTION_GRAYSCALE_UV	16	RGB grayscale and ultraviolet
EPS_BI_SCN_OPTION_DROPOUT_RED_UV	17	Red of RGB color and ultraviolet
EPS_BI_SCN_OPTION_DROPOUT_GREEN_UV	18	Green of RGB color and ultraviolet
EPS_BI_SCN_OPTION_DROPOUT_BLUE_UV	19	Blue of RGB color and ultraviolet

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 be used because the other API is being executed
ERR_RESET	-400	Process not being executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSCNMICRFunctionContinuously

Scans images successively, and reads MICR characters.



- This API allows for the scanning of ID cards.
- When scanning cards, IQA is not executed even if it has been set.
- The endorsement done with this API is carried out only on the station set immediately before.
- When printing the endorsement done with this API, set the station with BiSetPrintStation before executing this API.

Syntax

```
int BiSCNMICRFunctionContinuously
    (int nHandle, LPVOID lpvStruct, WORD wFunction)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

lpvStruct: The address of the parameter structure specified for each unit.
Refer to ["Supplemental Description for lpvStruct" on page 107](#).

wFunction (Constant)	Parameter to Set for lpvStruct
MF_EXEC	"NULL"
MF_SET_BASE_PARAM	Address of the MF_BASE01 structure
MF_SET_MICR_PARAM	Address of the MF_MICR01 structure
MF_SET_SCAN_FRONT_PARAM	Address of the MF_SCAN structure for front of face
MF_SET_SCAN_BACK_PARAM	Address of the MF_SCAN structure for back of face
MF_SET_PRINT_PARAM	Address of the MF_PRINT01 structure
MF_SET_PROCESS_PARAM	Address of the MF_PROCESS01 structure
MF_SET_IQA_PARAM	Address of the MF_IQA/MF_IQA01 structure
MF_SET_BARCODE_FRONT_PARAM	Address of the MF_BARCODE structure for front of face
MF_SET_BARCODE_BACK_PARAM	Address of the MF_BARCODE structure for back of face
MF_CLEAR_BASE_PARAM	Address of the MF_BASE01 structure
MF_CLEAR_MICR_PARAM	Address of the MF_MICR01 structure
MF_CLEAR_SCAN_FRONT_PARAM	Address of the MF_SCAN structure for front of face
MF_CLEAR_SCAN_BACK_PARAM	Address of the MF_SCAN structure for back of face
MF_CLEAR_PRINT_PARAM	Address of the MF_PRINT01 structure
MF_CLEAR_PROCESS_PARAM	Address of the MF_PROCESS01 structure
MF_CLEAR_IQA_PARAM	Address of the MF_IQA/MF_IQA01 structure
MF_CLEAR_BARCODE_FRONT_PARAM	Address of the MF_BARCODE structure for front of face
MF_CLEAR_BARCODE_BACK_PARAM	Address of the MF_BARCODE structure for back of face
MF_GET_BASE_DEFAULT	Address of the MF_BASE01 structure
MF_GET_MICR_DEFAULT	Address of the MF_MICR01 structure
MF_GET_SCAN_FRONT_DEFAULT	Address of the MF_SCAN structure for front of face
MF_GET_SCAN_BACK_DEFAULT	Address of the MF_SCAN structure for back of face
MF_GET_PRINT_DEFAULT	Address of the MF_PRINT01 structure
MF_GET_PROCESS_DEFAULT	Address of the MF_PROCESS01 structure
MF_GET_IQA_DEFAULT	Address of the MF_IQA/MF_IQA01 structure
MF_GET_BARCODE_FRONT_DEFAULT	Address of the MF_BARCODE structure for front of face
MF_GET_BARCODE_BACK_DEFAULT	Address of the MF_BARCODE structure for back of face

wFunction: Specifies the functions for the API to execute. For settings regarding reading, by specifying MF_SET_MICR_PARAM and so on and executing this function, the API is notified. If the members of the structure are changed after notification, it is necessary to perform notification again with this function. Include the header file submitted for the definition name used for this API.

Refer to ["Supplemental Description for wFunction" on page 108](#).

Supplemental Description for lpvStruct

For an explanation of the lpvStruct structure, refer to ["Structures" on page 277](#). However, the following values are not used with this API, or the values are not set.

<MF_BASE01 structure>

- dwNotifyType and uNotifyHandle are not used.
Notification of the reading status of this API is performed by the handler registered with BiSCNMICRSet-StatusBackFunction.
- hProgressWnd is not used
This API does not provide notification of the progress status.

<MF_SCAN structure>

- wImageID is not used.
Set the transaction number (ID) using BiSetTransactionNumber
- Do not set values in bStatus, bDetail, dwXSize, dwYSize, dwScanSize, and lpbScanData.
Set values when getting the scan image with BiGetScanImage.

<MF_MICR01 structure>

- bMicOcrSelect and blParsing are not used. Use them when executing BiGetMicrText.
- Do not set values in bStatus, bDetail, szMicrStr, stOcrReliableInfo, szAccountNumber, szAmount, szBankNumber, szSerialNumber, szEPC, szTransitNumber, lCheckType, and lCountryCode.
Set values when getting the MICR (OCR) text with BiGetMicrText.

Supplemental Description for wFunction

wFunction (Constant)	Description
MF_EXEC	SCAN / MICR / Transaction printing is carried out according to the specified parameters. The second parameter is ignored.
MF_SET_BASE_PARAM	This sets base parameters. The MF_BASE01 structure address is specified in lpvStruct.
MF_SET_MICR_PARAM	This sets MICR parameters. The MF_MICR structure address is specified in lpvStruct.
MF_SET_SCAN_FRONT_PARAM	This sets front side read scanning parameters. The MF_SCAN structure address is specified in lpvStruct. The MF_SCAN structure has a different address from the structure for reading the back side. When the same address is specified, ERR_PARAM is returned. The operation is identical when MF_SET_SCAN_PARAM is specified.
MF_SET_SCAN_BACK_PARAM	This sets the back side read scanning parameters. The MF_SCAN structure address is specified in lpvStruct. The MF_SCAN structure has a different address from the structure for reading the front side. When the same address is specified, ERR_PARAM is returned.
MF_SET_PRINT_PARAM	This sets transaction printing parameters. The MF_PRINT01 structure address is specified in lpvStruct.
MF_SET_PROCESS_PARAM	This sets process parameters. The MF_PROCESS01 structure address is specified in lpvStruct.
MF_SET_IQA_PARAM	This sets IQA parameters. The MF_IQA/MF_IQA01 structure address is specified in lpvStruct.
MF_SET_BARCODE_FRONT_PARAM	This sets the front side barcode decode parameters. The MF_BARCODE structure address is specified in lpvStruct.
MF_SET_BARCODE_BACK_PARAM	This sets the back side barcode decode parameters. The MF_BARCODE structure address is specified in lpvStruct.
MF_CLEAR_BASE_PARAM	This clears all the specifications for BASE/ MICR/ SCAN/ PRINT/PROCESS/ BARCODE/ IQA parameters. lpvStruct values are ignored.
MF_CLEAR_MICR_PARAM	This clears the MICR parameter specifications. lpvStruct values are ignored.
MF_CLEAR_SCAN_FRONT_PARAM	This clears the scan parameter specifications. lpvStruct values are ignored. The operation is identical when MF_CLEAR_SCAN_PARAM is specified.
MF_CLEAR_SCAN_BACK_PARAM	This clears the scan parameter specifications. lpvStruct values are ignored.
MF_CLEAR_PRINT_PARAM	This clears the transaction printing parameter specifications. lpvStruct values are ignored.
MF_CLEAR_PROCESS_PARAM	This clears the process parameter specifications. lpvStruct values are ignored.
MF_CLEAR_IQA_PARAM	This clears the IQA parameter specifications. lpvStruct values are ignored.
MF_CLEAR_BARCODE_FRONT_PARAM	This clears the front side barcode decode parameters specifications. lpvStruct values are ignored.
MF_CLEAR_BARCODE_BACK_PARAM	This clears the back side barcode decode parameters specifications. lpvStruct values are ignored.

wFunction (Constant)	Description
MF_GET_BASE_DEFAULT	This obtains the initial values for the device base structure. (Refer to "Default Values of the MF_BASE01 Structure" on page 280.)
MF_GET_MICR_DEFAULT	This obtains the initial values for the device MICR structure. (Refer to "Default Values of the MF_MICR01 Structure" on page 289.)
MF_GET_SCAN_DEFAULT	This obtains the initial values for the device SCAN structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_SCAN_FRONT_DEFAULT	This obtains the initial values for the device SCAN (front side) structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_SCAN_BACK_DEFAULT	This obtains the initial values for the device SCAN (back side) structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_PRINT_DEFAULT	This obtains the initial values for the transaction printing structure. With API, the initial values of iSize and iVersion are not returned. The application must specify these two values. Also, the initial value for the structure must be obtained after zero clear of all member variables except iSize and iVersion. (Refer to "Default Values of the MF_PRINT01 Structure" on page 297.)
MF_GET_PROCESS_DEFAULT	This obtains the initial values for the process structure. (Refer to "Default Values of the MF_PROCESS01 Structure" on page 311.)
MF_GET_IQA_DEFAULT	This obtains the initial values for the IQA structure. (Refer to "Default Values of the MF_IQA Structure" on page 320.)
MF_GET_BARCODE_FRONT_DEFAULT	This obtains the initial values for the device Barcode (front side) structure. (Refer to "Default Values of the MF_BARCODE Structure" on page 359.)
MF_GET_BARCODE_BACK_DEFAULT	This obtains the initial values for the device Barcode (back side) structure. (Refer to "Default Values of the MF_BARCODE Structure" on page 359.)

Return value

BiSCNMICRFunctionContinuously

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
ERR_OFFLINE	-110	Waiting to return from an offline state
ERR_WITHOUT_CB	-130	Cannot be executed as neither of BiSCNMICRSetStatusBackFunction/BiSCNMICRSetStatusBackWnd is invoked
ERR_PAPERINSERT_TIMEOUT	-300	Failed to insert paper
ERR_EXEC_FUNCTION	-310	Another API is running
ERR_RESET	-400	Device is being reset
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_SCAN	-450	Failed to read the image data
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_PAPER_PILED	-1010	Paper pilling error
ERR_PAPER_JAM	-1020	Paper jam error
ERR_COVER_OPEN	-1030	Cover open error
ERR_MICR_NODATA	-1040	MICR data is not existing
ERR_MICR_BADDATA	-1050	MICR data is not able to recognize
ERR_MICR_NOISE	-1070	Noise error has occurred during MICR reading
ERR_SCN_COMPRESS	-1080	Scan image data compressing error
ERR_PAPER_EXIST	-1090	API can not be execute because there is a paper on the path
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_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_SCAN	-450	Failed to read the image data
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_NOT_EXEC	-470	Reading process not executed
ERR_PAPER_PILED	-1010	Paper pilling error
ERR_PAPER_JAM	-1020	Paper jam error
ERR_COVER_OPEN	-1030	Cover open error
ERR_MICR_NODATA	-1040	MICR data is not existing
ERR_MICR_BADDATA	-1050	MICR data is not able to recognize
ERR_MICR_NOISE	-1070	Noise error has occurred during MICR reading
ERR_SCN_COMPRESS	-1080	Scan image data compressing error
ERR_PAPER_EXIST	-1090	API can not be execute because there is a paper on the path
ERR_PAPER_INSERT	-1100	Failed to insert paper

MF_SCAN.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_SCAN	-450	Device failed in image scanning
ERR_NOT_EXEC	-470	Process not being executed
ERR_SCN_COMPRESS	-1080	Scan image data compressing error

MF_MICR01.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_BUFFER_OVER_FLOW	-140	Insufficient buffer error
ERR_NOT_FOUND	-220	Data not found
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_NOT_EXEC	-470	Reading process not executed
ERR_MICR_NODATA	-1040	MICR data is not existing
ERR_MICR_BADDATA	-1050	MICR data is not able to recognize
ERR_MICR_NOISE	-1070	Noise error has occurred during MICR reading

MF_BARCODE.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	Data not found
ERR_NOT_EXEC	-470	Process not being executed
ERR_BARCODE_NODATA	-1130	Barcode cannot be detected

MF_PRINT01.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_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_NOT_EXEC	-470	Reading process not executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- For SCAN/MICR/transaction printing, use a structure to specify the parameters of the functions you wish to use and call them using this API. MF_SET_BASE_PARAM must always be set. When a structure is set again, change the members of the structure and then call this API again by specifying MF_SET_xxxx_PARAM. The structures contain all the results output by the TM-S9000/S2000 API, up until the execution of BiCloseMon-Printer.
- By specifying MF_EXEC as the 3rd parameter, the specified functions are executed for all check sheet inserted in the feeder. When the feeder is empty, it stops automatically. Be sure not to discard the structure in order to set the return value to the structure. Be sure to invoke MF_CLEAR_xxxx_PARAM before discarding the structure.
- After the processing has been returned from the handler that performs MF_DATARECEIVE_DONE notification processing to the TM-S9000/S2000 API, electronic endorsement is executed.
- A scan is performed using the highest resolution level set in the MF_SCAN structure.
- When a scan is executed with this API, even if RGB and infrared has been specified as the scan light source using BiSCNSetImageTypeOption in advance, only an image scanned using the light source specified using BiSCNSelectScanImage can be captured.
- To capture both RGB images for card scan, either BiSCNMICRFunctionContinuously or BiSCNMICRFunctionPostPrint is used.
- A single sheet of check sheet is read, and the read result is stored in the class set in the driver.



- Before invoking this API, execute of BiSCNMICRSetStatusBackFunction.
- When MF_CONTINUE, MF_MICR_RETRANS, MF_SCAN_FRONT_RETRANS, and MF_SCAN_BACK_RETRANS are specified in the 3rd parameter, ERR_NOT_SUPPORT is returned.

Processing status list

Main status	Sub status	Outline
MF_FUNCTION_START	-	The check sheet read processing was started.
MF_CHECKPAPER_PROCESS_START	-	The check sheet was inserted and the processing was started.
MF_DATARECEIVE_START	-	The data reception processing was started.
MF_DATARECEIVE_DONE	-	The data reception processing was completed.
MF_CHECKPAPER_PROCESS_DONE	-	The check sheet was ejected and the processing was completed.
MF_FUNCTION_DONE	MF_BASE01.iRet	The check sheet read processing was completed.
MF_ERROR_OCCURRED	ERR_PAPER_PILED	Double feed was detected. (Load results: bit5)
	ERR_FORM_LENGTH	A paper length error occurred. (Detailed load results: 44H, 45H)
	ERR_PAPER_JAM	A paper jam error occurred. (Detailed load results: 46H)
	ERR_MECHANICAL	A mechanical error occurred. (Detailed load results: 47H)
	ERR_COVER_OPEN	The processing was terminated because the cover was opened. (Detailed load results: 48H)
	ERR_MICR_NODATA	Magnetic waveform detection error (MICR details: 45H)
	ERR_MICR_BADDATA	Unanalyzable character detection error (MICR details: 46H)
	ERR_MICR_NOISE	Noise error (MICR details: 47H)
	ERR_PRINT_- DATA_LENGTH_EXCEED	the printing content exceeds printable area
	ERR_PRINT_DATA_UNRE- CEIVE	Print data unreceived error occurred.
	ERR_SCN_COMPRESS	Data compression error (SCN details: 47H)
	ERR_SCN_IQA	NOT_PASS is detected at the IQA validation.
	ERR_BARCODE_NODATA	Barcode cannot be detected. (BARCODE details: 45H)

BiSCNMICRFunctionPostPrint

Scans images, and reads MICR characters.



- This API allows for the scanning of ID cards.
- When scanning cards, IQA is not executed even if it has been set.
- The endorsement done with this API is carried out only on the station set immediately before.
- When printing the endorsement done with this API, set the station with BiSetPrintStation before executing this API.

Syntax

```
int BiSCNMICRFunctionPostPrint
    (int nHandle, LPVOID lpvStruct, WORD wFunction)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

lpvStruct: The address of the parameter structure specified for each unit.
Refer to ["Supplemental Description for lpvStruct" on page 116](#).

wFunction (Constant)	Parameter to Set for lpvStruct
MF_EXEC	"NULL"
MF_SET_BASE_PARAM	Address of the MF_BASE01 structure
MF_SET_MICR_PARAM	Address of the MF_MICR01 structure
MF_SET_SCAN_FRONT_PARAM	Address of the MF_SCAN structure for front of face
MF_SET_SCAN_BACK_PARAM	Address of the MF_SCAN structure for back of face
MF_SET_PRINT_PARAM	Address of the MF_PRINT01 structure
MF_SET_PROCESS_PARAM	Address of the MF_PROCESS01 structure
MF_SET_IQA_PARAM	Address of the MF_IQA/MF_IQA01 structure
MF_SET_BARCODE_FRONT_PARAM	Address of the MF_BARCODE structure for front of face
MF_SET_BARCODE_BACK_PARAM	Address of the MF_BARCODE structure for back of face
MF_CLEAR_BASE_PARAM	Address of the MF_BASE01 structure
MF_CLEAR_MICR_PARAM	Address of the MF_MICR01 structure
MF_CLEAR_SCAN_FRONT_PARAM	Address of the MF_SCAN structure for front of face
MF_CLEAR_SCAN_BACK_PARAM	Address of the MF_SCAN structure for back of face
MF_CLEAR_PRINT_PARAM	Address of the MF_PRINT01 structure
MF_CLEAR_PROCESS_PARAM	Address of the MF_PROCESS01 structure
MF_CLEAR_IQA_PARAM	Address of the MF_IQA/MF_IQA01 structure
MF_CLEAR_BARCODE_FRONT_PARAM	Address of the MF_BARCODE structure for front of face
MF_CLEAR_BARCODE_BACK_PARAM	Address of the MF_BARCODE structure for back of face
MF_GET_BASE_DEFAULT	Address of the MF_BASE01 structure
MF_GET_MICR_DEFAULT	Address of the MF_MICR01 structure
MF_GET_SCAN_FRONT_DEFAULT	Address of the MF_SCAN structure for front of face
MF_GET_SCAN_BACK_DEFAULT	Address of the MF_SCAN structure for back of face
MF_GET_PRINT_DEFAULT	Address of the MF_PRINT01 structure
MF_GET_PROCESS_DEFAULT	Address of the MF_PROCESS01 structure
MF_GET_IQA_DEFAULT	Address of the MF_IQA/MF_IQA01 structure

wFunction (Constant)	Parameter to Set for lpvStruct
MF_GET_BARCODE_FRONT_DEFAULT	Address of the MF_BARCODE structure for front of face
MF_GET_BARCODE_BACK_DEFAULT	Address of the MF_BARCODE structure for back of face

wFunction: Specifies the functions for the API to execute. For settings regarding reading, by specifying MF_SET_MICR_PARAM and so on and executing this function, the API is notified. If the members of the structure are changed after notification, it is necessary to perform notification again with this function. Include the header file submitted for the definition name used for this API.

Refer to ["Supplemental Description for wFunction" on page 117](#).

Supplemental Description for lpvStruct

For an explanation of the lpvStruct structure, refer to ["Structures" on page 277](#). However, the following values are not used with this API, or the values are not set.

<MF_BASE01 structure>

- dwNotifyType and uNotifyHandle are not used.

Notification of the reading status of this API is performed by the handler registered with BiSCNMICRSet-StatusBackFunction.

- hProgressWnd is not used

This API does not provide notification of the progress status.

<MF_SCAN structure>

- wImageID is not used.

Set the transaction number (ID) using BiSetTransactionNumber

- Do not set values in bStatus, bDetail, dwXSize, dwYSize, dwScanSize, and lpbScanData.

Set values when getting the scan image with BiGetScanImage.

<MF_MICR01 structure>

- bMicOcrSelect and blParsing are not used. Use them when executing BiGetMicrText.

- Do not set values in bStatus, bDetail, szMicrStr, stOcrReliableInfo, szAccountNumber, szAmount, szBankNumber, szSerialNumber, szEPC, szTransitNumber, lCheckType, and lCountryCode.

Set values when getting the MICR (OCR) text with BiGetMicrText.

Supplemental Description for wFunction

wFunction (Constant)	Description
MF_EXEC	SCAN / MICR / Transaction printing is carried out according to the specified parameters. The second parameter is ignored.
MF_SET_BASE_PARAM	This sets base parameters. The MF_BASE01 structure address is specified in lpvStruct.
MF_SET_MICR_PARAM	This sets MICR parameters. The MF_MICR01 structure address is specified in lpvStruct.
MF_SET_SCAN_FRONT_PARAM	This sets front side read scanning parameters. The MF_SCAN structure address is specified in lpvStruct. The MF_SCAN structure has a different address from the structure for reading the back side. When the same address is specified, ERR_PARAM is returned. The operation is identical when MF_SET_SCAN_PARAM is specified.
MF_SET_SCAN_BACK_PARAM	This sets the back side read scanning parameters. The MF_SCAN structure address is specified in lpvStruct. The MF_SCAN structure has a different address from the structure for reading the front side. When the same address is specified, ERR_PARAM is returned.
MF_SET_PRINT_PARAM	This sets transaction printing parameters. The MF_PRINT01 structure address is specified in lpvStruct.
MF_SET_PROCESS_PARAM	This sets process parameters. The MF_PROCESS01 structure address is specified in lpvStruct.
MF_SET_IQA_PARAM	This sets IQA parameters. The MF_IQA/MF_IQA01 structure address is specified in lpvStruct.
MF_SET_BARCODE_FRONT_PARAM	This sets the front side barcode decode parameters. The MF_BARCODE structure address is specified in lpvStruct.
MF_SET_BARCODE_BACK_PARAM	This sets the back side barcode decode parameters. The MF_BARCODE structure address is specified in lpvStruct.
MF_CLEAR_BASE_PARAM	This clears all the specifications for BASE/ MICR/ SCAN/ PRINT/ PROCESS/ BARCODE/ IQA parameters. lpvStruct values are ignored.
MF_CLEAR_MICR_PARAM	This clears the MICR parameter specifications. lpvStruct values are ignored.
MF_CLEAR_SCAN_FRONT_PARAM	This clears the scan parameter specifications. lpvStruct values are ignored. The operation is identical when MF_CLEAR_SCAN_PARAM is specified.
MF_CLEAR_SCAN_BACK_PARAM	This clears the scan parameter specifications. lpvStruct values are ignored.
MF_CLEAR_PRINT_PARAM	This clears the transaction printing parameter specifications. lpvStruct values are ignored.
MF_CLEAR_PROCESS_PARAM	This clears the process parameter specifications. lpvStruct values are ignored.
MF_CLEAR_IQA_PARAM	This clears the IQA parameter specifications. lpvStruct values are ignored.
MF_CLEAR_BARCODE_FRONT_PARAM	This clears the front side barcode decode parameters specifications. lpvStruct values are ignored.
MF_CLEAR_BARCODE_BACK_PARAM	This clears the back side barcode decode parameters specifications. lpvStruct values are ignored.

wFunction (Constant)	Description
MF_GET_BASE_DEFAULT	This obtains the initial values for the device base structure. (Refer to "Default Values of the MF_BASE01 Structure" on page 280.)
MF_GET_MICR_DEFAULT	This obtains the initial values for the device MICR structure. (Refer to "Default Values of the MF_MICR01 Structure" on page 289.)
MF_GET_SCAN_DEFAULT	This obtains the initial values for the device SCAN structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_SCAN_FRONT_DEFAULT	This obtains the initial values for the device SCAN (front side) structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_SCAN_BACK_DEFAULT	This obtains the initial values for the device SCAN (back side) structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_PRINT_DEFAULT	This obtains the initial values for the transaction printing structure. With API, the initial values of iSize and iVersion are not returned. The application must specify these two values. Also, the initial value for the structure must be obtained after zero clear of all member variables except iSize and iVersion. (Refer to "Default Values of the MF_PRINT01 Structure" on page 297.)
MF_GET_PROCESS_DEFAULT	This obtains the initial values for the device Barcode (front side) structure. (Refer to "Default Values of the MF_BARCODE Structure" on page 359.)
MF_GET_IQA_DEFAULT	This obtains the initial values for the device Barcode (back side) structure. (Refer to "Default Values of the MF_BARCODE Structure" on page 359.)
MF_GET_BARCODE_FRONT_DEFAULT	This obtains the initial values for the device Barcode (front side) structure.
MF_GET_BARCODE_BACK_DEFAULT	This obtains the initial values for the device Barcode (back side) structure.

Return value

BiSCNMICRFunctionPostPrint

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
ERR_OFFLINE	-110	Waiting to return from an offline state
ERR_WITHOUT_CB	-130	Cannot be executed as neither of BiSCNMICRSetStatusBackFunction/BiSCNMICRSetStatusBackWnd is invoked
ERR_PAPERINSERT_TIMEOUT	-300	Failed to insert paper
ERR_EXEC_FUNCTION	-310	Another API is running
ERR_RESET	-400	Device is being reset
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_SCAN	-450	Failed to read the image data
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_NOT_EXEC	-470	Reading process not executed
ERR_PAPER_PILED	-1010	Paper pilling error
ERR_PAPER_JAM	-1020	Paper jam error
ERR_COVER_OPEN	-1030	Cover open error
ERR_MICR_NODATA	-1040	MICR data is not existing
ERR_MICR_BADDATA	-1050	MICR data is not able to recognize
ERR_MICR_NOISE	-1070	Noise error has occurred during MICR reading
ERR_SCN_COMPRESS	-1080	Scan image data compressing error
ERR_PAPER_EXIST	-1090	API can not be execute because there is a paper on the path
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_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_SCAN	-450	Failed to read the image data
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_NOT_EXEC	-470	Reading process not executed
ERR_PAPER_PILED	-1010	Paper pilling error
ERR_PAPER_JAM	-1020	Paper jam error
ERR_COVER_OPEN	-1030	Cover open error
ERR_MICR_NODATA	-1040	MICR data is not existing
ERR_MICR_BADDATA	-1050	MICR data is not able to recognize
ERR_MICR_NOISE	-1070	Noise error has occurred during MICR reading
ERR_SCN_COMPRESS	-1080	Scan image data compressing error
ERR_PAPER_EXIST	-1090	API can not be execute because there is a paper on the path
ERR_PAPER_INSERT	-1100	Failed to insert paper

MF_SCAN.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_SCAN	-450	Device failed in image scanning
ERR_NOT_EXEC	-470	Process not being executed
ERR_SCN_COMPRESS	-1080	Scan image data compressing error

MF_MICR01.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_BUFFER_OVER_FLOW	-140	Insufficient buffer error
ERR_NOT_FOUND	-220	Data not found
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_NOT_EXEC	-470	Reading process not executed
ERR_MICR_NODATA	-1040	MICR data is not existing
ERR_MICR_BADDATA	-1050	MICR data is not able to recognize
ERR_MICR_NOISE	-1070	Noise error has occurred during MICR reading

MF_BARCODE.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	Data not found
ERR_NOT_EXEC	-470	Process not being executed
ERR_BARCODE_NODATA	-1130	Barcode cannot be detected

MF_PRINT01.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_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_NOT_EXEC	-470	Reading process not executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- For SCAN/MICR/transaction printing, use a structure to specify the parameters of the functions you wish to use and call them using this API. MF_SET_BASE_PARAM must always be set. When a structure is set again, change the members of the structure and then call this API again by specifying MF_SET_xxxx_PARAM. The structures contain all the results output by the TM-S9000/S2000 API, up until the execution of BiCloseMon-Printer.
- By specifying MF_EXEC as the 3rd parameter, the specified functions are executed for all check sheet inserted in the feeder. When the feeder is empty, it stops automatically. Be sure not to discard the structure in order to set the return value to the structure. Be sure to invoke MF_CLEAR_xxxx_PARAM before discarding the structure.
- After the processing has been returned from the handler that performs MF_DATARECEIVE_DONE notification processing to the TM-S9000/S2000 API, electronic endorsement is executed.
- A scan is performed using the highest resolution level set in the MF_SCAN structure.
- When a scan is executed with this API, even if RGB and infrared has been specified as the scan light source using BiSCNSetImageTypeOption in advance, only an image scanned using the light source specified using BiSCNSelectScanImage can be captured.
- To capture both RGB images for card scan, either BiSCNMICRFunctionContinuously or BiSCNMICRFunctionPostPrint is used.
- A single sheet of Check sheet is read, and the read result is stored in the class set in the driver.



- Before invoking this API, execute of BiSCNMICRSetStatusBackFunction.
- When MF_CONTINUE, MF_MICR_RETRANS, MF_SCAN_FRONT_RETRANS, and MF_SCAN_BACK_RETRANS are specified in the 3rd parameter, ERR_NOT_SUPPORT is returned.

Processing status list

Main status	Sub status	Outline
MF_FUNCTION_START	-	The Check sheet read processing was started.
MF_CHECKPAPER_PROCESS_START	-	The Check sheet was inserted and the processing was started.
MF_DATARECEIVE_START	-	The data reception processing was started.
MF_DATARECEIVE_DONE	-	The data reception processing was completed.
MF_CHECKPAPER_PROCESS_DONE	-	The Check sheet was ejected and the processing was completed.
MF_FUNCTION_DONE	MF_BASE01.iRet	The check sheet read processing was completed.
MF_ERROR_OCCURED	ERR_PAPER_PILED	Double feed was detected. (Load results: bit5)
	ERR_FORM_LENGTH	A paper length error occurred. (Detailed load results: 44H, 45H)
	ERR_PAPER_JAM	A paper jam error occurred. (Detailed load results: 46H)
	ERR_MECHANICAL	A mechanical error occurred. (Detailed load results: 47H)
	ERR_COVER_OPEN	The processing was terminated because the cover was opened. (Detailed load results: 48H)
	ERR_MICR_NODATA	Magnetic waveform detection error (MICR details: 45H)
	ERR_MICR_BADDATA	Unanalyzable character detection error. (MICR details: 46H)
	ERR_MICR_NOISE	Noise error (MICR details: 47H)
	ERR_PRINT_- DATA_LENGTH_EXCEED	The printing content exceeds printable area.
	ERR_PRINT_DATA_UNRE- CEIVE	Print data unreceived error occurred.
	ERR_SCN_COMPRESS	Data compression error. (SCN details: 47H)
	ERR_SCN_IQA	NOT_PASS is detected at the IQA validation.
	ERR_BARCODE_NODATA	Barcode cannot be detected. (BARCODE details: 45H)

BiSCNMICRFunction

Acquire scanning an image and reading MICR characters.



- This API is compatible with the TM-J9000/TM-S1000 API.
- This API allows for the scanning of ID cards.

Syntax

```
int BiSCNMICRFunction(int nHandle, LPVOID lpvStruct, WORD wFunction)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpvStruct: The address of the parameter structure specified for each unit.
Refer to ["Structures" on page 277](#).

wFunction (Constant)	Parameter to Set for lpvStruct
MF_EXEC	"NULL"
MF_CONTINUE	"NULL"
MF_MICR_RETRANS	"NULL"
MF_SCAN_FRONT_RETRANS	"NULL"
MF_SCAN_BACK_RETRANS	"NULL"
MF_SET_BASE_PARAM	Address of the MF_BASE01 structure
MF_SET_MICR_PARAM	Address of the MF_MICR01 structure
MF_SET_SCAN_FRONT_PARAM	Address of the MF_SCAN structure for front of face
MF_SET_SCAN_BACK_PARAM	Address of the MF_SCAN structure for back of face
MF_SET_PRINT_PARAM	Address of the MF_PRINT01 structure
MF_SET_BARCODE_FRONT_PARAM	Address of the MF_BARCODE structure for front of face
MF_SET_BARCODE_BACK_PARAM	Address of the MF_BARCODE structure for back of face
MF_CLEAR_BASE_PARAM	Address of the MF_BASE01 structure
MF_CLEAR_MICR_PARAM	Address of the MF_MICR01 structure
MF_CLEAR_SCAN_FRONT_PARAM	Address of the MF_SCAN structure for front of face
MF_CLEAR_SCAN_BACK_PARAM	Address of the MF_SCAN structure for back of face
MF_CLEAR_PRINT_PARAM	Address of the MF_PRINT01 structure
MF_CLEAR_BARCODE_FRONT_PARAM	Address of the MF_BARCODE structure for front of face
MF_CLEAR_BARCODE_BACK_PARAM	Address of the MF_BARCODE structure for back of face
MF_GET_BASE_DEFAULT	Address of the MF_BASE01 structure
MF_GET_MICR_DEFAULT	Address of the MF_MICR01 structure
MF_GET_SCAN_FRONT_DEFAULT	Address of the MF_SCAN structure for front of face
MF_GET_SCAN_BACK_DEFAULT	Address of the MF_SCAN structure for back of face
MF_GET_PRINT_DEFAULT	Address of the MF_PRINT01 structure
MF_GET_BARCODE_FRONT_DEFAULT	Address of the MF_BARCODE structure for front of face
MF_GET_BARCODE_BACK_DEFAULT	Address of the MF_BARCODE structure for back of face

- wFunction: Specifies the functions for the API to execute. For settings regarding reading, by specifying MF_SET_MICR_PARAM and so on and executing this function, the API is notified. If the members of the structure are changed after notification, it is necessary to

perform notification again with this function. Include the header file submitted for the definition name used for this API.

Refer to ["Supplemental Description for wFunction" on page 108](#).

Supplemental Description for wFunction

wFunction (Constant)	Description
MF_EXEC	SCAN / MICR / Transaction printing is carried out according to the specified parameters. The second parameter is ignored.
MF_CONTINUE	If multiple sheets are sent, BiSCNMICRFunction stops scanning and waits for the application. If you would like to restart scanning, please call BiSCNMICRFunction with MF_CONTINUE. If you wouldn't like to restart scanning, please call BiSCNMICRCancelFunction. ERR_PARAM may be returned for double feed. The second parameter of BiSCNMICRFunction is ignored.
MF_MICR_RETRANS	Reread previously read MICR data. If MF_EXEC has not been used, ERR_PARAM is returned.
MF_SCAN_FRONT_RETRANS	Reread SCAN data for previously read front side. This makes the image corresponding to wImageID for front side structure currently set the target of reading. If MF_EXEC has not been used, ERR_PARAM is returned.
MF_SCAN_BACK_RETRANS	Reread SCAN data for previously read back side. This makes the image corresponding to wImageID for back side structure currently set the target of reading. If MF_EXEC has not been used, ERR_PARAM is returned.
MF_SET_BASE_PARAM	This sets base parameters. The MF_BASE01 structure address is specified in lpvStruct.
MF_SET_MICR_PARAM	This sets MICR parameters. The MF_MICR01 structure address is specified in lpvStruct.
MF_SET_SCAN_FRONT_PARAM	This sets front side read scanning parameters. The MF_SCAN structure address is specified in lpvStruct. The MF_SCAN structure has a different address from the structure for reading the back side. When the same address is specified, ERR_PARAM is returned. The operation is identical when MF_SET_SCAN_PARAM is specified.
MF_SET_SCAN_BACK_PARAM	This sets back side read scanning parameters. The MF_SCAN structure address is specified in lpvStruct. The MF_SCAN structure has a different address from the structure for reading the front side. When the same address is specified, ERR_PARAM is returned.
MF_SET_PRINT_PARAM	This sets transaction printing parameters. The MF_PRINT01 structure address is specified in lpvStruct.
MF_SET_BARCODE_FRONT_PARAM	This sets front side barcode decode parameters. The MF_BARCODE structure address is specified in lpvStruct.
MF_SET_BARCODE_BACK_PARAM	This sets back side barcode decode parameters. The MF_BARCODE structure address is specified in lpvStruct.

wFunction (Constant)	Description
MF_CLEAR_BASE_PARAM	This clears all the specifications for BASE/ MICR/ SCAN/ PRINT/ PROCESS/ BARCODE parameters. lpvStruct values are ignored.
MF_CLEAR_MICR_PARAM	This clears the MICR parameter specifications. lpvStruct values are ignored.
MF_CLEAR_SCAN_FRONT_PARAM	This clears the scan parameter specifications. lpvStruct values are ignored. The operation is identical when MF_CLEAR_SCAN_PARAM is specified.
MF_CLEAR_SCAN_BACK_PARAM	This clears the scan parameter specifications. lpvStruct values are ignored.
MF_CLEAR_PRINT_PARAM	This clears the transaction printing parameter specifications. lpvStruct values are ignored.
MF_CLEAR_BARCODE_FRONT_PARAM	This clears front side barcode decode parameters specifications. lpvStruct values are ignored.
MF_CLEAR_BARCODE_BACK_PARAM	This clears back side barcode decode parameters specifications. lpvStruct values are ignored.
MF_GET_BASE_DEFAULT	This obtains the initial values for the device base structure. (Refer to "Default Values of the MF_BASE01 Structure" on page 280.)
MF_GET_MICR_DEFAULT	This obtains the initial values for the device MICR structure. (Refer to "Default Values of the MF_MICR01 Structure" on page 289.)
MF_GET_SCAN_DEFAULT	This obtains the initial values for the device SCAN structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_SCAN_FRONT_DEFAULT	This obtains the initial values for the device SCAN (front side) structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_SCAN_BACK_DEFAULT	This obtains the initial values for the device SCAN (back side) structure. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_PRINT_DEFAULT	This obtains the initial values for the transaction printing structure. With API, the initial values of iSize and iVersion are not returned. The application must specify these two values. Also, the initial value for the structure must be obtained after zero clear of all member variables except iSize and iVersion. (Refer to "Default Values of the MF_PRINT01 Structure" on page 297.)
MF_GET_BARCODE_FRONT_DEFAULT	This obtains the initial values for the device Barcode (front side) structure. (Refer to "Default Values of the MF_BARCODE Structure" on page 359.)
MF_GET_BARCODE_BACK_DEFAULT	This obtains the initial values for the device Barcode (back side) structure. (Refer to "Default Values of the MF_BARCODE Structure" on page 359.)

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
ERR_OFFLINE	-110	Waiting to return from an offline state
ERR_PAPERINSERT_TIMEOUT	-300	Failed to insert paper
ERR_EXEC_FUNCTION	-310	Another API is running
ERR_RESET	-400	Device is being reset
ERR_THREAD	-420	Failed to the start of thread
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_SCAN	-450	Failed to read the image data
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_PAPER_JAM	-1020	Paper jam error
ERR_PAPER_EXIST	-1090	API can not be execute because there is a paper on the path
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_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_SCAN	-450	Failed to read the image 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_SCAN.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_SCAN	-450	Device failed in image scanning
ERR_NOT_EXEC	-470	Process not being executed

MF_MICR01.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_BUFFER_OVER_FLOW	-140	Insufficient buffer error
ERR_NOT_FOUND	-220	Data not found
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_NOT_EXEC	-470	Reading process not executed

MF_BARCODE.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	Data not found
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_NOT_EXEC	-470	Process not being executed
ERR_BARCODE_NODATA	-1130	Barcode cannot be detected

MF_PRINT01.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_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_NOT_EXEC	-470	Reading process not executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- A single sheet of check sheet is read, and the read result is stored in the class set in the driver.
- Specify the structure for the parameters of the function you wish to use among SCAN, MICR, and Transaction printing to call BiSCNMICRFunction.
- Be sure to set MF_SET_BASE_PARAM.
- To set the parameter structure again, change the members of the parameter structure and specify MF_SET_xxxx_PARAM again to call BiSCNMICRFunction.
- The entire information on the structure is stored with TM-S9000/S2000 API until BiCloseMonPrinter is executed.
- Specifying MF_EXEC for the third parameter executes the specified function. BiSCNMICRFunction performs in order of SCAN, MICR, and Transaction.
- As you are to set the return value to the parameter structure, do not scrap the structure. If you still scrap the structure, do so after calling MF_CLEAR_xxxx_PARAM.
- Returns ERR_PARAM (Not ERR_NOT_SUPPORT) if any unsupported functions are specified.
- A scan is performed using the highest resolution level set in the MF_SCAN structure.
- When a scan is executed with this API, even if RGB and infrared has been specified as the scan light source using BiSCNSetImageTypeOption in advance, only an image scanned using the light source specified using BiSCNSelectScanImage can be captured.
- To capture both RGB images for card scan, either BiSCNMICRFunctionContinuously or BiSCNMICRFunctionPostPrint is used.
- When the use of OCR is set to the MICR settings (the MF_MICR01 structure's bMicOcrSelect MF_MICR_USE_OCR bit is set to ON), ERR_PARAM is returned while executing MF_EXEC execution when the scan parameter of the front surface reading is not set. This is because the scanned image of the front surface is required for OCR processing.

- When a valid electronic endorsement process is specified by the settings of the transaction printing (MF_PRINT_TYPE_ENDORSE_NORMAL or MF_PRINT_TYPE_ELECTRIC_ENDORSE_ONLY is specified in dwEndorseType of MF_PRINT01 structure), ERR_PARAM is returned while executing MF_EXEC if the scan parameter of the back surface reading is not set. This is because a scanned image of the back surface is required for the electronic endorsement process. When a valid electronic endorsement process is specified (MF_PRINT_TYPE_ENDORSE_NORMAL is specified in dwEndorseType of MF_PRINT01 structure) only after the transaction printing is successful, no data is set to the bStatus, bDetail, dwXSize, dwYSize, dwScanSize, lpbScanData parameters of the back surface scan till the transaction printing process is complete, even if scanning of the back surface is successful.
- Caution is required for such MF_xxx structure's member variables that store the address values. Executing this function with a nonexistent address value specified may cause TRAP. Make sure to specify Null when an address setting is not necessary.
- Caution is required for MF_xxx structure's member variables that store character strings. Executing this function without a '\0' that indicates the end of the character string in the variable may cause TRAP. '\0' must exist in the variable.
- The double delivery detection notification does not operate when this function's MF_EXEC is executed with MF_BASE_MESSAGE_NO_MESSAGE set to the dwNotifyType member variable of the MF_BASE01 structure.

BiSCNMICRCancelFunction

BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint is interrupted.

Syntax

int **BiSCNMICRCancelFunction**(int nHandle, WORD wEjectType)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

wEjectType: This parameter is ignored.

wEjectType (Constant)	Parameter to Set for lpvStruct
MF_EJECT_DISCHARGE	Ejects the paper into a packet.
MF_EJECT_RELEASE	Specifies that the paper should be unclamped and left in the same location.

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	Waiting to return from an offline state
ERR_WITHOUT_CB	-130	Cannot be executed as neither of BiSCNMICRSetStatusBackFunction is invoked
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint is interrupted. Alternatively, paper-eject is executed after the ERR_LINE_OVERFLOW error occurs with the BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint.
- When interrupting the BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint, if the interruption is not complete after a maximum of 20 seconds has elapsed, ERR_TIMEOUT is returned. Once interrupted, the results are set in the storage area for the structure specified by BiSCNMICRFunctionContinuously/BiSCNMICRFunctionPostPrint.
- The only times that ERR_EXEC_FUNCTION is returned by this API is when there are overlapping calls of this API with multiple threads and when the device is initialized after print setup and turning the device power off and on.



Paper is ejected into the specified pocket according to the scan settings set for each structure.

BiSCNDeleteCroppingArea

Deletes a registered cropping area.

Syntax

```
int BiSCNDeleteCroppingArea(int nHandle, BYTE AreaNo)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
AreaNo: Specifies the number (0 to 255) of the cropping area to be deleted.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

When 0 is specified for bAreaNo, the entire cropping area is deleted.

BiSCNSelectScanFace

Specifies the back or front for an image that is being read.



Scan-type APIs execute scans using the light source set with this API.

Syntax

```
int BiSCNSelectScanFace(int nHandle, BYTE bFace)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
bFace: This specifies the back or front for an image that is being read.

Constant	Value	Description
MF_SCAN_FACE_FRONT	0	Specified for the front.
MF_SCAN_FACE_BACK	1	Specified for the back.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiGetPrnCapability

Obtains the device information specified by the device ID.

Syntax

```
int BiGetPrnCapability (int nHandle, BYTE prnID,  
                        LPBYTE pBuffSize, LPBYTE pBuff )
```

Argument

nHandle:	Handle obtained with the return value of the communication initialization API.
prnID:	Specifies the device ID from which obtain information. To check if UV light source is utilized, specify 112 that means Type ID (B). See Device Information for details on " Device ID " on page 381.
pBuffSize:	Specifies the size of the memory to which the device information is stored. Values of 1 - 80 can be specified. (Recommended value: 80) Returns the actual read data size after calling this function. In the case of insufficient buffer capacity, the required byte size is returned.
pBuff:	Specifies the address of the memory to which the device information is stored.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_OFFLINE	-110	Cannot be used because the device is waiting for online reset.
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to "[Return value](#)" on page 61.

Description

Specifying an unsupported device ID and issuing this command results in either of the following. The result depends on the product and firmware.

- Time error occurs (Return value: ERR_TIMEOUT).
- Returns "Success" (Return value: SUCCESS), and sets a value to the parameter pBuff. However, Epson does not guarantee the obtained pBuff parameter value.

See "[Device ID](#)" on page 381 for details on Device ID.

BiSCNGetImageFormat

Acquires the format of the image.

Syntax

```
int BiSCNGetImageFormat(int nHandle, LPBYTE pFormat)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pFormat: Specifies the memory address where the format of the notified image data is set.
The set value is as shown below.

Constant	Value	Description
EPS_BI_SCN_TIFF	1	TIFF format CCITT (Group 4) compressed data
EPS_BI_SCN_RASTER	2	Raster format uncompressed data
EPS_BI_SCN_BITMAP	3	Bitmap format uncompressed data
EPS_BI_SCN_TIFF256	4	TIFF format uncompressed data
EPS_BI_SCN_JPEGHIGH	5	JPEG format high compression (size priority) data
EPS_BI_SCN_JPEGNORMAL	6	JPEG format normal compression data
EPS_BI_SCN_JPEGLOW	7	JPEG format low compression (quality priority) data
EPS_BI_SCN_JTIFF	8	TIFF format JPEG compressed data

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSCNSetImageFormat

Selects the format of the scanning image data.



The selected format is enabled until BiCloseMonPrinter is executed.

Syntax

```
int BiSCNSetImageFormat(int nHandle, BYTE bFormat)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
bFormat: Specifies the format of image data notified. The valid specification values are as shown below. The default value is EPS_BI_SCN_TIFF(1).

Constant	Value	Description
EPS_BI_SCN_TIFF	1	TIFF format CCITT (Group 4) compressed data
EPS_BI_SCN_RASTER	2	Raster format uncompressed data
EPS_BI_SCN_BITMAP	3	Bitmap format uncompressed data
EPS_BI_SCN_TIFF256	4	TIFF format uncompressed data
EPS_BI_SCN_JPEGHIGH	5	JPEG format high compression (size priority) data
EPS_BI_SCN_JPEGNORMAL	6	JPEG format normal compression data
EPS_BI_SCN_JPEGLow	7	JPEG format low compression (quality priority) data
EPS_BI_SCN_JTIFF	8	TIFF format JPEG compressed data



To save as an image file of ANSI X9.100-181-2007 standard, it must be a TIFF format with CCITT(Group 4) compression data (bFormat = EPS_BI_SCN_TIFF), black and white (bColorDepth of BiSCNSetImageQuality = EPS_BI_SCN_1BIT), and whose resolution is 200 dpi (sResolution of MF_SCAN = MF_SCAN_DPI_200).

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- This function is valid for the scanner unit currently selected.
- In the case of Executing BiSCNMICRFunction, The set reading quality is applied beginning with the next image data read. (Reading quality is not applied to image data that has already been read.)
- When using BiSCNMICRFunctionPostPrint or BiSCNMICRFunctionContinuously, the image data format setting made by this API can be applied even after acquiring an image using BiGetScanImage in MF_DATA_RECEIVE_DONE callback. To do that, call BiGetScanImage again after setting the image data format using this API.

BiSCNGetScanArea

The read area of the device scanner image is acquired.

Syntax

```
int BiSCNGetScanArea
    (int nHandle, LPBYTE pStartX,
     LPBYTE pStartY, LPBYTE pEndX, LPBYTE pEndY)
```

Argument

- nHandle:
- Handle obtained with the return value of the communication initialization API.
- pStartX:
- Returns the start X coordinate of the read area.
- pStartY:
- Returns the start Y coordinate of the read area.
- pEndX:
- Returns the end X coordinate of the read area.
- pEndY:
- Returns the end Y coordinate of the read area.

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	Unsupported



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The origin of the surface coordinates (0,0) corresponds the bottom-right corner of the check sheet, when oriented as if to be inserted. The origin of the rear face coordinates (0,0) corresponds the bottom-left corner of the check sheet, when oriented as if to be inserted.

BiSCNSetScanArea

Configures the reading area of the image data.

Syntax

```
int BiSCNSetScanArea
    (int nHandle, BYTE bStartX,
     BYTE bStartY, BYTE bEndX, BYTE bEndY)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- bStartX: Specifies the start X coordinate (0 to 108) of the read area in units of mm.
The initial value is 0.
- bStartY: Specifies the start Y coordinate (0 to 252) of the read area in units of mm.
The initial value is 0.
- bEndX: Specifies the end X coordinate (0 to 110, and larger than bStartX.) of the read area in units of mm. The initial value is 0. When 0 is specified, the maximum value supported by the device is specified.
- bEndY: Specifies the end Y coordinate (0 to 254, and larger than bStartY.) of the read area in units of mm. The initial value is 0. When 0 is specified, the internal measured value of the device is specified.

Return value

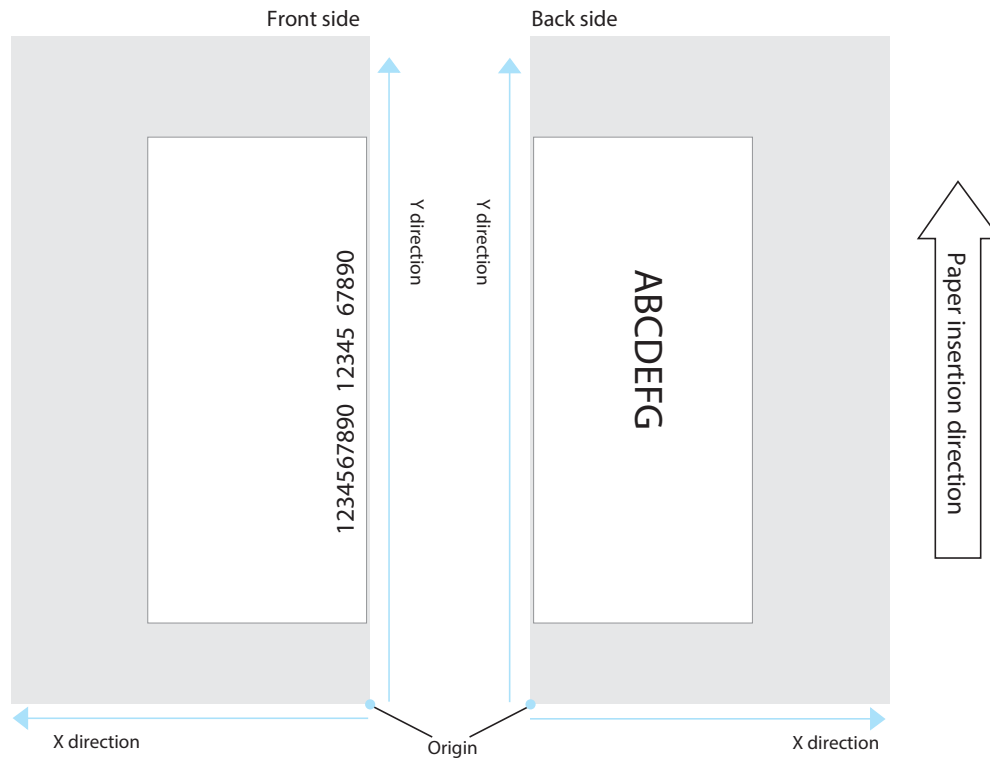
Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The origin of the front face coordinates (0,0) corresponds the bottom-right corner of the check sheet, when oriented as if to be inserted. The origin of the rear face coordinates (0,0) corresponds the bottom-left corner of the check sheet, when oriented as if to be inserted. The set value remains valid until the device is closed.



Note

- When the start point is beyond the end point (Start \geq End), ERR_PARAM is returned to the return value.
- When a value exceeding the area that the device can read is specified, the maximum value within the area that the device can read is set.
- The read area set by this API is applied from the next read processing.
- If an odd number is specified, it is rounded to the nearest even number. The start point (bStartX, bStartY) is rounded down, and the end point (bEndX, bEndY) is rounded up to the nearest even number.
- This API does not support card scan processing. When card scan processing is performed, ERR_NOT_SUPPORT is returned to the return value.

BiSCNGetImageQuality

Sets the image scanning quality.

Syntax

```
int BiSCNGetImageQuality
    (int nHandle, LPBYTE lpbColorDepth, char* pcThreshold
    , LPBYTE lpbColor, LPBYTE lpbExOption)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

LPBYTE lpbColorDepth:

Specifies the memory address where the tonal gradation is set. The set value is as shown below.

Constant	Value	Description
EPS_BI_SCN_1BIT	1	Black and White (1 bit)
EPS_BI_SCN_8BIT	8	256 grayscale (8 bit)
EPS_BI_SCN_24BIT	24	Color (24 bit)

pcThreshold: Specifies the memory address where the brightness threshold value is set.

lpbColor: Specifies the memory address where the color is set. The set value is as shown below.

Constant	Value	Description
EPS_BI_SCN_MONOCHROME	1	Monochrome
EPS_BI_SCN_COLOR	8	Color

lpbExOption: Specifies the memory address where the density adjustment types is set. The set value is as shown below.

Constant	Value	Description
EPS_BI_SCN_MANUAL	49	Applies the value of bThreshold for Black and White.
EPS_BI_SCN_SHARP	50	Sharpening
EPS_BI_SCN_SHARP_CUSTOM	51	These constants are compatible with the TM-S1000 API, and as such, do not function with the TM-S9000II, the TM-S2000II, the TM-S9000, and the TM-S2000.
EPS_BI_SCN_SHARP_CUSTOM2	52	
EPS_BI_SCN_SHARP_CUSTOM3	53	

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- Refer to the following for details on this API

bColorDepth	bExOption	Scan result
EPS_BI_SCN_1BIT	EPS_BI_SCN_MANUAL	A value set to bThreshold is applied
	EPS_BI_SCN_SHARP	bThreshold is automatically set and the set value is applied
EPS_BI_SCN_8BIT	EPS_BI_SCN_MANUAL	No special handling is applied.
	EPS_BI_SCN_SHARP	Sharpening
EPS_BI_SCN_24BIT	EPS_BI_SCN_MANUAL	No special handling is applied.
	EPS_BI_SCN_SHARP	Sharpening

- This function is valid for the scanner unit currently selected.
- If EPS_BI_SCN_8BIT or EPS_BI_SCN_24BIT is set for bColorDepth, the density threshold specified with bThreshold will not be applied to the image load results.
- If black-and-white mode is specified for scanned images with BiSCNSetImageTypeOption, black-and-white images will be retrieved even if EPS_BI_SCN_COLOR is specified for bColor.
- If the card unit is selected, when acquiring an image, EPS_BI_SCN_8BIT is acquired, even if EPS_BI_SCN_1BIT is selected.
- In the case of Executing BiSCNMICRFunction, The set reading quality is applied beginning with the next image data read. (Reading quality is not applied to image data that has already been read.)

BiSCNSetImageQuality

Sets the image scanning quality.

Syntax

int **BiSCNSetImageQuality**

(int nHandle, BYTE bColorDepth, char bThreshold
, BYTE bColor, BYTE bExOption)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bColorDepth: Specifies the tonal gradation (the number of bits used for 1 pixel).

Constant	Value	Description
EPS_BI_SCN_1BIT	1	Black and White (1 bit)
EPS_BI_SCN_8BIT (Default)	8	256 grayscale (8 bit)
EPS_BI_SCN_24BIT	24	Color (24 bit)

bThreshold: Brightness threshold value (-128 ~ 127) for Black and White. Used when bColorDepth=EPS_BI_SCN_1BIT, and bExOption=EPS_BI_SCN_MANUAL. The value -128 to 127 corresponds to 0 to 255 of the brightness. When “0” is specified, the intermediate brightness “128” is applied.

bColor: Specifies the color. Valid specification values are as shown below.
If bColorDepth is set to EPS_BI_SCN_1BIT/EPS_BI_SCN_8BIT, specify EPS_BI_SCN_MONOCHROME. If bColorDepth is set to EPS_BI_SCN_24BIT, specify EPS_BI_SCN_COLOR.

Constant	Value	Description
EPS_BI_SCN_MONOCHROME	1	Monochrome
EPS_BI_SCN_COLOR	8	Color

bExOption: Specifies density adjustment types. Valid specification values are as shown below.

Constant	Value	Description
EPS_BI_SCN_MANUAL	49	Applies the value of bThreshold for Black and White.
EPS_BI_SCN_SHARP	50	Sharpening
EPS_BI_SCN_SHARP_CUSTOM	51	These constants are compatible with the TM-S1000 API, and as such, do not function with the TM-S9000II, the TM-S2000II, the TM-S9000, and the TM-S2000. Use EPS_BI_SCN_MANUAL or EPS_BI_SCN_SHARP.
EPS_BI_SCN_SHARP_CUSTOM2	52	
EPS_BI_SCN_SHARP_CUSTOM3	53	

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- Refer to the following for details on this API

bColorDepth	bColor	bExOption	Scan result
EPS_BI_SCN_1BIT	EPS_BI_SCN_MONOCHROME	EPS_BI_SCN_MANUAL	A value set to bThreshold is applied
		EPS_BI_SCN_SHARP	bThreshold is automatically set and the set value is applied
EPS_BI_SCN_8BIT	EPS_BI_SCN_MONOCHROME	EPS_BI_SCN_MANUAL	No special handling is applied.
		EPS_BI_SCN_SHARP	Sharpening
EPS_BI_SCN_24BIT	EPS_BI_SCN_COLOR	EPS_BI_SCN_MANUAL	No special handling is applied.
		EPS_BI_SCN_SHARP	Sharpening

- This function is valid for the scanner unit currently selected.
- If EPS_BI_SCN_8BIT or EPS_BI_SCN_24BIT is set for bColorDepth, the density threshold specified with bThreshold will not be applied to the image load results.
- If black-and-white mode is specified for scanned images with BiSCNSetImageTypeOption, black-and-white images will be retrieved even if EPS_BI_SCN_COLOR is specified for bColor.
- If the card unit is selected, when acquiring an image, EPS_BI_SCN_8BIT is acquired, even if EPS_BI_SCN_1BIT is selected.
- In the case of Executing BiSCNMICRFunction, The set reading quality is applied beginning with the next image data read. (Reading quality is not applied to image data that has already been read.)

BiSCNSelectScanImage

Selects the light source of the scanned image to be retrieved.

The result of a scan executed using the light source specified with this API is returned.



The selected setting is retained as the information on the scan face set with BiSCNSelectScanFace. If BiSCNSelectScanFace has been called first, the scan face setting specified with that API is updated.

Syntax

```
int BiSCNSelectScanImage(int nHandle, unsigned char ucImage)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

ucImage: Type of image to be retrieved. The following values can be specified independently:

Constant	Value	Description
MF_SCAN_IMAGE_VISIBLE	0	RGB (visible light) image
MF_SCAN_IMAGE_INFRARED	1	Infrared image
MF_SCAN_IMAGE_2ND_LIGHTSOURCE	1	Infrared or ultraviolet image
MF_SCAN_IMAGE_MERGED	2	Visible image and ultraviolet image are merged
MF_SCAN_IMAGE_MERGED2	3	Visible image and inverse ultraviolet image are merged

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSCNGetCroppingArea

Acquires cropping area information from the device.



This API is compatible with the TM-J9000 API.

Syntax


int **BiSCNGetCroppingArea** (int nHandle, LPWORD pBuffSize, LPBYTE pBuff)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pBuffSize: Specifies the size of the memory in which the cropping area information is set. After the BiSCNGetCroppingArea designation has been issued, the size of the actually read data is returned. When the memory size specified in pBuffSize is insufficient, the required memory size is returned. Nothing is returned in the event of any other error.
- pBuff: Specifies the memory address that acquires the cropping area information.

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	Unsupported
ERR_BUFFER_OVER_FLOW	-140	Buffer overflow error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

pBuff acquires the cropping area information in the format as shown below.

Cropping area number

Start X coordinate of the cropping area

Start Y coordinate of the cropping area

End X coordinate of the cropping area

End Y coordinate of the cropping area

BiSCNSetCroppingArea

Sets the cropping area.

Syntax

```
int BiSCNSetCroppingArea(int nHandle, BYTE bAreaNo, BYTE bStartX
, BYTE bStartY, BYTE bEndX, BYTE bEndY)
```

Argument

nHandle:	Handle obtained with the return value of the communication initialization API.
bAreaNo:	Specifies the cropping area number (1 to 255).
bStartX:	Specifies the start X coordinate (0 to 108) of the cropping area in units of mm.
bStartY:	Specifies the start Y coordinate (0 to 254) of the cropping area in units of mm.
bEndX:	Specifies the end X coordinate (2 to 110, and larger than bStartX) of the cropping area in units of mm. When a value that exceeds the area that can be cropped, the maximum value of the cropping area is set.
bEndY:	Specifies the end Y coordinate (2 to 254, and larger than bStartY) of the read area in units of mm. When a value that exceeds the area that can be cropped, the maximum value of the cropping area is set.

Return value

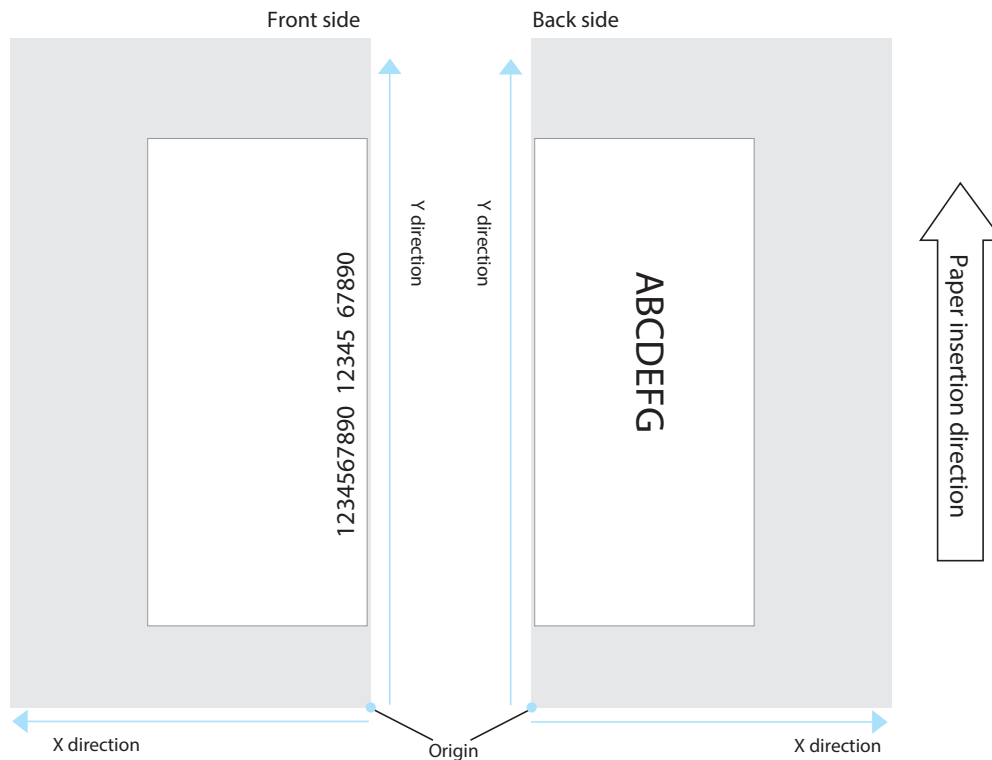
Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The origin of the surface coordinates (0,0) corresponds the bottom-right corner of the check sheet, when oriented as if to be inserted. The origin of the rear face coordinates (0,0) corresponds the bottom-left corner of the check sheet, when oriented as if to be inserted. The set value remains valid until the device is closed.



- When the start point is beyond the end point (Start \geq End), ERR_PARAM is returned to the return value.
- When the specified Cropping area number already exists, set a new Cropping area.
- When a value that exceeds the area that can be cropped for the device, the maximum value of the area that the device can crop is set.
- The read area set by this API is applied from the next read processing.
- If an odd number is specified, it is rounded to the nearest even number. The start point (bStartX, bStartY) is rounded down, and the end point (bEndX, bEndY) is rounded up to the nearest even number.

BiSCNGetImageAdjustment

Acquires the brightness adjustment value, contrast adjustment value, and gamma correction value that are necessary for image correction.

Syntax

```
int BiSCNGetImageAdjustment
    (int nHandle, int* pnBrightness,
     int* pnContrast, unsigned long* pulGamma)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

pnBrightness: Specifies the memory address where the brightness adjustment value is set. The set value is as shown below.

Value	Description
Integer in the range from -100 to 100	-
EPS_BI_SCN_BRIGHTNESS_DEFAULT	Brightness adjustment default value. The setting is the same as the setting when no brightness adjustment is performed.
EPS_BI_SCN_BRIGHTNESS_NONE	Does not perform brightness adjustment.

pnContrast: Specifies the memory address where the contrast adjustment value is set. The set value is as shown below.

Value	Description
Integer in the range from -100 to 100	-
EPS_BI_SCN_CONTRAST_DEFAULT	Contrast adjustment default value. The setting is the same as the setting when no contrast adjustment is performed.
EPS_BI_SCN_CONTRAST_NONE	Does not perform contrast adjustment.

pulGamma: Specifies the memory address where the gamma correction value is set. The set value is as shown below.

Value	Description
EPS_BI_SCN_GAMMA_NONE	Does not perform gamma correction.
EPS_BI_SCN_GAMMA_DEFAULT	Gamma correction default value. The setting is the same as the EPS_BI_SCN_GAMMA_NONE.
EPS_BI_SCN_GAMMA_10	Does not perform gamma correction.
EPS_BI_SCN_GAMMA_18	Sets the gamma correction value to 1.8.
EPS_BI_SCN_GAMMA_22	Sets the gamma correction value to 2.2.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSCNSetImageAdjustment

Sets the brightness adjustment value, contrast adjustment value, and gamma correction value that are needed for image correction.



This API can be called inside the MF_DATARECEIVE_DONE callback that is issued when a scan is activated.

Syntax

```
int BiSCNSetImageAdjustment(int nHandle, int nBrightness
                             , int nContrast, unsigned long ulGamma)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

nBrightness: Specifies the brightness adjustment value. When 0 is set for brightness, brightness will not be applied.

Value	Description
Integer in the range from -100 to 100	If a value other than 0 is specified, brightness is applied at that value. If 0 is specified, brightness is not applied.
EPS_BI_SCN_BRIGHTNESS_DEFAULT	Brightness adjustment default value. The setting is the same as the setting when no brightness adjustment is performed.
EPS_BI_SCN_BRIGHTNESS_NONE	Does not perform brightness adjustment.

nContrast: Specifies the contrast adjustment value. When 0 is set for contrast, contrast will not be applied.

Value	Description
Integer in the range from -100 to 100	If a value other than 0 is specified, contrast is applied at that value. If 0 is specified, contrast is not applied.
EPS_BI_SCN_CONTRAST_DEFAULT	Contrast adjustment default value. The setting is the same as the setting when no contrast adjustment is performed.
EPS_BI_SCN_CONTRAST_NONE	Does not perform contrast adjustment.

ulGamma: Specifies the Gamma correction value. When 0 is set for gamma, gamma will not be applied.

Value	Description
EPS_BI_SCN_GAMMA_NONE	Does not perform gamma correction.
EPS_BI_SCN_GAMMA_DEFAULT	Gamma correction default value. The setting is the same as the EPS_BI_SCN_GAMMA_NONE.
EPS_BI_SCN_GAMMA_10	Does not perform gamma correction.
EPS_BI_SCN_GAMMA_18	Sets the gamma correction value to 1.8.
EPS_BI_SCN_GAMMA_22	Sets the gamma correction value to 2.2.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiMICRClearSpaces

Clears spaces included in MICR data obtained using ["BiGetMicrText" on page 157](#).

Syntax

```
int BiMICRClearSpaces(int nHandle, BYTE bClearSpace)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
bClearSpace: Specifies clearance of spaces in MICR data. Specify the following constants.

Constant	Description
CLEAR_SPACE_DISABLE (Default)	Does not clear spaces
CLEAR_SPACE_ENABLE	Clears spaces

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	A scan is being executed.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- The MICR data to be cleared is the following members in ["MF_MICR01" on page 286](#) structure.

Member	Description
szMicrStr	MICR character string
stOcrReliableInfo	Information on reliability of MICR character recognition

- The above settings are valid from the point when BiGetMicrText is invoked after invoking this API until BiCloseMonPrinter is invoked.

BiSetOcrABAreaOrigin

Specifies the origin of the OCR area for the MF_OCR_AB structure.

Syntax

int **BiSetOcrABAreaOrigin** (int nHandle, BYTE bOrigin)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
bOrigin: Specifies the origin of the OCR area. Specify the following values. The default value is OCR_ORIGIN_TOP_LEFT.

Constant	Description
OCR_ORIGIN_TOP_LEFT	Sets the origin to the top left corner in relation to the document insertion direction.
OCR_ORIGIN_BOTTOM_LEFT	Sets the origin to the bottom left corner in relation to the document insertion point.
OCR_ORIGIN_TOP_RIGHT	Sets the origin to the top right corner in relation to the document insertion point.
OCR_ORIGIN_BOTTOM_RIGHT	Sets the origin to the bottom right corner in relation to the document insertion point.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed

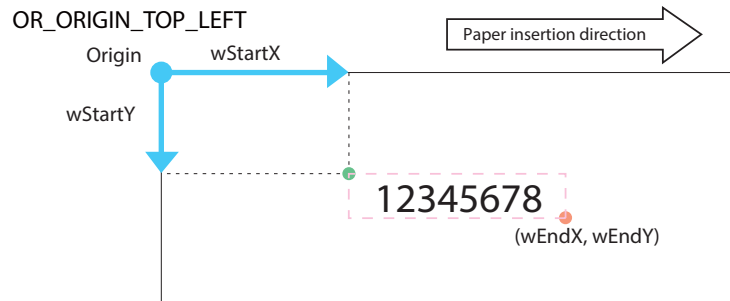


For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

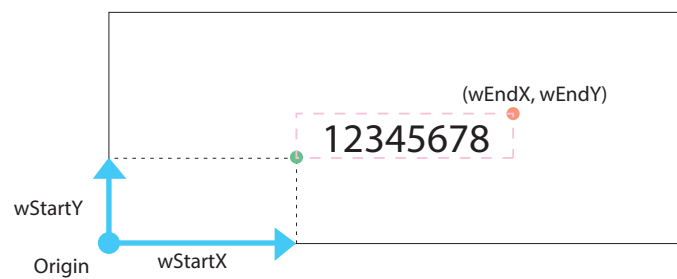
Description

The original specified using this API is valid until BiCloseMonPrinter is invoked.

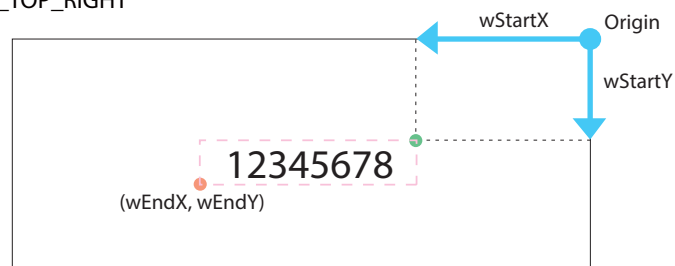
See the following illustration for defining ranges with different origins.



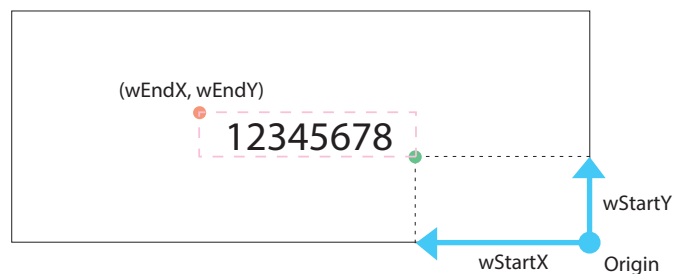
OCR_ORIGIN_BOTTOM_LEFT



OCR_ORIGIN_TOP_RIGHT



OCR_ORIGIN_BOTTOM_RIGHT



BiGetMicrText

Gets MICR text or OCR text.

Syntax

```
int BiGetMicrText(int nHandle, DWORD dwTransactionNumber
, LPMF_MICR ptMicr)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwTransactionNumber: Specifies the transaction number (ID) for the MICR text acquired. This is a DWORD type.

ptMicr: A buffer to store the MICR string.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_BUFFER_OVERFLOW	-140	Buffer overflow error
ERR_NOT_FOUND	-220	No data error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset
ERR_NOT_EXEC	-470	Process not being executed
ERR_MICR_NODATA	-1040	MICR data is not existing
ERR_MICR_BADDATA	-1050	MICR data is not able to recognize



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Gets the MICR text read with BiSCNMICRFunctionContinuously. After the reading status MF_DATARECEIVE_DONE notification, the reading results are saved in the various parameters of the MF_MICR01 structure by specifying the relevant transaction number (ID) in dwTransactionNumber.

When getting the MICR reading result, specify MF_MICR_USE_MICR in bMicOcrSelect of the MF_MICR01 structure, when getting the OCR reading result, specify MF_MICR_USE_OCR, and when getting the logical multiplication of the MICR and OCR reading results, specify MF_MICR_USE_MICR | MF_MICR_USE_OCR. The MICR/OCR reading result is set to szMicrStr of MF_MICR01 structure.

Information on reliability of the read characters is set to stOcrReliableInfo of MF_MICR01 structure.

Specifying MF_MICR_USE_MICR | MF_MICR_USE_OCR for the bMicOcrSelect makes a comparison between the reading result of MICR and that of OCR, and if any difference is found, “?” will be returned.

Note

- The interval during which the MICR reading result and OCR reading result corresponding to the transaction number (ID) can be acquired is from MF_DATARECEIVE_DONE notification to MF_CHECKPAPER_PROCESS_DONE notification.
When the process is returned from the MF_CHECKPAPER_PROCESS_DONE notification handler to the API, the MICR reading result/OCR reading result saved by the API is discarded.
- MICR magnetic waveform data is sent from the device when starting the reading. The position information is acquired from the magnetic waveform data. Therefore, if the position information could not be acquired from the magnetic waveform data when MF_MICR_USE_OCR has been specified to bMicOcrSelect, the OCR recognition cannot be made.
- When the font to read MICR (MF_MICR01.bFont) is CMC7, the OCR recognition result (MF_OCR_AB.stOcrReliableInfo) cannot be obtained. The relation between bMicOcrSelect and bFont in MF_MICR01 structure for OCR reading process is described below.

bFont	bMicOcrSelect		
	MF_MICR_USE_MICR	MF_MICR_USE_OCR	MF_MICR_USE_MICR MF_MICR_USE_OCR
MF_MICR_FONT_E13B	MICR magnetic waveform + OCR recognition *		
MF_MICR_FONT_CMC7	MICR magnetic waveform	Error (ERR_MICR_NODATA)	

* For the TM-S1000, specifying MF_MICR_USE_OCR returns OCR recognition data.

- Parsing can be used when the font to read MICR (MF_MICR01.bFont) is E13B. When CMC7 is used, after calling BiGetMicrText, ERR_MICR_PARSE is returned as a return value.

BiGetOcrABText

Performs the OCR recognition for the OCR-A font or the OCR-B font and acquires the result.



This API only supports the images that is created by the driver. Do not pass the edited image.

Syntax

```
int BiGetOcrABText (int nHandle, DWORD dwTransactionNumber
, BYTE bImageSource, LPCSTR szFileName
, LPMF_OCR_AB ptOcrAB)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwTransactionNumber:

Specifies the transaction number (ID) for the scan image acquired.

bImageSource:

Specify an image targeted for the OCR recognition. One of the following values can be specified.

Constant	Description
OCR_SOURCE_TRANSACTION_NUMBER	For an image file stored in the driver.
OCR_SOURCE_IMAGE_FILE	For an image file saved by the driver.

szFileName: Specify an image file name targeted for the OCR recognition.

ptOcrAB: Specify the memory address of the MF_OCR_AB structure. For the MF_OCR_AB structure, refer to "[MF_OCR_AB](#)" on page 290.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	No data error
ERR_IMAGE_FILEOPEN	-230	Open failure
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset
ERR_MICR_NODATA	-1040	data is not existing



For a list of return values and troubleshooting actions, refer to "[Return value](#)" on page 61.

Description

Performs the OCR recognition for the OCR-A font or the OCR-B font and acquires the result. Necessary conditions for the OCR recognition other than the targeted images are set to the MF_OCR_AB structure. The OCR recognition results are stored in the OUT attribute parameter of the MF_OCR_AB structure.

- When OCR_SOURCE_TRANSACTION_NUMBER is specified to bImageSource, image that are read immediately before and stored in the driver can be targeted for the OCR recognition. In this case, szFileName is ignored. After MF_DATARECEIVE_DONE is notified, the recognition result is stored in the OUT attribute parameter of the MF_OCR_AB structure by specifying the pertinent transaction number to dwTransactionNumber.
- When OCR_SOURCE_IMAGE_FILE is specified to bImageSource, the image files saved by the driver are targeted for OCR recognition. In this case, dwTransactionNumber is ignored.

Whenever image files exist it is OK to execute this API. The image files must meet the following conditions.

- Saved when either EPS_BI_SCN_BITMAP or EPS_BI_SCN_TIFF256 is specified with BiSCNSetImageFormat.
- Saved when EPS_BI_SCN_8BIT is specified to bColorDept parameter with bBiSCNSetImageQuality.
- Saved when EPS_BI_SCN_MANUAL is specified to bExOption parameter with BiSCNSetImageQuality.

Note

When OCR_SOURCE_TRANSACTION_NUMBER is specified to bImageSource, the OCR recognition results corresponding to the transaction number (ID) for the period between MF_DATARECEIVE_DONE notification to MF_CHECKPAPER_PROCESS_DONE notification can be acquired.

The stored OCR recognition results will be discarded when the process is returned to the TM-S9000/S2000 API from the MF_CHECKPAPER_PROCESS_DONE notification handler.

BiGetScanImage

Gets the scan image.

Syntax

```
int BiGetScanImage(int nHandle, DWORD dwTransactionNumber
, LPMF_SCAN ptScan)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwTransactionNumber: Specifies the transaction number (ID) for the scan image acquired. This is a DWORD type.

ptScan: A buffer to store the scan results.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	No data error
ERR_SCAN	-450	Device failed in image scanning
ERR_NOT_EXEC	-470	Process not being executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- Gets the scan image scanned with BiSCNMICRFunctionContinuously / BiSCNMICRFunctionPostPrint. After the scanning status MF_DATARECEIVE_DONE notification, the scanning results are saved in the various parameters of the MF_SCAN structure by specifying the relevant transaction number (ID) in dwTransactionNumber. When configuring a valid value for sResolution, bAddInfoDataSize and pAddInfoData that are part of MF_SCAN structure, the image which these values refer to will be stored.
- To get the front side scanning results, specify MF_SCAN_FACE_FRONT with BiSCNSelectScanFace, then execute this API. To get the back side scanning results, specify MF_SCAN_FACE_BACK with BiSCNSelectScanFace, then execute this API.
- When acquiring both the front and back of a scan image using this API, set the scan image settings (BiSCNSetImageFormat/BiSCNSetImageQuality/BiSCNSelectScanImage/BiSCNSetImageAdjustment) and then get the image.
- Based on the combination of settings, some scan data may not be acquired. Refer to the following.

BiSCNSetImageTypeOption	BiSCNSelectScanUnit	BiSCNSetImageQuality	Support
EPS_BI_SCN_OPTION_COLOR	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	✓
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	✓
EPS_BI_SCN_OPTION_COLOR_IR	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	✓
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	✓ (*2)
EPS_BI_SCN_OPTION_COLOR_UV (*1)	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	✓
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	✓ (*2)
EPS_BI_SCN_OPTION_GRAYSCALE	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_GRAYSCALE_IR	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	-

BiSCNSetImageTypeOption	BiSCNSelectScanUnit	BiSCNSetImageQuality	Support
EPS_BI_SCN_OPTION_GRAYSCALE_UV (*1)	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_IR	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_UV (*1)	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_DROPOUT_RED	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_DROPOUT_RED_IR	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_DROP-OUT_RED_UV (*1)	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_DROPOUT_GREEN	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-

BiSCNSetImageTypeOption	BiSCNSelectScanUnit	BiSCNSetImageQuality	Support
EPS_BI_SCN_OPTION_DROP- OUT_GREEN_IR	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_DROP- OUT_GREEN_UV (*1)	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_DROPOUT_BLUE	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_DROP- OUT_BLUE_IR	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	-
EPS_BI_SCN_OPTION_DROP- OUT_BLUE_UV (*1)	Check Paper	EPS_BI_SCN_1BIT	✓
		EPS_BI_SCN_8BIT	✓
		EPS_BI_SCN_24BIT	-
	Card	EPS_BI_SCN_1BIT	-
		EPS_BI_SCN_8BIT	✓ (*2)
		EPS_BI_SCN_24BIT	-

*1: The setting is allowed with TM-S2000II UV model and TM-S2000 UV model.

*2: This combination of settings is supported with TM-S2000II UV model and TM-S2000 UV model.

Note

- The BiSCNSetImageQuality, BiSCNSetImageFormat, and BiSCNSetScanArea setting values are reflected in the scanning result when BiSCNMICRFunctionContinuously is executed. Set BiSCNSetImageQuality, and BiSCNSetImageFormat before executing BiSCNMICRFunctionContinuously scanning.
- When this API is executed, the scanned image address is set in lpvScanData of the MF_SCAN structure. This memory is obtained automatically by the API, and it must not be discarded. Therefore, it is necessary for the application to discard it at the appropriate time. To discard this memory, use the application to specify the address of this memory in the GlobalFree function of the Windows API.
- Be sure to configure a valid value for bAddInfoDataSize and pAddInfoData that are part of MF_SCAN structure.
- The interval during which the scan image corresponding to the transaction number (ID) can be acquired is from MF_DATARECEIVE_DONE notification to MF_CHECKPAPER_PROCESS_DONE notification. When the process is returned from the MF_CHECKPAPER_PROCESS_DONE notification handler to the API, the scan image saved by the API is discarded.
- To acquire images scanned using the RGB light source, MF_SCAN_IMAGE_VISIBLE should be set in advance by BiSCNSelectScanImage.
- To acquire images scanned using the infrared light source or the ultraviolet light source, MF_SCAN_IMAGE_2ND_LIGHTSOURCE should be set in advance by BiSCNSelectScanImage.

BiGetScanImageSize

Gets the scan image size.

Syntax

```
int BiGetScanImageSize(int nHandle, DWORD dwTransactionNumber
, LPMF_SCAN ptScan)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwTransactionNumber: Specifies the transaction number (ID) for the scan image acquired. This is a DWORD type.

ptScan: A buffer to store the scan results.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	No data error
ERR_SCAN	-450	Device failed in image scanning
ERR_NOT_EXEC	-470	Process not being executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- Gets the scan size scanned with BiSCNMICRFunctionContinuously / BiSCNMICRFunctionPostPrint.
- After the notification of the scanning status MF_DATARECEIVE_DONE, the scan size is stored in the dwXSize and dwYSize of the structure MF_SCAN through the specification of the related transaction number (ID) in the dwTransactionNumber. Values are not stored in other OUT attribute elements of the MF_SCAN structure.
- To get the front side scanning results, specify MF_SCAN_FACE_FRONT with BiSCNSelectScanFace, then execute this API. To get the back side scanning results, specify MF_SCAN_FACE_BACK with BiSCNSelectScanFace, then execute this API.
- The interval during which the scan image size corresponding to the transaction number (ID) can be acquired is from MF_DATARECEIVE_DONE notification to MF_CHECKPAPER_PROCESS_DONE notification. When the process is returned from the MF_CHECKPAPER_PROCESS_DONE notification handler to the API, the scan image size saved by the API is discarded.

BiGetBarcodeData

Decodes a barcode from the scanned image data, and obtains the result.

Syntax

```
int BiGetBarcodeData (int nHandle, DWORD dwTransactionNumber
, LPMF_BARCODE ptBarcode)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- dwTransactionNumber: Specifies a transaction number (ID) for the BARCODE decode.
- ptBarcode: Specifies the memory address of the MF_BARCODE structure.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	No data error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset
ERR_BARCODE_NODATA	-1130	Barcode cannot be detected.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Decodes a barcode from the scanned image data, and obtains the result.

The result of barcode decode is set in the MF_BARCODE structure. For the details, refer to ["MF_BARCODE" on page 353](#).

To decode a barcode, specify the front/back side with BiSCNSelectScanFace.

Note

- The decoding result of BARCODE corresponding to the transaction number (ID) is obtainable from the processing status of MF_DATARECEIVE_DONE to the notification of MF_CHECKPAPER_PROCESS_DONE. The decoding result of BARCODE kept by API is discarded when the handler of MF_CHECKPAPER_PROCESS_DONE sends back the processing to API.
- When decoding multiple barcodes at the same time, the correct result cannot be obtained if the barcodes other than given below are decoded.
 - Barcode symbols are the same
 - Barcodes are lined in the horizontal direction
- This API is MF_SET_BARCODE_FRONT_PARAM or MF_SET_BARCODE_BACK_PARAM, and can obtain the decoding result different from the MF_BARCODE structure which was set earlier. In such a case, where to eject or the franking process is set according to the current setting. The decoding result can be obtained even if the MF_BARCODE structure is not set earlier.

BiDecodeBarcode

Decodes a barcode from the specified image file, and obtains the result.

Syntax

```
int BiDecodeBarcode (int nHandle, LPCSTR szFileName
                      , LPMF_BARCODE ptBarcode)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
 szFileName: Specify an image file name targeted for the barcode decoding.
 ptBarcode: Specifies the memory address of the MF_BARCODE structure.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_IMAGE_FILEOPEN	-230	Open failure
ERR_IMAGE_UNKNOWN-FORMAT	-240	Format injustice
ERR_BARCODE_NODATA	-1130	Barcode cannot be detected.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Decodes a barcode from the image file, and obtains the result.

The result of barcode decode is set in the MF_BARCODE structure. For the details, refer to ["MF_BARCODE" on page 353](#).

Note

- Make sure to use the image file created in the following way.
 - Saved file with EPS_BI_SCN_BITMAP or EPS_BI_SCN_TIFF256 specified in BiSCNSetImageFormat
 - Saved file with EPS_BI_SCN_8BIT to bColorDepth specified in BiSCNSetImageQuality
 - Saved file with EPS_BI_SCN_SHARP to bExOption specified in BiSCNSetImageQuality
- When decoding multiple barcodes at the same time, the correct result cannot be obtained if the barcodes other than given below are decoded.
 - Barcode symbols are the same
 - Barcodes are lined in the horizontal direction

BiDecodeBarcodeMemory

Decodes the barcode from the image data on the application memory, and obtains the result.

Syntax

```
int BiDecodeBarcodeMemory (int nHandle, LPBYTE lpImageBuffer
                             , DWORD dwBufferSize, LPMF_BARCODE ptBarcode)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

lpImageBuffer: Specifies the memory of image data.

dwBufferSize: Specifies the memory size of specified image data in lpImageBuffer.
Value can be specified to any whole number, 1 or higher.

ptBarcode: Specifies the memory address of the MF_BARCODE structure.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_IMAGE_UNKNOWN-FORMAT	-240	Format injustice
ERR_BARCODE_NODATA	-1130	Barcode cannot be detected.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Decodes a barcode from the image file, and obtains the result.


The result of barcode decode is set in the MF_BARCODE structure. For the details, refer to ["MF_BARCODE" on page 353](#).

Note

- Make sure to use the image file created in the following way.
 - Saved file with EPS_BI_SCN_BITMAP or EPS_BI_SCN_TIFF256 specified in BiSCNSetImageFormat
 - Saved file with EPS_BI_SCN_8BIT to bColorDepth specified in BiSCNSetImageQuality
 - Saved file with EPS_BI_SCN_SHARP to bExOption specified in BiSCNSetImageQuality
- When decoding multiple barcodes at the same time, the correct result cannot be obtained if the barcodes other than given below are decoded.
 - Barcode symbols are the same
 - Barcodes are lined in the horizontal direction

BiSetMonInterval

This API is compatible. No operation is possible with the TM-S9000/S2000 API.

 This API is compatible with the TM-J9000/TM-S1000 API.

Syntax


int **BiSetMonInterval** (int nHandle, WORD wNoPrnInterval, WORD wPrnInterval)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- wNoPrnInterval: Not used
- wPrnInterval: Spooler status monitoring interval is set in units of msec.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset

 For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiGetPrintStation

Gets the set station for printing.

Syntax

int **BiGetPrintStation** (int nHandle, LPWORD lpwStation)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
lpwStation: A value specifiable by BiSetPrintStation can be acquired.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetPrintStation

Sets the station for printing.



- The following settings will be retained for each station that has been set with this API:
 - * BiLoadTemplatePrintArea execution results
 - * BiSetPrintPosition setting
 - * BiSetPrintSize setting
 - * BiSetPrintAlignment setting
- BiSCNMICRFunctionContinuously/BiPrintCutSheet cannot change the station with a callback during execution. Before executing each API, please be sure to set the print station with this API.

Syntax

```
int BiSetPrintStation(int nHandle, WORD wStation)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

wStation: Specifies the station for printing. One of the following values can be set.

Constant	Description
MF_ST_ROLLPAPER (Not supported in the TM-S2000II and the TM-S2000)	Sets roll paper as the station for printing.
MF_ST_VALIDATION	Sets validation as the station for printing.
MF_ST_E_ENDORSEMENT	Sets electronic endorsement as the station for printing.
MF_ST_E_ENDORSEMENT_BACK	Sets back face electronic endorsement as the station for printing. (Same as MF_ST_E_ENDORSEMENT.)
MF_ST_E_ENDORSEMENT_FRONT	Sets front face electronic endorsement as the station for printing.
MF_ST_PHYSICAL_ENDORSEMENT	Sets physical endorsement as the station for printing.
MF_ST_FEEDER	Sets U-shaped mechanism as the station for printing.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiGetPrintPosition

Gets the currently set printing start position.

Syntax

```
int BiGetPrintPosition (int nHandle, LPWORD lpwHorizontal
                        , LPWORD lpwVertical)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpwHorizontal: Specifies the memory address where the horizontal printing start position is saved.
- lpwVertical: Specifies the memory address where the vertical printing start position is saved.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Gets the setting value of the currently set printing start position.

BiSetPrintPosition

Sets the printing start position.



- This API can only be executed if MF_ST_E_ENDORSEMENT is specified in BiSetPrintStation. This API can be called when electronic endorsement (MF_ST_E_ENDORSEMENT, MF_ST_E_ENDORSEMENT_BACK, MF_ST_E_ENDORSEMENT_FRONT) is specified for BiSetPrintStation.
- If data is being buffered with BiTemplatePrint, nothing will happen even when this API is executed.

Syntax

int **BiSetPrintPosition** (int nHandle, WORD wHorizontal, WORD wVertical)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- wHorizontal: Horizontal printing start position (unit: mm).
- wVertical: Vertical printing start position (unit: mm).

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_NOT_SUPPORT	-100	Unsupported



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

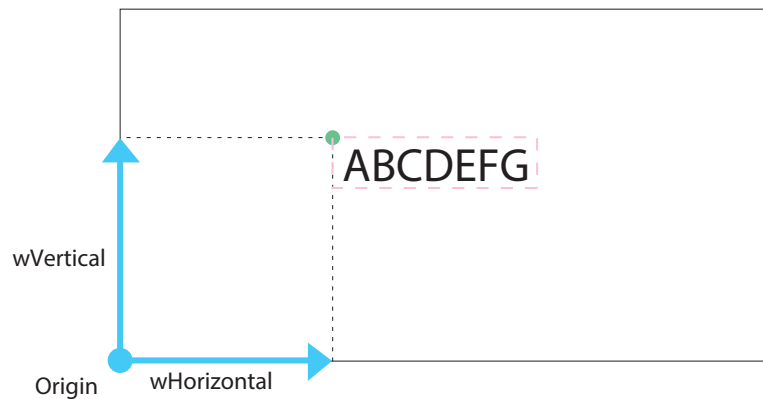
Description

Specifies the printing start position.

The printing start position specified in this API is saved until BiCloseMonPrinter is executed.

The origin for the printing start position specified in this API is the bottom left of the scan image of the back side of the check paper.

Refer to the model diagram below for the specified printing start position and print data expansion method.



Note

When multiple print data is expanded with the same printing start position, it is expanded in duplicate. Adjustment of the printing start position should be performed by the application.

BiSetEndorseDirection

Specifies the direction of electronic endorsement.

Syntax

int **BiSetEndorseDirection** (int nHandle, BYTE bDirection)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
bDirection: Specifies the direction for electronic endorsement. One of the following values can be set.

Constant	Value	Description
EENDORSE_DIRECTION_LEFTRIGHT	1	From left to right (normal direction)
EENDORSE_DIRECTION_TOPBOTTOM	2	From top to bottom (Rotate 90° clockwise)
EENDORSE_DIRECTION_RIGHTLEFT	3	From right to left (upside down)
EENDORSE_DIRECTION_BOTTOMTOP	4	From bottom to top (Rotate 90° counterclockwise)

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error



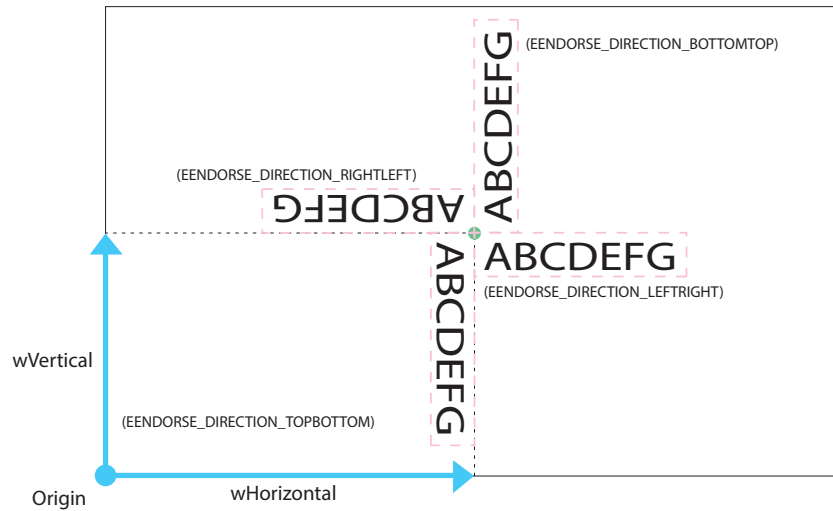
For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The electronic endorsement printing can be executed by executing BiPrintText, BiPrintImage and BiPrintMemoryImage.

Selection of electronic endorsement on the front side/back side can be made by setting BiSetPrintStation; however, the setting of this API is applied to both sides regardless of the front side/back side setting.

The electronic endorsement is rotated around the supporting point specified with BiSetPrintPosition.



BiBufferedPrint

Switches to the buffered print mode and executes buffered printing.



If physical endorsement is selected for BiSetPrintStation, this API cannot be used.

Syntax

int **BiBufferedPrint**(int nHandle, DWORD dwFunction)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
dwFunction: Specifies the function to execute. One of the following values can be set.

Constant	Description
MF_PRT_BUFFERING	Buffers the print data.
MF_PRT_EXEC	Prints the buffered print data.
MF_PRT_CLEAR	Clear the buffered print data and exits buffering mode.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- This API provides data buffering for batch printing and enables the printing of buffered data. The execution status of this API is retained for each station that can be specified using BiSetPrintStation. Execution of batch printing using this API contributes to greater printing performance than when printing methods (BiPrintText, BiPrintImage, BiPrintMemoryImage, BiAutoCutRollPaper) are individually executed.
- On the other hand, when this API is called to buffer print data and execute batch printing, with electronic endorsement (MF_ST_E_ENDORSEMENT, MF_ST_E_ENDORSEMENT_BACK, MF_ST_E_ENDORSEMENT_FRONT) specified using BiSetPrintStation, the electronic endorsement can be extracted while receiving scanned data. Data items that do not require updates are extracted to electronic endorsement using this API so that data items to be extracted inside the handler that notifies the read status MF_DATA_RECEIVE_DONE can be limited to only those items that require routine updates.

BiGetPrintSize

Gets the print size set in the station specified in BiSetPrintStation.

Syntax

int **BiGetPrintSize** (int nHandle, LPWORD lpwWidth, LPWORD lpwHeight)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpwWidth: Specifies the memory address where the horizontal print size is saved.
- lpwHeight: Specifies the memory address where the vertical print size is saved.

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 be used due to another API being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiGetPrintControl

Gets the currently set print control method.



This API is compatible with the TM-J9000 API.

Syntax


int **BiGetPrintControl**(int nHandle, LPBYTE lpbSpeed, LPBYTE lpbDirection)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpbSpeed: Specifies the memory address where the currently set printing speed and print quality are saved.
- lpbDirection: Specifies the memory address where the currently set ink-jet head control method is saved.

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 be used due to another API being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetPrintControl

Sets the print control method.



This API is compatible with the TM-J9000 API.

Syntax

int **BiSetPrintControl**(int nHandle, BYTE bSpeed, BYTE bDirection)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bSpeed: Specifies the printing speed and print quality. One of the following values can be set.

Constant	Description
MF_PRINT_SPEED_NORMAL	Prints with high quality.
MF_PRINT_SPEED_HIGH	Prints with high speed.
MF_PRINT_SPEED_ECONOMY	Prints saving ink.



The TM-S9000/S2000 API ignores this parameter.

bDirection: Specifies the ink-jet head control method. One of the following values can be set.

Constant	Description
MF_PRINT_DIRECTION_DOUBLE	Prints in double direction. There may be some slight overlap between passes (lines), but printing is fast.
MF_PRINT_DIRECTION_SINGLE	Prints in single direction. Printing is slow, but there is no overlap between passes (lines).



The TM-S9000/S2000 API ignores this parameter.

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 be used due to another API being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetPrintSize

Sets the print size.



If MF_ST_PHYSICAL_ENDORSEMENT is selected for BiSetPrintStation, use ["BiTemplatePrint" on page 218](#).

Syntax

int **BiSetPrintSize**(int nHandle, WORD wWidth, WORD wHeight)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
wWidth: Horizontal print size (unit: mm).



- If MF_ST_ROLLPAPER is set for the station, a whole number from 0 to 72 can be designated. (TM-S9000II, TM-S9000)
- If MF_ST_VALIDATION is set for the station, a whole number from 0 to 104 can be designated.
- If MF_ST_E_ENDORSEMENT is set for the station, a whole number from 0 to 255 can be designated.
- If MF_ST_FEEDER is set for the station, a whole number from 0 to the width of the possible printing range designated in the API Setting File/10 can be designated.

wHeight: Vertical print size (unit: mm).



- If MF_ST_ROLLPAPER is set for the station, a whole number from 0 to 65535 can be designated. If a value which exceeds the possible setting range is set, an error is not returned. The information of the low two bytes is set. (TM-S9000II, TM-S9000)
Ex:) If 65540 is set, because 0x00010004, the information of low two bytes, 4, is set.
- If MF_ST_VALIDATION is set for the station, a whole number from 0 to 27 can be designated.
- If MF_ST_E_ENDORSEMENT is set for the station, a whole number from 0 to 255 can be designated.
- If MF_ST_FEEDER is set for the station, a whole number from 0 to the height of the possible print range designated in the API Setting File/10 can be designated

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	Unsupported
ERR_SIZE	-1000	Size excess error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Sets the size of the image printed with BiPrintImage, BiPrintMemoryImage. API automatically enlarges or shrink the image to fit in the specified size. The setting values are saved in each station that can be specified with BiSetPrintStation. If the size specified with this API exceeds the printing area of the station set with BiSetPrintStation, the ERR_SIZE error is returned and the setting value is changed. The setting values specified in this API are saved until BiCloseMonPrinter is executed. The default value for horizontal print size and vertical print size is 0.

BiGetPrintImageMethod

Acquires the print method for BiSetPrintImageMethod.

Syntax

int **BiGetPrintImageMethod** (int nHandle, int* pnMethod)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pnMethod: iSpecifies the memory address where the image print method is set. The set value is as shown below.

Constant	Description
PRINTIMAGEMETHOD_BW	Mono color print (already available)
PRINTIMAGEMETHOD_MULTITONE	Multiple tone print

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetPrintImageMethod

Selected for station printing method used by BiPrintImage.



- This API is for use with the roll paper printing. It can be used when the BiSetPrintStation setting is MF_ST_ROLLPAPER. If executed on the TM-S2000II and the TM-S2000, it won't operate.
- If BiSetPrintStation has been called first, the station setting specified with that API is updated.
- The settings specified with this API will be applied to the printing result.

Syntax

```
int BiSetPrintImageMethod(int nHandle, int nMethod)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

nMethod: How to print images. The following values can be specified independently:

Constant	Description
PRINTIMAGEMETHOD_BW	Mono color print (already available)
PRINTIMAGEMETHOD_MULTITONE	Multiple tone print

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiPrintBarCode

Executes barcode printing. Prints barcodes from the station specified in BiSetPrintStation.

This API is valid for roll paper printer in the TM-S9000II and the TM-S9000.



- Barcodes cannot be printed from electronic endorsement. If this API is executed when MF_ST_E_ENDORSEMENT is specified in BiSetPrintStation, the ERR_NOT_SUPPORT error is returned.
- Since this API does not add a quiet zone, the application should make allowance for a quiet zone.
- Barcode printing cannot be executed for endorsement printing. If this API is executed with MF_ST_PHYSICAL_ENDORSEMENT specified using BiSetPrintStation, the error ERR_NOT_SUPPORT will be returned.
- Barcode printing cannot be executed for cut sheet printing. If this API is executed with MF_ST_FEEDER specified using BiSetPrintStation, the error ERR_NOT_SUPPORT will be returned.

Syntax

```
int BiPrintBarCode(int nHandle, LPBYTE pbyData
                    , DWORD dwDataLength, DWORD dwSymbol
                    , DWORD dwTextPosition)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pbyData: Specifies the memory address where the barcode data is saved.



When designating composite barcode, begin by designating 2-dimensional symbol data, then designate 1-dimensional symbol data in order. And designate 0x3b as the separation between the 2-dimensional symbol data and the 1-dimensional symbol data.

Ex:)

The following is composite barcode data where the first 5 bytes are 2-dimensional symbol data, and the data following 0x3b is one-dimensional symbol data.

0x41,0x41,0x41,0x41,0x41,0x3b,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x31

- dwDataLength: Specifies the length of barcode data.

dwSymbol: Specifies the type of barcode. One of the following values can be set.

Constant	Barcode type
CODE_UPCA	UPC-A
CODE_UPCE	UPC-E
CODE_EAN13	EAN13
CODE_EAN8	EAN8
CODE_CODE39	Code39
CODE_ITF	ITF
CODE_CODABAR	Codabar
CODE_CODE93	Code93
CODE_CODE128	Code128
CODE_GS1_128	GS1-128
CODE_GS1_DATABAR_OMINIDIRECTIONAL	GS1 DataBar Omnidirectional
CODE_GS1_DATABAR_TRUNCATED	GS1 DataBar Truncated
CODE_GS1_DATABAR_LIMITED	GS1 DataBar Limited
CODE_GS1_DATABAR_EXPANDED	GS1 DataBar Expanded
CODE_PDF417	PDF417
CODE_QRCODE	QR Code
CODE_MAXICODE	Maxicode
CODE_GS1_DATABAR_STACKED	GS1 DataBar Stacked
CODE_GS1_DATABAR_STACKED_OMINIDIRECTIONAL	GS1 DataBar Stacked Omnidirectional
CODE_GS1_DATABAR_EXPANDED_STACKED	GS1 DataBar Expanded Stacked
CODE_COMPOSIT	Output as a composite (See "Barcode symbols that can be made into composites" on page 191)

dwTextPosition: Specifies the HRI text position. One of the following values can be set.

Constant	Description
HRI_NONE	No HRI text is added.
HRI_ABOVE	HRI text is added at the top of the barcode.
HRI_BELOW	HRI text is added at the bottom of the barcode.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset
ERR_SIZE	-1000	Size excess error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Data ranges that can be set for barcodes.

Constant	Length	Characters that can be specified
CODE_UPCA	11 to 12	0x30 to 0x39
CODE_UPCE	11 to 12	0x30 to 0x39
CODE_EAN13	12 to 13	0x30 to 0x39
CODE_EAN8	7 to 8	0x30 to 0x39
CODE_CODE39	1 to 255	0x30 to 0x39, 0x41 to 0x5a, 0x20, 0x24, 0x25, 0x2b, 0x2d, 0x2e, 0x2f
CODE_ITF	1 to 255 (odd numbers)	0x30 to 0x39
CODE_CODABAR	1 to 255	0x30 to 0x39, 0x41 to 0x44, 0x24, 0x2b, 0x2d, 0x2e, 0x2f, 0x3a
CODE_CODE93	1 to 255	0x00 to 0x7f
CODE_CODE128	2 to 255	0x00 to 0x7f, "Special characters" on page 190

Special characters

Special characters	ASCII
SHIFT	{S
CODE A	{A
CODE B	{B
CODE C	{C
FNC1	{1
FNC2	{2
FNC3	{3
FNC4	{4
'{'	{{

Barcode symbols that can be made into composites

Constant	Barcode type
CODE_UPCA	UPC-A
CODE_UPCE	UPC-E
CODE_EAN13	EAN13
CODE_EAN8	EAN8
CODE_GS1_128	GS1-128
CODE_GS1_DATABAR_OMINIDIRECTIONAL	GS1 DataBar Omnidirectional
CODE_GS1_DATABAR_TRUNCATED	GS1 DataBar Truncated
CODE_GS1_DATABAR_LIMITED	GS1 DataBar Limited
CODE_GS1_DATABAR_EXPANDED	GS1 DataBar Expanded
CODE_GS1_DATABAR_STACKED	GS1 DataBar Stacked
CODE_GS1_DATABAR_STACKED_OMINIDIRECTIONAL	GS1 DataBar Stacked Omnidirectional
CODE_GS1_DATABAR_EXPANDED_STACKED	GS1 DataBar Expanded Stacked

BiPrintImage

Executes image printing from a file.



If MF_ST_PHYSICAL_ENDORSEMENT is selected for BiSetPrintStation, use ["BiSCNPrintMemoryImage"](#) on page 198.

Syntax

```
int BiPrintImage(int nHandle, LPSTR pFileName)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pFileName: Specifies the location of the image file to print with a full path. Image file that can be specified are below.
- BMP format (Uncompressed image data only)
 - JPEG format (Baseline DCT, Progressive)
 - TIFF format (CCITT Group 3/Group 4 compressed data, uncompressed data only)



The following formats are not supported.

- * BMP format (Compressed image data)
- * JPEG format (Lossless compression, hierarchical coding)
- * TIFF format (Palette color, PackBits/JPEG/LZW/ZIP compression)

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_IMAGE_FILEOPEN	-230	Open failure
ERR_IMAGE_UNKNOWN-FORMAT	-240	Format injustice
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset
ERR_SIZE	-1000	Size excess error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Image printing is executed for the station specified by BiSetPrintStation.

Printing is executed after the image is scaled according to the print size specified with BiSetPrintSize.



- If a station specified using BiSetPrintStation is MF_ST_PHYSICAL_ENDORSEMENT, the print start position must be set using BiSetPrintPosition. If BiSetPrintSize has not been executed, ERR_SIZE will be returned.
- If the image file specified with pfileName does not exist, the error ERR_IMAGE_FILEOPEN will be returned. On the other hand, if an irregular file is specified with pfileName, the error ERR_IMAGE_UNKNOWNFORMAT will be returned.
- If the image size exceeds 4096*4096, an error of ERR_IMAGE_UNKNOWNFORMAT is returned even when the image format meets the supported format.

BiGetPrintAlignment

Gets the inline printing position set in the station specified in BiSetPrintStation.

Syntax

```
int BiGetPrintAlignment (int nHandle, LPDWORD lpdwAlignment)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpdwAlignment: Specifies the memory address where the inline printing position is saved.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the product is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetPrintAlignment

Sets the inline printing position.



Only the following settings are applied to the print station specified for BiSetPrintStation in the settings for this API.

- * MF_ST_VALIDATION
- * MF_ST_ROLLPAPER
- * MF_ST_FEEDER

Syntax

int **BiSetPrintAlignment** (int nHandle, DWORD dwAlignment)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwAlignment: Specifies the inline printing position.

Value	Description
Integer higher than 0 (unit: mm)	Print location shifts a specified amount from the left. MF_ST_VALIDATION: 0 to 104 MF_ST_ROLLPAPER: 0 to 72 MF_ST_FEEDER: 0 to 104
MF_PRINT_ALIGNMENT_LEFT	Prints with left alignment.
MF_PRINT_ALIGNMENT_CENTER	Prints with center alignment.
MF_PRINT_ALIGNMENT_RIGHT	Prints with right alignment.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset
ERR_SIZE	-1000	Size excess error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Specifies the inline printing position of the text, barcode, or image printed with BiPrintText, BiPrintBarCode, or BiPrintImage.

The setting values specified in this API are saved until BiCloseMonPrinter is executed. MF_PRINT_ALIGNMENT_LEFT is specified as the default for the inline printing position.

BiPrintMemoryImage

Executes image printing from memory.



If physical endorsement is selected for BiSetPrintStation, use ["BiSCNPrintMemoryImage" on page 198](#)

Syntax

```
int BiPrintMemoryImage(int nHandle, LPBYTE lpblImageData
, DWORD dwDataSize)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

lpblImageData: Specifies image data for printing. Image files that can be specified are below.

- BMP format (Uncompressed image data only)
- JPEG format (Baseline DCT, Progressive)
- TIFF format (CCITT Group 3/Group 4 compressed data, uncompressed data only)



The following formats are not supported.

- * BMP format (Compressed image data)
- * JPEG format (Lossless compression, hierarchical coding)
- * TIFF format (Palette color, PackBits/JPEG/LZW/ZIP compression)

dwDataSize: Specifies a size of image data to be printed.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_IMAGE_UNKNOWN-FORMAT	-240	Format injustice
ERR_SIZE	-1000	Size excess error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Image printing is executed for the station specified by BiSetPrintStation.

Printing is executed after the image is scaled according to the print size specified with BiSetPrintSize.



- If file data that does not meet the rule is specified to lpblImageData, the ERR_IMAGE_UNKNOWNFORMAT error is returned.
- If the image size exceeds 4096*4096, an error of ERR_IMAGE_UNKNOWNFORMAT is returned even when the image format meets the supported format.

BiSCNPrintMemoryImage

Executes image printing from memory.



This API is for use with physical endorsement. It can be used when the BiSetPrintStation setting is MF_ST_PHYSICAL_ENDORSEMENT.

Syntax

```
int BiSCNPrintMemoryImage
    (int nHandle, unsigned long ulTransactionNumber
    , LPBYTE lpblImageData, DWORD dwDataSize)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

ulTransactionNumber:

Transaction number (ID) corresponding to the check sheet from which the processing status has been issued.

lpblImageData: Specifies image data for printing. Image files that can be specified are below.

- BMP format (Uncompressed image data only)
- JPEG format (Baseline DCT, Progressive)
- TIFF format (CCITT Group 3/Group 4 compressed data, uncompressed data only)



The following formats are not supported.

- * BMP format (Compressed image data)
- * JPEG format (Lossless compression, hierarchical coding)
- * TIFF format (Palette color, PackBits/JPEG/LZW/ZIP compression)

dwDataSize: Specifies a size of image data to be printed.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	Data not found
ERR_IMAGE_UNKNOWN-FORMAT	-240	Format injustice
ERR_SIZE	-1000	Size excess error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Image printing is executed for the station specified by BiSetPrintStation.


Printing is executed after the image is scaled according to the print size specified with BiSetPrintSize.



- If file data that does not meet the rule is specified to lpblImageData, the ERR_IMAGE_UNKNOWNFORMAT error is returned.
- If the image size exceeds 4096*4096, an error of ERR_IMAGE_UNKNOWNFORMAT is returned even when the image format meets the supported format.

BiPrintText

Executes text printing.



If physical endorsement is selected for BiSetPrintStation, use ["BiSCNPrintText" on page 202](#)

Syntax


```
int BiPrintText(int nHandle, LPSTR szText, MF_DECORATE tDecorate)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- szText: Specifies the memory address where the text for printing is saved.
- tDecorate: Specifies the MF_DECORATE structure where the text decoration information is saved.
Refer to ["MF_DECORATE" on page 360](#).

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Prints text with the attributes specified with the MF_DECORATE structure, from the station specified in BiSetPrintStation.



- If a station specified using BiSetPrintStation is MF_ST_PHYSICAL_ENDORSEMENT, the printed strings will be MF_PRINT_BLACK, even when MF_PRINT_COLOR or MF_PRINT_MIXED is specified for dwAttribute of the MF_DECORATE structure.
- When specifying the MF_PRINT_SYSTEMFONT in the wFont of the MF_DECORATE structure, the thickness of the underline is the same whether you specify MF_PRINT_UNDERLINE_1 or MF_PRINT_UNDERLINE_2.
- An ERR_PARAM is returned if BiSetPrintStation is one of the following, and anything other than SYSTEMFONT is specified for wFont.
 - * MF_ST_E_ENDORSEMENT
 - * MF_ST_E_ENDORSEMENT_BACK
 - * MF_ST_E_ENDORSEMENT_FRONT
 - * MF_ST_PHYSICAL_ENDORSEMENT
- An ERR_PARAM is returned if a value outside those following is specified for wFontSize when wFont is not SYSTEMFONT.
 - * MF_PRINT_FONT_W1_H1
 - * MF_PRINT_FONT_W1_H2
 - * MF_PRINT_FONT_W2_H1
 - * MF_PRINT_FONT_W2_H2

BiSCNPrintText

Executes text printing.



This API is for use with physical endorsement. It can be used when the BiSetPrintStation setting is MF_ST_PHYSICAL_ENDORSEMENT.

Syntax

```
int BiSCNPrintText(int nHandle
                    , unsigned long ulTransactionNumber
                    , LPSTR szText, MF_DECORATE* pDecorate)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- ulTransactionNumber: Transaction number (ID) corresponding to the check sheet from which the processing status has been issued.
- szText: Specifies the memory address where the text for printing is saved.
- pDecorate: Specifies the MF_DECORATE structure where the text decoration information is saved.
Refer to "[MF_DECORATE](#)" on page 360.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	Data not found
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to "[Return value](#)" on page 61.

Description

Prints text with the attributes specified with the MF_DECORATE structure, from the station specified in BiSetPrintStation.



- If a station specified using BiSetPrintStation is MF_ST_PHYSICAL_ENDORSEMENT, the printed strings will be MF_PRINT_BLACK, even when MF_PRINT_COLOR or MF_PRINT_MIXED is specified for dwAttribute of the MF_DECORATE structure.
- When specifying the MF_PRINT_SYSTEMFONT in the wFont of the MF_DECORATE structure, the thickness of the underline is the same whether you specify MF_PRINT_UNDERLINE_1 or MF_PRINT_UNDERLINE_2.
- An ERR_PARAM is returned if BiSetPrintStation is one of the following, and anything other than SYSTEMFONT is specified for wFont.
 - * MF_ST_E_ENDORSEMENT
 - * MF_ST_E_ENDORSEMENT_BACK
 - * MF_ST_E_ENDORSEMENT_FRONT
 - * MF_ST_PHYSICAL_ENDORSEMENT
- An ERR_PARAM is returned if a value outside those following is specified for wFontSize when wFont is not SYSTEMFONT.
 - * MF_PRINT_FONT_W1_H1
 - * MF_PRINT_FONT_W1_H2
 - * MF_PRINT_FONT_W2_H1
 - * MF_PRINT_FONT_W2_H2

BiPrintMultipleToneImage

Multiple-tone images registered in NVRAM are printed. Not supported in the TM-S2000II and the TM-S2000.



- The roll sheet must be selected in advance using BiSetPrintStation.
- If images registered in NVRAM are larger than the width of the roll sheet, printing is terminated without anything being printed.
- The image size set using BiSetPrintSize will not be applied, and the image will be output according to the registered image size.

Syntax

```
int BiPrintMultipleToneImage(int nHandle, char szKeyCode1, char szKeyCode2)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- szKeyCode1: Specify the first key code for the multiple tone image to be printed.
A value from 0x20 to 0x7E can be specified.
- szKeyCode2: Specify the second key code for the multiple tone image to be printed.
A value from 0x20 to 0x7E can be specified.



If the image for the designated key code is not found, the image is not printed.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Paper other than roll papers is set in the print station.
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiUpdateEndorseText

Updates an endorse character string.



This API is compatible with the TM-J9000/TM-S1000 API.

Syntax

```
int BiUpdateEndorseText(int nHandle, LPSTR lpString[3]
                        , DWORD dwAttribute[3]
                        , WORD wFont[3]
                        , WORD wFontSize[3])
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpString[3]: Specifies an endorse character string. lpString[0] is the first line, lpString[1] is the second line, and lpString[2] is the 3rd line.
- dwAttribute[3]: Specifies an attribute of the endorse character string. dwAttribute[0] is the first line, dwAttribute[1] is the second line, and dwAttribute[2] is the 3rd line.

dwAttribute (Constant)	Description
MF_PRINT_BOLD	Executes emphasized printing
MF_PRINT_UNDERLINE_1	Adds a 1-line width of UnderLine
MF_PRINT_UNDERLINE_2	Adds a two-line width of UnderLine
MF_PRINT_REVERSEVIDEO	Executes reverse printing
MF_PRINT_BLACK	Prints a character with the first color (usually it is black)
MF_PRINT_COLOR	<ul style="list-style-type: none"> Prints a character with the second color. Even if this value is set, text is printed using the 1st color.
MF_PRINT_MIXED	<ul style="list-style-type: none"> Prints a character with the first and second colors Even if this value is set, text is printed using the 1st color.
MF_PRINT_1ST_COLOR	Prints a character with the first color (usually it is black)
MF_PRINT_2ND_COLOR	Prints a character with the second color
MF_PRINT_NO_ATTRIBUTE	Not specifying the attribute

- wFont[3]: Specifies a font of the endorse character string. wFont[0] is the first line, wFont[1] is the second line, and wFont[2] is the 3rd line.

wFont (Constant)	Description
MF_PRINT_SYSTEMFONT	Prints with the system font
MF_PRINT_FONT_A	Prints with the FontA
MF_PRINT_FONT_B	Prints with the FontB

wFontSize[3]: Specifies a font size of the endorse character string. wFontSize [0] is the first line, wFontSize [1] is the second line, and wFontSize [2] is the 3rd line.

wFontSize (Constant)	Description
MF_PRINT_FONT_W1_H1	A font with 1 unit horizontal and 1 vertical
MF_PRINT_FONT_W1_H2	A font with 1 unit horizontal and 2 vertical
MF_PRINT_FONT_W2_H1	A font with 2 units horizontal and 1 vertical
MF_PRINT_FONT_W2_H2	A font with 2 units horizontal and 2 vertical

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	A scan is being executed.
ERR_NOT_EXEC	-470	Process not being executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The endorse character string can be updated by invoking this API from the notification handler of MF_DATARECEIVE_DONE, the reading status of BiSCNMICRFunctionPostPrint.



When the MF_PRINT structure is not set with BiSCNMICRFunctionPostPrint and endorsement printing is not executed, if the API is executed, the ERR_EXEC_FUNCTION error is returned and the endorse character string cannot be updated.

BilInsertValidation

Inserts the paper in validation and makes the product ready for printing.



This API is compatible with the TM-J9000 API.

Syntax

int **BilInsertValidation** (int nHandle, DWORD dwTimeout)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
dwTimeout: Specifies the timeout time until the paper is inserted in the validation slot.
A value from 0 to 300 in second units can be specified. When 0 is specified, no timeout time is set, and the product waits until paper is inserted.

Return value


Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Cannot R/W to product
ERR_PARAM	-90	Parameter error
ERR_PAPERINSERT_TIMEOUT	-300	Paper insertion time exceeded.
ERR_EXEC_FUNCTION	-310	Cannot be used due to another API being executed.
ERR_RESET	-400	Cannot be used because the device is being reset.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiRemoveValidation

Ejects the paper inserted in validation, and performs monitoring until it is removed from the insertion slot.

 This API is compatible with the TM-J9000 API.

Syntax


int **BiRemoveValidation** (int nHandle, DWORD dwTimeout)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- dwTimeout: Specifies the timeout time until the paper is inserted in the validation slot.
A value from 0 to 300 in second units can be specified. When 0 is specified, no timeout time is set, and the product waits until paper is inserted.

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 R/W to product
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used due to another API being executed.
ERR_RESET	-400	Cannot be used because the device is being reset.

 For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiGetPrintCutSheetSettings

Acquires the action settings for cut paper printing.



A value specifiable by BiSetPrintCutSheetSettings can be acquired.

Syntax

```
int BiGetPrintCutSheetSettings
    (int nHandle, unsigned char* pucNumber,
     unsigned char* pucScan)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
 pucNumber: Specifies the memory address where the number of cut papers to print is saved.
 pucScan: Specifies the memory address where the option for simultaneous scan execution is set.
 The set value is as shown below.

Constant	Description
BI_PRINTCUTSHEET_SCAN	Scans print data when performing cut sheet printing.
BI_PRINTCUTSHEET_NOSCAN	Does not scan print data when performing cut sheet printing.

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 be used due to another API being executed.
ERR_RESET	-400	Cannot be used because the device is being reset.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetPrintCutSheetSettings

Sets the operation of cut sheet paper printing.



The detailed scan settings are specified using ["BiPrintCutSheet"](#) on page 211.

Syntax

```
int BiSetPrintCutSheetSettings
    (int nHandle, unsigned char ucNumber
    , unsigned char ucScan)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

ucNumber: Number of cut sheet papers to be printed.
The following value can be specified independently: Integer in the range from 1 to 255

ucScan: Whether or not simultaneous scan is executed.
The following values can be specified independently:

Constant	Description
BI_PRINTCUTSHEET_SCAN	Scans print data when performing cut sheet printing.
BI_PRINTCUTSHEET_NOSCAN	Does not scan print data when performing cut sheet printing.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value"](#) on page 61.

BiPrintCutSheet

Prints a cut sheet and executes simultaneous scanning.

Print data is specified inside the print ready callback, in the same way as when endorsement printing is executed using BiSCNMICRFunctionContinuously.



- This API is used in the same manner as ["BiSCNMICRFunctionContinuously" on page 106](#).
- Before executing this API, please be sure to set the MF_ST_FEEDER with BiSetPrintStation.
- Notification that an error has occurred is in the callback registered in BiSCNMICRSetStatusBackFunction/BiSCNMICRSetStatusBackWndEx.
If notification of an error is required, register the callback in BiSCNMICRSetStatusBackFunction/BiSCNMICRSetStatusBackWndEx beforehand.

Syntax

```
int BiPrintCutSheet(int nHandle, void* lpvStruct
                    , unsigned short usFunction)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpvStruct: The address of the parameter structure specified for each unit.
Refer to ["Supplemental Description for lpvStruct" on page 116](#).

wFunction (Constant)	Parameter to Set for lpvStruct
MF_EXEC	"NULL"
MF_SET_BASE_PARAM	Address of the MF_BASE01 structure
MF_SET_SCAN_PRINTAREA_PARAM	Address of the MF_SCAN structure for the print side
MF_SET_SCAN_NOPRINTAREA_PARAM	Address of the MF_SCAN structure for the non-print side
MF_SET_PROCESS_PARAM	Address of the MF_PROCESS01 structure
MF_CLEAR_BASE_PARAM	Address of the MF_BASE01 structure
MF_CLEAR_SCAN_PRINTAREA_PARAM	Address of the MF_SCAN structure for the print side
MF_CLEAR_SCAN_NOPRINTAREA_PARAM	Address of the MF_SCAN structure for the non-print side
MF_CLEAR_PROCESS_PARAM	Address of the MF_PROCESS01 structure
MF_GET_BASE_DEFAULT	Address of the MF_BASE01 structure
MF_GET_SCAN_PRINTAREA_PARAM_DEFAULT	Address of the MF_SCAN structure for the print side
MF_GET_SCAN_NOPRINTAREA_PARAM_DEFAULT	Address of the MF_SCAN structure for the non-print side
MF_GET_PROCESS_DEFAULT	Address of the MF_PROCESS01 structure

usFunction: Specifies the functions for the API to execute.
 For settings regarding processing, by specifying MF_SET_SCAN_PRINTAREA_PARAM and so on and executing this function, the API is notified. If the members of the structure are changed after notification, it is necessary to perform notification again with this function. Include the header file submitted for the definition name used for this API.
 Refer to ["Supplemental Description for wFunction" on page 117](#).

Supplemental Description for lpvStruct

For an explanation of the lpvStruct structure, refer to ["Structures" on page 277](#). However, the following values are not used with this API, or the values are not set.

<MF_BASE01 structure>

- dwNotifyType and uNotifyHandle are not used.
 Notification of the reading status of this API is performed by the handler registered with BiSCNMICRSetStatusBackFunction.
- hProgressWnd is not used
 This API does not provide notification of the progress status.

<MF_SCAN structure>

- wImageID is not used.
 Set the transaction number (ID) using BiSetTransactionNumber
- Do not set values in bStatus, bDetail, dwXSize, dwYSize, dwScanSize, and lpbScanData.
 Set values when getting the scan image with BiGetScanImage.

<MF_PROCESS structure>

Following member can be set.

- dwStartWaitTime;
- bDoubleFeedErrorSelect;
- bDoubleFeedErrorEject;
- bDoubleFeedCancel;
- bNearFullSelect;
- bResultPartialData;
- bPrnDataLenExceedErrorEject;
- bPrnDataLenExceedCancel;

Supplemental Description for wFunction

wFunction (Constant)	Description
MF_EXEC	If MF_EXEC is set, the printing of a cut sheet is started.
MF_SET_BASE_PARAM	If MF_SET_BASE_PARAM is set, the information of the specified BASE structure will be set.
MF_SET_SCAN_FRONT_PARAM	If MF_SET_SCAN_PRINTAREA_PARAM is set, the information of the specified SCAN structure is set as the print side information.
MF_SET_SCAN_BACK_PARAM	If MF_SET_SCAN_NOPRINTAREA_PARAM is set, the information of the specified SCAN structure is set as the non-print side information.
MF_SET_PROCESS_PARAM	This sets process parameters. The MF_PROCESS01 structure address is specified in lpvStruct.
MF_CLEAR_BASE_PARAM	If MF_CLEAR_BASE_PARAM is set, the information of the specified BASE structure is cleared.
MF_CLEAR_SCAN_FRONT_PARAM	If MF_CLEAR_SCAN_PRINTAREA_PARAM is set, the information of the specified SCAN structure (print side) is cleared.
MF_CLEAR_SCAN_BACK_PARAM	If MF_CLEAR_SCAN_NOPRINTAREA_PARAM is set, the information of the specified SCAN structure (non-print side) is cleared.
MF_CLEAR_PROCESS_PARAM	This clears the process parameter specifications. lpvStruct values are ignored.
MF_GET_BASE_DEFAULT	If MF_GET_BASE_PARAM_DEFAULT is specified, the default BASE setting is set in lpvStruct. (Refer to "Default Values of the MF_BASE01 Structure" on page 280.)
MF_GET_SCAN_FRONT_DEFAULT	If MF_GET_SCAN_PRINTAREA_PARAM_DEFAULT is specified, the default print side SCAN setting is set in lpvStruct. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_SCAN_BACK_DEFAULT	If MF_GET_SCAN_PRINTAREA_PARAM_DEFAULT is specified, the default non-print side SCAN setting is set in lpvStruct. (Refer to "Default Values of the MF_SCAN Structure" on page 285.)
MF_GET_PROCESS_DEFAULT	This obtains the initial values for the process structure. (Refer to "Default Values of the MF_PROCESS01 Structure" on page 311.)

Return value*BiPrintCutSheet*

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_OFFLINE	-110	Waiting to return from an offline state
ERR_PAPERINSERT_TIMEOUT	-300	Failed to insert paper
ERR_EXEC_FUNCTION	-310	Another API is running
ERR_RESET	-400	Device is being reset
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_SCAN	-450	Failed to read the image data
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_PAPER_JAM	-1020	Paper jam error
ERR_PAPER_EXIST	-1090	API can not be execute because there is a paper on the path
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_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_MICR	-440	Failed to read the MICR data
ERR_SCAN	-450	Failed to read the image data
ERR_LINE_OVERFLOW	-460	Line overflow occurred during transaction printing
ERR_NOT_EXEC	-470	Reading process not executed
ERR_PAPER_JAM	-1020	Paper jam error
ERR_PAPER_EXIST	-1090	API can not be execute because there is a paper on the path
ERR_PAPER_INSERT	-1100	Failed to insert paper

MF_SCAN.iRet

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_ABORT	-430	Canceled by BiSCNMICRCancelFunction
ERR_SCAN	-450	Device failed in image scanning
ERR_NOT_EXEC	-470	Process not being executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiAutoCutRollPaper

Executes auto-cutter for the roll paper. Not supported in the TM-S2000II and the TM-S2000.



- If this API is called while data is being buffered using BiBufferedPrint/BiTemplatePrint, auto-cutter will not be executed when it is called, but will be executed when buffered data is printed.
- The roll sheet must be selected in advance using BiSetPrintStation.

Syntax

```
int BiAutoCutRollPaper(int nHandle, int nMethod)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

nMethod: How to execute auto-cutter. The following values can be specified independently:

Constant	Description
AUTOCUT_CUT	Sets roll paper as the station for printing.
AUTOCUT_FEEDANDCUT	Sets validation as the station for printing.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Paper other than roll sheets is set in the print station
ERR_EXEC_FUNCTION	-310	A scan is being executed
ERR_RESET	-400	Cannot be used because the device is being reset
ERR_PAPER_INSERT	-1100	No paper is set in the print station



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiLoadTemplatePrintArea

Loads a template printing setting file.

The area information loaded with this API can be specified with ["BiSetTemplatePrintArea" on page 220](#).

Refer to ["Template printing setting file" on page 403](#) for details.



- If correct information and invalid information coexist, only the correct information is loaded.
- If this API is executed again, the previously loaded information will be discarded.
- If more than 255 normal data items exist, only 255 data items will be loaded, with the remaining data items being ignored.

Syntax

int **BiLoadTemplatePrintArea**

(int nHandle, char* pszFilePath, int* pnLoadNum)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pszFilePath: Template printing setting file to be loaded. The full path to the file must be specified. Even when just the file name or the relative path is specified, it will not be regarded as an error as long as the file can be loaded.
- pnLoadNum: The number of areas loaded is stored. If an error occurs, nothing will be set. If NULL is specified, nothing happens.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset
ERR_DATA_INVALID	-480	Not have print area information, contains incorrect information on print area.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiTemplatePrint

Buffers data for template printing as well as prints buffered data.



- The execution status of this API is retained for each station that can be specified using BiSetPrintStation. If BiSetPrintPosition is executed while data is being buffered, the value will be not applied.
- If an API for printing is executed while data is being buffered, the data will be saved internally instead of being printed.
- If BiSetPrintSize is executed while data is being buffered, the value will be not applied.

Syntax

int **BiTemplatePrint**(int nHandle, int nFunction)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

nFunction: Specifies the execute mode. The following values can be specified independently:

Constant	Description
TEMPLATEPRINT_BUFFERING	<p>Starts the buffering mode of template-specified print. During buffering mode, even if a print-related API is executed, printing is not executed and print information is retained.</p> <ul style="list-style-type: none"> • If TEMPLATEPRINT_BUFFERING is set, format printing will be enabled in buffering mode. • If TEMPLATEPRINT_BUFFERING is set during buffering mode, nothing will happen, and SUCCESS will be returned.
TEMPLATEPRINT_EXEC	<p>Prints the information retained when in the buffering mode and terminates the buffering mode. After termination, clears all the information specified during buffering.</p> <ul style="list-style-type: none"> • If buffered data is present when TEMPLATEPRINT_EXEC is set, the buffered data will be printed, which will subsequently be cleared along with the area information. • If no buffered data is present when TEMPLATEPRINT_EXEC is set, nothing will happen, and SUCCESS will be returned. • If TEMPLATEPRINT_EXEC is set, buffering mode will be cancelled.
TEMPLATEPRINT_CLEAR	<p>Clears the retained information and terminates the buffering mode without printing. After termination, clears all the information specified during buffering.</p> <ul style="list-style-type: none"> • If TEMPLATEPRINT_CLEAR is set, buffered data will be cleared along with the area information. • If no buffered data is present when TEMPLATEPRINT_CLEAR is set, nothing will happen, and SUCCESS will be returned. • If TEMPLATEPRINT_CLEAR is set, buffering mode will be cancelled.
TEMPLATEPRINT_EXEC_NOCLEAR	<p>Prints the information retained when in the buffering mode. After print execution, will retain the present information.</p> <ul style="list-style-type: none"> • If buffered data is present when TEMPLATEPRINT_EXEC_NOCLEAR is set, the buffered data will be printed (but the information will not be cleared). • If no buffered data is present when TEMPLATEPRINT_EXEC_NOCLEAR is set, nothing will happen, and SUCCESS will be returned.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- Execution of batch printing using this API contributes to greater printing performance than when BiPrint-Text is individually executed.
- On the other hand, when this API is called to buffer print data and execute batch printing, with MF_ST_E_ENDORSEMENT specified using BiSetPrintStation, the electronic endorsement can be extracted while receiving scanned data. Data items that do not require updates are extracted to electronic endorsement using this API so that data items to be extracted inside the handler that notifies the read status MF_DATARECEIVE_DONE can be limited to only those items that require routine updates.
- If EXEC_NOCLEAR is executed, area information will be retained, but buffering mode will be cancelled, as is the case with EXEC_CLEAR. Therefore, buffering must be executed again.

BiSetTemplatePrintArea

Selects print areas from print area information and sets the selected areas.

Print-type APIs are executed for the areas specified with this API.



- Any portions that lie outside the print area will not be printed.
- To specify area names, print area information must be loaded in advance with ["BiLoadTemplatePrintArea" on page 217](#).
- BiTemplatePrint must be called in advance with buffering mode enabled.
- This API can be used only when BiTemplatePrint is executed in buffering mode.
- Although the area that lies outside the printable area can be specified, nothing will be printed.

Syntax

```
int BiSetTemplatePrintArea(int nHandle, int nAreaSelectMode
                           , char* pszAreaName
                           , LPPRINTAREAINFO psInfo
                           , unsigned long ulTransactionNumber)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

nAreaSelectMode:

How to set areas.

Constant	Description
SELECTPRINTAREA_AREANAME	For the area information, sets the data defined by the area name in the loaded setting data.
SELECTPRINTAREA_DIRECT	For the area information, sets the data defined by the specified structure.

pszAreaName: Area name.



- NULL can be specified if direct specification is selected.
- The value will be ignored if direct specification is selected.

psInfo: Print area information (Pointer to the print area information structure).

Refer to ["PRINTAREAINFO Structure" on page 221](#).



- NULL can be specified if area name specification is selected.
- The value will be ignored if area specification is selected.

ulTransactionNumber:

Specifies the target transaction number.



When executing at stations other than the physical endorsement station, please designate TRANS_NO_SPECIFIED.
If a value other than TRANS_NO_SPECIFIED is designated, the operation is definitely done to the physical endorsement station.

PRINTAREAINFO Structure

```
typedef struct {
    char szAreaName[128];           IN
    int nMeasure;                   IN
    int nOriginX;                   IN
    int nOriginY;                   IN
    int nWidth;                     IN
    int nHeight;                    IN
    int nRotate;                    IN
}PRINTAREAINFO, *LPPRINTAREAINFO;
```

szAreaName: Area name.

nMeasure: Specifies the coordinates of the origin, and the unit of measurement for the values specified for the area width and area height.

Each of the following values can be specified independently:

Constant	Description
MEASURE_MM	The unit of measurement should be mm.
MEASURE_INCH_01	The unit of measurement should be 0.1 inches.

nOriginX: The x coordinate of the origin position. The top left of the printable area is set as the origin. (Maximum value that can be designated for the X direction larger than 0.)

nOriginY: The y coordinate of the origin position. The top left of the printable area is set as the origin. (Maximum value that can be designated for the Y direction larger than 0.)

nWidth: Area width.
(Maximum value that can be designated for the X direction larger than 1.)

nHeight: Area height.
(Maximum value that can be designated for the Y direction larger than 1.)

nRotate: Area rotation. Each of the following values can be specified independently:

Constant	Description
IMAGEROTATE_0	The printing does not rotate.
IMAGEROTATE_90	The printing rotates 90 degrees counterclockwise around the origin.
IMAGEROTATE_180	The printing rotates 180 degrees counterclockwise around the origin.
IMAGEROTATE_270	The printing rotates 270 degrees counterclockwise around the origin.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	Data not found
ERR_NOT_EXEC	-470	Process not being executed

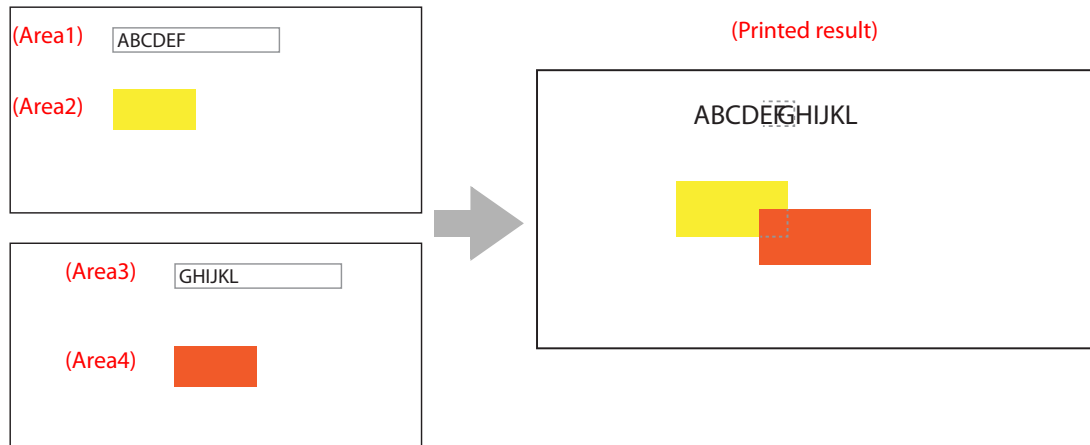


For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description


When there are overlapping areas, the one printed later will lie on top of the one printed earlier.

Example



BiClearTemplatePrintData

Clears the print data located in the print area specified with BiSetTemplatePrintArea.




- BiTemplatePrint must be called in advance with buffering mode enabled.
- This API can be used only when BiTemplatePrint is executed in buffering mode.

Syntax

```
int BiClearTemplatePrintData  
    (int nHandle, unsigned long ulTransactionNumber)
```

Argument


nHandle: Handle obtained with the return value of the communication initialization API.
ulTransactionNumber:
 Specifies the target transaction number to be cleared.



When executing at stations other than the physical endorsement station, please designate TRANS_NO_SPECIFIED.
If a value other than TRANS_NO_SPECIFIED is designated, the operation is definitely done to the physical endorsement station.

Return value


Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_NOT_FOUND	-220	Transaction number (ID) not found
ERR_NOT_EXEC	-470	Format-specified print mode has not been executed with BiTemplate-Print.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiGetTransactionNumber

Gets the currently set transaction number.



- The transaction number (ID) acquired by this API is the value used for the next scanning process, and it cannot be used as a parameter for BiGetMicrText and BiGetScanImage.
- When getting MICR text or a scan image with BiGetMicrText and BiGetScanImage, use the transaction number (ID) provided with the scanning status MF_DATARECEIVE_DONE or MF_CHECKPA-
PER_PROCESS_DONE.

Syntax


```
int BiGetTransactionNumber  
    (int nHandle, LPDWORD lpdwTransactionNumber)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpdwTransactionNumber: Specifies the memory address where the transaction number (ID) is saved.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetTransactionNumber

Sets the transaction number.

Syntax

```
int BiSetTransactionNumber
(int nHandle, DWORD dwTransactionNumber)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwTransactionNumber: Specifies the transaction number (ID) to set. The range of values that can be set is from 0 to 999999999.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Sets the transaction number (ID) used for sequential printing, BiSCNMICRFunctionContinuously scanning status notification. After MF_CHECKPAPER_PROCESS_START notification with BiSCNMICRFunctionContinuously, 1 is added to the setting value. If MF_CHECKPAPER_PROCESS_START notification is performed at the maximum value of 999999999, the setting value returns to 0.

The transaction number (ID) default value is set to 1.

<About sequential printing function>

- It is the function that specifies a format for transaction number (ID) printing. It prints using the format of the keyword surrounded by <>.
- If the keyword surrounded by <> is specified to the endorsement printing character string before the reading process, character string is made using the value acquired by BiGetTransactionNumber. The character string is valid until either BiBufferedPrint MF_PRT_CLEAR or BiBufferedPrint invokes BiPrintText in the MF_PRT_EXEC status.
- If the keyword surrounded by <> is specified to the endorsement printing character string during the MF_DATARECEIVE_DONE callback, character string is made using the transaction ID for the check sheet. This character string is cleared after the MF_DATARECEIVE_DONE callback is notified.

- There are three patterns for the keywords that can be specified for the sequential printing as follows:

- <0000>

The number of "0"s indicates the number of columns to be printed. The number of columns that can be set is from 1 to 9. If the transaction number set is less than the number of the columns, 0 is added automatically.

- <xxxx>

The number of "x"s indicates the number of columns to be printed. The number of the columns that can be set is from 1 to 9. If the transaction number set is less than the number of the columns, a space is added automatically.

- <llll>

(small letter of "l" in one-byte): The number of "l"s indicates the number of columns to be printed. The number of the columns that can be set is from 1 to 9. If the transaction number set is less than the number of the columns, the columns are left aligned automatically.



- '<' and '>' are used as special symbols. When printing '<' or '>', specify &< or &> respectively.
- If a keyword is specified that is outside the rule mentioned above, for example <00xxabc> (0 and x are mixed, the characters other than 0 or x are included), the character string surrounded by '<' and '>' is printed as a usual character string.



With sequential printing, if a number of digits n smaller than the transaction number (ID) currently set is specified, the latter n digits of the transaction number (ID) are actually printed.

<Example>

When the transaction number (ID) currently set is 12345 and the sequential printing is specified with 4-column <xxxx>, "2345" is actually printed.

BiSetTransactionNumberWithIncremental

Sets the transaction number and the incremental value.

Syntax

```
int BiSetTransactionNumberWithIncremental
    (int nHandle, DWORD dwTransactionNumber
    , DWORD dwIncremental)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwTransactionNumber: Specifies the transaction number (ID) to set. The range of values that can be set is from 0 to 999999999.

dwIncremental: Specifies the incremental value of transaction number to set. The range of values that can be set is from 1 to 999999999.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Sets the transaction number (ID) used for sequential printing, BiSCNMICRFunctionContinuously scanning status notification. The set value will be incremented by dwIncremental after the MF_CHECKPAPER_PROCESS_START notification with BiSCNMICRFunctionContinuously and BiSCNMICRFunctionPostPrint. If MF_CHECKPAPER_PROCESS_START notification is performed at the maximum value of 999999999, the setting value returns to 0.

The transaction number (ID) default value is set to 1. The default incremental value is set to 1.

BiSetWaterfallMode

Specifies the Waterfall mode.

Syntax

```
int BiSetWaterfallMode(int nHandle, BYTE bWaterfallMode)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bWaterfallMode:

Specifies the Waterfall mode.

Constant	Description
WATERFALL_MODE_DISABLE	Disables Waterfall mode.
WATERFALL_MODE_STANDARD	Prioritizes the main pocket as the pocket to eject document to. Ejects to the sub pocket when the main pocket is detected as being near full. When the main pocket near full status has been removed document will return to being ejected to the main pocket.
WATERFALL_MODE_INHERIT_POCKET	Prioritizes the pocket that document is ejected to when scanning completed previously as the pocket to eject document to. Switches pockets that document is ejected to when the pocket that document is ejected to is detected as being near full. The pocket that document is ejected to will not switch over when both pockets are detected as being near full.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

When a pocket near full is detected when the Waterfall mode is started, the first document is ejected to the following pocket.

Waterfall mode	Pocket status		Ejection pocket
	Main Pocket	Sub Pocket	
WATERFALL_MODE_STANDARD	Not NearFull	Not NearFull	Main Pocket
	NearFull	Not NearFull	Sub Pocket
	Not NearFull	NearFull	Main Pocket
	NearFull	NearFull	Main Pocket
WATERFALL_MODE_INHERIT_POCKET	Not NearFull	Not NearFull	Inherit Pocket
	NearFull	Not NearFull	Sub Pocket
	Not NearFull	NearFull	Main Pocket
	NearFull	NearFull	Inherit Pocket



- When a pocket near full is detected from both pockets, it is recommended to set bNearFullSelect of MF_PROCESS01 structure to MF_NEARFULL_NOT_PERMIT, to stop the reading process. (See ["MF_PROCESS01" on page 298](#))
- If the product is scanning, this API returns ERR_EXEC_FUNCTION. This API works for High speed and Confirmation.
- When the Waterfall process is enabled, the ejection pocket setting for MF_PROCESS01 structure and BiSelectErrorEjectAtContinuously is disabled.

BiGetIQAResult

Gets IQA result.



- This API can be used when the BiSCNSetImageQuality settings are EPS_BI_SCN_1BIT(Black and White) or EPS_BI_SCN_8BIT(RGB grayscale).
- For card scans, you cannot get IQA results with this API.

Syntax

int **BiGetIQAResult**

(int nHandle, DWORD dwTransactionNumber,
LPMF_IQA_RESULT lpResult)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwTransactionNumber:

Specifies the transaction number (ID) for the MICR text acquired.

lpResult: Specifies the memory address of the MF_IQA_RESULT/MF_IQA_RESULT01 structure.

The following result is set for each IQA validation item.

Constant	Description
IQARESULT_NOT_TESTED	IQA validation is not executed.
IQARESULT_PASS	Passed IQA validation.
IQARESULT_NOT_PASS	Not passed IQA validation.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_BUFFER_OVER_FLOW	-140	Buffer overflow error
ERR_NOT_FOUND	-220	No data error
ERR_NOT_EXEC	-470	Process not being executed
ERR_EXEC_FUNCTION	-310	Cannot be used due to another API being executed.
ERR_RESET	-400	Cannot be used because the device is being reset.



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Gets IQA result with BiSCNMICRFunctionContinuously. After the reading status MF_DATARECEIVE_DONE notification, the reading results are saved in the various parameters of the MF_IQA_RESULT/MF_IQA_RESULT01 structure by specifying the relevant transaction number (ID) in dwTransactionNumber.

BiGetVersion

Acquires a driver version or module version.

Syntax

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

Argument

nDriverType: The driver regarded as the acquisition object is designated.

Constant	Description
DRIVER_TYPE_S9000	TM-S9000 Driver
DRIVER_TYPE_S2000	TM-S2000 Driver

nType: The type of the version to be acquired. One of the following values can be specified.

Constant	Description
VERSION_TYPE_DRIVER	Driver version
VERSION_TYPE_USB	TMUSBDriver version
VERSION_TYPE_MICR	Magnetic waveform analysis module version
VERSION_TYPE_MICR_E13B	Magnetic waveform analysis module version(E13B)
VERSION_TYPE_MICR_CMC7	Magnetic waveform analysis module version(CMC7)
VERSION_TYPE_OCR	OCR recognition module version
VERSION_TYPE_IMAGE	Image processing module version
VERSION_TYPE_IQA	IQA module version
VERSION_TYPE_BARCODE	BARCODE module version
VERSION_TYPE_PH	Communication module version
VERSION_TYPE_MICR_PARSING	MICR parsing module version
VERSION_TYPE_BMPTORASTER	16-tone module version

lpVersion: Sets the address of the structure for storing the version acquisition result.
Refer to ["VERSION_INFO Structure" on page 231](#).

VERSION_INFO Structure

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

lpszDescription: Sets the information of the driver name.

lpszVersion: Sets the version information acquired.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	No data error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Acquires this driver version or the module version used with this driver. It is possible to execute this API before executing BiOpenMonPrinter.

BiESCNEnable

Set so that scanner extended functions can be used.

Before calling BiOpenMonPrinter, it is necessary to enable scanner extended functions by calling this argument.

Syntax

int **BiESCNEnable**(BYTE bStoreType)

Argument

bStoreType: Select a storing method for a cropped image stored using BiESCNStoreImage.

Constant	Value	Description
CROP_STORE_MEMORY	0	Save in memory
CROP_STORE_FILE	1	Save in a file

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_PARAM	-90	Parameter error
ERR_ENABLE	-160	Cannot be used because BiOpenMonPrinter is called



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Secure save table area with BiOpenMonPrinter.

After calling back image data reading, process the image data acquired by the device and save it in the WORK AREA.

Arguments of the scanner extended functions (BiESCN~) can be used.

Note

If this argument is called after calling BiOpenMonPrinter, the scanner extended functions cannot be used and the way of saving a cropped image cannot be changed.

BiESCNGetAutoSize

Acquire the value of capAutoSize.

Syntax

int **BiESCNGetAutoSize**(int nHandle, LPBYTE pCapAutoSize)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pCapAutoSize: Select a memory address to set a capAutoSize value.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

If "1" is selected for capAutoSize, after reading image data, AutoSize processing (cut black part of the image data off) is executed, and the processed image data is saved in the WORK AREA. The width and height of the image data are automatically set to documentWidth and documentHeight.

If "0" is selected for capAutoSize, AutoSize processing and automatic setting for the width and height of the image data are not executed.

BiESCNSetAutoSize

Select the value of capAutoSize.

Syntax

```
int BiESCNSetAutoSize(int nHandle, BYTE bCapAutoSize)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bCapAutoSize: Select a value for a capAutoSize.

Constant	Value	Description
CROP_AUTOSIZE_DISABLE	0	AutoSize processing disabled.
CROP_AUTOSIZE_ENABLE	1	AutoSize processing enabled.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Acquire a value of capAutoSize (AutoSize processing flag).

If an argument other than CROP_AUTOSIZE_ENABLE or CROP_AUTOSIZE_DISABLE is selected, the error (ERR_PARAM) is returned.

The AutoSize processing flag that has been set is used in the next image data reading process (AutoSize processing is not used for the image data that have been already read and saved in the WORK AREA.)

BiESCNGetCutSize

Acquires a value of CutSize.

Syntax

```
int BiESCNGetCutSize (int nHandle, LPWORD pCutSize)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pCutSize: Specify the memory address to store the CutSize value in increments of 0.1 mm.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The CutSize value acquired by this API is the image’s left and right margins to be cropped out.

When other than zero has been specified as the CutSize, the scanned-in image is cropped to the specified size and stored in the work area.

When zero has been specified as the CutSize, cropping operation is not performed.

BiESCNSetCutSize

Sets a value of CutSize.

Syntax

```
int BiESCNSetCutSize (int nHandle, WORD wCutSize)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

wCutSize: Specify the width of left and right margins of an image data to be cropped out within a range of 0 to 1500 in increments of 0.1 mm. The default after executing BiOpenMon-Printer is zero.

pCutSize	Description
0	No cropping operation is performed.
1 to 1500	The image is cropped to the specified size.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Set the value specified by wCutSize to cutSize.

The cutSize is enabled only when capAutoSize has been set to CROP_AUTOSIZE_ENABLE.

Note

The specified cutSize is applied from the next scanning onward. It is not applied to images that has already been scanned and stored in the work area.

If the specified cutSize value is larger than half the width of the paper, the value is rounded down to half the width of scanned-in image.

BiESCNGetRotate

Obtains the information whether the setting for rotating image data 90° has been made to the driver. Also, obtains the value of capRotate.

Syntax

```
int BiESCNGetRotate(int nHandle, LPBYTE pCapRotate)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pCapRotate: Specify the address of the memory in which a return value of capRotate is stored.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Obtains the value of capRotate.

- When CROP_ROTATE_ENABLE is set to capRotate, performs rotation of the image data (rotates the image data 90° to the right or left) after reading it, and stores the edited image data in the WORK AREA.
- When CROP_ROTATE_DISABLE is set to capRotate, does not perform any rotation process.

BiESCNSetRotate

Specifies whether rotate the obtained image data 90° to the driver. Also sets a value to capRotate.



The set Rotate processing flag is not applied until the scanning of the next image data. (The Rotate process is not applied to the image data already scanned and stored in the WORD AREA.)

Syntax

```
int BiESCNSetRotate(int nHandle, BYTE bCapRotate)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bCapRotate: Specify the value for capRotate.

Constant	Value	Description
CROP_ROTATE_ENABLE	1	Processing Rotate enabled
CROP_ROTATE_DISABLE	0	Processing Rotate disabled

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiESCNGetDeSkew

Obtains a threshold value of the skew angle currently set in the driver.

Syntax

```
int BiESCNGetDeSkew(int nHandle, LPWORD lpwAngle)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- lpwAngle: Specify the address of the memory to store the threshold value of the skew angle.
(The unit of the argument is 0.01°)

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiESCNSetDeSkew

Specifies a threshold value for the skew angle to execute DeSkew.

Syntax

int **BiESCNSetDeSkew** (int nHandle, WORD wAngle)

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
wAngle: Specify a threshold value for the skew angle. (unit: 0.01×)The following values can also be set.

Constant	Value	Description
DESKEW_ALL	0	Executes DeSkew.
DESKEW_DISABLE	65535	Does not execute DeSkew.

When a value other than DESKEW_ALL or DESKEW_DISABLE is specified and if the value is other than 1 - 8999, a parameter error occurs.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The driver reads a check and detects the skew angle. If the value exceeds the value set for this function, DeSkew is executed.

If the detected value is smaller than the one set for this function, DeSkew is not executed.

The default value for the driver is 150 (tilt of 1.5°).

Note

Even if DESKEW_All is specified, Deskew is not activated unless the following conditions are met.

- The skew angle must be 10 degrees or less.
- The entire check image must be included within the range where the image can be scanned.

BiESCNGetDocumentSize

Acquire the values of documentWidth and documentHeight.



This API is compatible with the TM-J9000 API.

Syntax

```
int BiESCNGetDocumentSize
    (int nHandle, LPWORD pDocumentWidth,
     LPWORD pDocumentHeight)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

pDocumentWidth: Select a memory address to set a value of the width of the image data (unit: 0.1 mm).

pDocumentHeight: Select a memory address to set a value of the height of the image data (unit: 0.1 mm).

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Acquire the values of documentWidth and documentHeight (the width and height of the image data saved in the WORK AREA) by using the unit of 0.1 mm.

Note

If automatic update by reading the image data or a change with BiESCNSetDocumentSize() is not executed, the default value (width=0, height=0) is acquired.

BiESCNSetDocumentSize

Sets documentWidth and documentHeight of the image data.



- The documentWidth and documentHeight settings made for this API are reflected on CROP_AREA_ENTIRE_IMAGE of bCropArea, as specified with BiESCNDefineCropArea.
- This API is compatible with the TM-J9000.

Syntax

```
int BiESCNSetDocumentSize  
    (int nHandle, WORD wDocumentWidth,  
     WORD wDocumentHeight)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- wDocumentWidth: This specifies the width of the image data (100 to 3000) in units of 0.1 mm.
- wDocumentHeight: This specifies the height of the image data (100 to 3000) in units of 0.1 mm.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiESCNDefineCropArea

Deletes the CropArea registration and the registered CropAreas.

Syntax

```
int BiESCNDefineCropArea
    (int nHandle, BYTE bCropAreaID, WORD wStartX,
     WORD wStartY, WORD wEndX, WORD wEndY)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bCropAreaID: This specifies the CropAreaID (1 to 255) to be registered. This is a BYTE type.

A user can use an ID from "2" to "255" for registering a CropArea.

Constant	Value	Description
CROP_AREA_RESET_ALL	0	All of the registered CropArea data is deleted.
CROP_AREA_ENTIRE_IMAGE	1	Zero is set for the CropArea start X and start Y coordinates, and the values of wdocumentWidth and wdocumentHeight are set for the end X and end Y coordinates.

wStartX: This specifies the start X coordinate of CropArea in units of 0.1 mm.

wStartY: This specifies the start Y coordinate (0 to documentHeight -1) of CropArea in units of 0.1 mm.

wEndX: This specifies the end X coordinate (Value larger than wStartX) of CropArea in units of 0.1 mm.

Constant	Value	Description
CROP_AREA_RIGHT	65535	Sets the value of wdocumentWidth as the end X coordinate

wEndY: This specifies the end Y coordinate (Value larger than wStartY) of CropArea in units of 0.1 mm.

Constant	Value	Description
CROP_AREA_BOTTOM	65535	This sets the value of wdocumentHeight as the end Y coordinate

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset

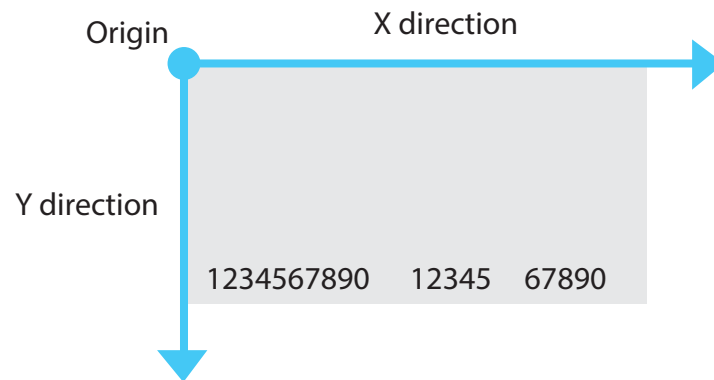


For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Registers the bCropAreaID that sets the CropArea, in the CropArea definition table of the TM-S9000/S2000 API. Any specified bCropAreaID data that has already been registered is overwritten.

When CROP_AREA_RESET_ALL is specified for bCropAreaID, all of the CropAreas in the CropArea definition table are deleted. The CropArea origin is the top-left corner.



Note

- Up to 255 CropAreas can be registered.
- All of the CropAreas registered in the CropArea definition table are deleted each time BiCloseMonPrinter is called. Register the CropAreas after calling BiCloseMonPrinter.
- When the start coordinate is beyond the end coordinate (Start \geq End), ERR_PARAM is returned to the return value.

BiESCNGetMaxCropAreas

Acquires the maximum supported data count that can be registered for CropArea.

Syntax

int **BiESCNGetMaxCropAreas**(int nHandle, LPBYTE pMaxCropAreas)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pMaxCropAreas: This specifies the memory address in which the maximum supported data count that can be registered for CropArea is stored.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

When acquisition is successful, "255" is set in pMaxCropAreas.

BiESCNStoreImage

Crops the CropArea specified with bCropAreaID from the image data in the work area, and then saves it to either a file or memory using the save method specified with BiESCNEnable.

Syntax

```
int BiESCNStoreImage(
    (int nHandle, DWORD dwFileIndex, LPSTR pFileID,
    LPSTR plmageTagData, BYTE bCropAreaID)
```

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- dwFileIndex: This specifies FileIndex (an identifier) of the Crop image to be saved. NULL can also be specified.
- pFileID: This specifies FileID (an identifier) of the Crop image to be saved. This is an LPSTR type. A character string of up to 64 bytes can be specified. NULL can also be specified. Note that none of \ / : , ; * ? " < > | can be used.
- pImageTagData: This specifies ImageTagData (an identifier) of the Crop image to be saved. This is an LPSTR type. A character string of up to 64 bytes can be specified. NULL can also be specified. Note that none of \ / : , ; * ? " < > | can be used.
- bCropAreaID: This specifies the CropAreaID (1 to 255) registered with BiESCNDefineCropArea.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Memory is insufficient
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_DISK_FULL	-170	There is insufficient free space on the disk
ERR_NO_IMAGE	-180	The image data does not exist
ERR_ENTRY_OVER	-190	It is not possible to register more than the maximum allowed number of items.
ERR_CROPAREAID	-200	The specified CropArea does not exist
ERR_EXIST	-210	The specified data has already been saved
ERR_IMAGE_FILEOPEN	-230	Open failure
ERR_IMAGE_UNKNOWN-FORMAT	-240	Format injustice
ERR_IMAGE_FAILED	-250	Image data creation failed
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The cropped image data has the same format as the original image data. All JPEG formats, however, are saved using standard JPEG compression.

If the size of the CropArea exceeds that of the WORK AREA image data, then the excess appears white.

The cropped image data is saved to either memory or a file, using the method specified with BiESCNEnable.

When saving to a file, the file name is formed from the device handle and each identifier (nHandle & original image data name & dwFileIndex & "_" & pFileID & "_" & pImageTagData), and is stored to the folder created by the installer.

Example:

nHandle = 1, *dwFileIndex* = 1, *pFileID* = "AA", *pImageTagData* = "BBB"

<File name>

Original image data name:	File name to be stored in the Crop image save table
Image.jpg	1Image1_AA_BBB.jpg

Crop image save destination for each execution environment

- Windows 2000, Windows XP:
 <TM-S9000II and TM-S9000>
 Documents and Settings\All Users\EPSON\TM-S9000\Temp\
 <TM-S2000II and TM-S2000 >
 Documents and Settings\All Users\EPSON\TM-S2000\Temp\
- Windows Vista, Windows 7, Windows 8:
 <TM-S9000II and TM-S9000>
 ProgramData\EPSON\TM-S9000\Temp\
 <TM-S2000II and TM-S2000 >
 ProgramData\EPSON\TM-S2000\Temp\

Note

- When NULL is specified for all of the identifiers, ERR_PARAM is returned as the return value.
- The pFileID and pImageTagData arguments are not case-sensitive. Therefore, if data with the same name but in a different case already exists, then ERR_EXIST is returned as the return value.

<Example>

When data named pFileID="AAA", pImageTagData="BBB" has already been saved, and you wish to save data named pFileID="aaa", pImageTagData="bbb", ERR_EXIST is returned as the return value.

- When saving data to a file, and a file with the same name as the save file already exists, that file is overwritten.
- All of the CropAreas registered in the CropArea definition table are deleted each time BiCloseMonPrinter is called.

The raster format is not supported for crop image data. When raster format image data is read from a device, ERR_WORKAREA_UNKNOWNFORMAT is returned as the return value.

BiESCNClearImage

Deletes the stored Crop image data.

Syntax

```
int BiESCNClearImage
    (int nHandle, BYTE bFlag, DWORD dwFileIndex,
     LPSTR pFileID, LPSTR pImageTagData)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bFlag: This specifies the deletion method. By using the deletion methods appropriately, it is possible to specify the Crop image data to be deleted. Refer to the explanation.

Macro Definition (Constant)	Value	Description
CROP_CLEAR_ALL_IMAGE	0	Deletes all the Crop image save data
CROP_CLEAR_BY_FILEINDEX	1	Deletes the Crop image data specified by dwFileIndex
CROP_CLEAR_BY_FILEID	2	Deletes the Crop image data specified by pFileID
CROP_CLEAR_BY_IMAGETAGDATA	4	Deletes the Crop image data specified by pImageTagData

dwFileIndex: This specifies FileIndex (an identifier) of the Crop image data to be deleted.

pFileID: This specifies FileID (an identifier) of the Crop image data to be deleted. Note that none of \ / : , ; * ? " < > | can be used.

pImageTagData: This specifies ImageTagData (an identifier) of the Crop image data to be deleted. Note that none of \ / : , ; * ? " < > | can be used.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_NOT_FOUND	-220	No data error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

By using `bFlag` appropriately, it is possible to specify the Crop image data to be deleted.

<Example>

When the Crop image data specified with `dwFileIndex` and `pImageTagData` is to be deleted

bFlag = CROP_CLEAR_BY_FILEINDEX + CROP_CLEAR_BY_IMAGETAGDATA;

Note

All of the CropAreas registered in the CropArea definition table are deleted each time `BiCloseMonPrinter` is called.

BiESCNRetrievelmage

Acquires the Crop image data that is saved to memory or a file.

Syntax

```
int BiESCNRetrievelmage(int nHandle, DWORD dwFileIndex,
                          LPSTR pFileID, LPSTR plmageTagData,
                          LPDWORD plmageSize, LPBYTE plmageData)
```

Argument

nHandle:	Handle obtained with the return value of the communication initialization API.
dwFileIndex:	This specifies FileIndex (an identifier) of the Crop image data to be acquired. While NULL can be specified, doing so prevents a search being made for the FileIndex.
pFileID:	This specifies FileID (an identifier) of the Crop image data to be acquired. Note that none of \ / : , ; * ? " < > can be used. While NULL can be specified, doing so prevents a search being made for the FileID.
pImageTagData:	This specifies ImageTagData (an identifier) of the Crop image data to be acquired. Note that none of \ / : , ; * ? " < > can be used. While NULL can be specified, doing so prevents a search being made for the ImageTagData.
pImageSize:	Specifies the size of the memory in which the size of the acquired Crop image data is set. After calling this API, the size of the actually acquired CROP image data is returned. When there is insufficient capacity, the required size is returned together with the return value.
pImageData:	Specifies the memory address where the Crop image data is set.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_BUFFER_OVER_FLOW	-140	ERR_BUFFER_OVER_FLOW
ERR_NOT_FOUND	-220	No data error
ERR_IMAGE_FILEOPEN	-230	Open failure
ERR_IMAGE_FILEREAD	-290	Read of the image data file failed
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Acquires Crop image data, corresponding to the specified identifier, from the Crop image save table (memory or file) of the TM-S9000/S2000 API.

When there are multiple saved items of Crop image data corresponding to the specified identifier, only the first such item to be found is acquired.

Example:

- [1] When dwFileIndex = 1, pFileID = NULL, pImageTagData = NULL are specified
- [2] When dwFileIndex = 2, pFileID = NULL, pImageTagData = NULL are specified
- [3] When dwFileIndex = NULL, pFileID = "B", pImageTagData = NULL are specified

dwFileIndex	pFileID	pImageTagData	Image Data	
1	"A"	NULL	← [1]
1	"B"	NULL	← [2]
1	"C"	NULL	
2	"A"	NULL	← [3]
2	"B"	NULL	
2	"C"	NULL	
3	"A"	"A"	
3	"B"	NULL	

Save order



Note

- All of the CropAreas registered in the CropArea definition table are deleted each time BiCloseMonPrinter is called.
- When NULL is specified for all of the identifiers, ERR_PARAM is returned as the return value.

BiESCNGetRemainingImages

Acquires the CropArea remaining count that can be registered.

Syntax

```
int BiESCNGetRemainingImages(int nHandle, LPBYTE pRemainingImages)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pRemainingImages: This specifies the memory address where the CropArea remaining count that can be registered is set.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Unsupported
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The maximum remaining count that can be acquired is 255.

BiSetBehaviorToScnResult

Sets the behavior to the result for scanning.

Syntax

```
int BiSetBehaviorToScnResult(int nHandle, BYTE bEject
                               , BYTE bStamp, BYTE bNextCheck)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bEject: Sets the ejection method of check sheets.

Constant	Description
MF_EJECT_MAIN_POCKET	Ejects into the main pocket.
MF_EJECT_SUB_POCKET	Ejects into the sub pocket.
MF_EJECT_NOEJECT	Does not eject.

bStamp: Sets whether to enable a franker.

Constant	Description
MF_STAMP_DISABLE	Does not perform franking.
MF_STAMP_ENABLE	Performs franking.



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.

bNextCheck: Sets whether to feed the next check sheet when ejecting the current one.

Constant	Description
MF_PROCESS_CONTINUE_OVERLAP	Starts the next reading process while ejecting documents.
MF_PROCESS_CONTINUE_NOOVERLAP	Starts the next reading process after ejecting documents.
MF_PROCESS_CONTINUE_CANCEL	Cancels the next reading process.

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	Unsupported



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

This API specifies the behavior of the check sheet loaded in a situation where MF_ACTIVATE_MODE_CONFIRMATION is specified for ActivationMode in the PROCESS setting.

BiSetNumberOfDocuments

The number of checks to be loaded is specified in the continuous scan execution API.

Syntax

```
int BiSetNumberOfDocuments(int nHandle, BYTE bNumber)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
bNumber: Specifies the number (0 to 100) of documents to read. The initial value is 0.

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	A scan is being executed.
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- If 0 is specified for bNumber, all the documents set in the ASF (Auto Sheet Feeder) are read.
- If 1 or larger is specified for bNumber, reading ends when reading of the specified number of documents is complete. The setting made using this API is valid until BiCloseMonPrinter is run.
- If reading of the documents set in the ASF is completed before reading the number of documents specified using this API, the error code (ERR_LESS_CHECKS) is sent, indicating that the specified number of documents have not been read to SubStatus of the reading status MF_FUNCTION_DONE.



This API can be used to set the number of checks to be loaded only when MF_ACTIVATE_MODE_HIGH_SPEED is set for ActivationMode in the PROCESS setting that is specified with the continuous scan execution API.

If MF_ACTIVATE_MODE_CONFIRMATION is set, the setting specified with this API will be ignored. Application can control number of scanning document using ["BiSetBehaviorToScnResult" on page 254](#).

BiSelectErrorEjectAtContinuously

Sets the paper eject method used for when an error is detected while reading check paper continuously.



- This API is compatible with the TM-J9000 API.
- For the TM-S9000II, the TM-S2000II, the TM-S9000 and the TM-S2000, please use MF_PROCESS01 structure for setting paper eject method.

Syntax

```
int BiSelectErrorEjectAtContinuously
    (int nHandle , DWORD dwError
    , DWORD dwSetting)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwError: Specifies the type of error for setting the paper eject method. The following values can be set individually or as a logical sum.

Constant	Description
EPS_BI_ERROREJECT_ERROR_ALL	All four of the following: double feed, magnetic waveform detection error, unrecognized character detection error, and noise error.
EPS_BI_ERROREJECT_ERROR_DOUBLEFEED	Double feed.
EPS_BI_ERROREJECT_ERROR_NODATA	Magnetic waveform detection error.
EPS_BI_ERROREJECT_ERROR_BADDATA	Unrecognized character detection error.
EPS_BI_ERROREJECT_ERROR_NOISE	Noise error.

dwSetting: Specifies the paper eject method. One of the following values can be set.

Constant	Description
EPS_BI_ERROREJECT_SETTING_DISCHARGE	Paper ejected to main pocket.
EPS_BI_ERROREJECT_SETTING_RELEASE	Paper released at slip position.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

- If the process that can be set for this function to be performed when an error occurs continues for errorSelect in BiMICRSelectDataHandling, the settings for this function are ignored.
- When BiSCNMICRFunctionContinuously is executed, the paper eject method that can be set for this function to be performed when an error occurs will follow the method set for this function, and the settings for wErrorEject in MF_BASE01 are ignored.
- On the other hand, the paper eject method for when the same error occurs in BiSCNMICRFunction and BiSCNMICRFunctionPostPrint follow the settings of wErrorEject in MF_BASE01, and the settings in this function are ignored.
- The order of priority for error handling in BiSCNMICRFunctionContinuously is as follows.
 1. Setting of BiMICRSelectDataHandling
 2. Setting of this API

BiSelectJamDetect

Sets the paper detection sensor used for paper jam detection.



This API is compatible with the TM-J9000 API.

Syntax

```
int BiSelectJamDetect(int nHandle, DWORD dwJamDetect)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

dwJamDetect: Specifies the sensor for jam detect. One of the following values can be set.

Constant	Description
EPS_BI_JAM_DETECT_USE_UPPER_LEFT_SENSOR	An upper left paper sensor is used for paper jam detection. Although paper jam detection accuracy is high, when the check paper with which the upper left broke is scanned, it becomes a paper jam error.
EPS_BI_JAM_DETECT_NOT_USE_UPPER_LEFT_SENSOR	An upper left paper sensor is not used for paper jam detection. Although paper jam detection accuracy is a little inferior, the check paper with which the upper left broke can be scanned.

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 be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiMICRGetStatus

Acquires the MICR status.



This API is compatible with the TM-J9000 API.

Syntax

```
int BiMICRGetStatus (int nHandle, LPBYTE pStatus)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
pStatus: Specifies the memory address that acquires the MICR status.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_OFFLINE	-110	It was opened in the offline state, so it cannot be used until the online state is recovered.
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Refer to ["MICR Status" on page 369](#) regarding the acquired MICR status.

BiConfirmBufferedData

Acquires the number of document data that are being stored in the driver after scan processing is completed.



This API can be called inside the MF_DATARECEIVE_DONE callback that is issued when a scan is activated.

Syntax

```
int BiConfirmBufferedData(int nHandle, LPDWORD lpdwBufferedCount)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

lpdwBufferedCount:

Specifies the memory address that is set for the number of document data that are being stored in the driver after scan processing is completed.

Return value


Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiMICRCleaning

Cleans the MICR mechanism.



If this function is called, the mechanism waits for the cleaning sheet to be inserted. Insert the cleaning sheet and carry out cleaning of the mechanism.

Syntax


```
int BiMICRCleaning(int nHandle)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BlinkHeadCleaning

Ink-jet head cleaning is executed.

The API finishes at the time when the cleaning command is sent, instead of waiting until the completion of head cleaning.

Syntax

```
int BlinkHeadCleaning(int nHandle)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiOpenDrawer

A drawer connected to the DKC (Drawer Kick Connector) is opened. Not supported in the TM-S2000II and the TM-S2000.

Syntax

```
int BiOpenDrawer(int nHandle, BYTE byDrawer, BYTE byPulse)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

byDrawer: The drawer to be opened is selected.

Constant	Description
EPS_BI_DRAWER_1	Drawer 1 (3rd Drawer Kick Connector pin) will be opened.

byPulse: Specifies the time when the drawer kick signal is on.

Constant	Description
EPS_BI_PLUSE_100	Signal for 100 milliseconds
EPS_BI_PLUSE_200	Signal for 200 milliseconds
EPS_BI_PLUSE_300	Signal for 300 milliseconds
EPS_BI_PLUSE_400	Signal for 400 milliseconds
EPS_BI_PLUSE_500	Signal for 500 milliseconds
EPS_BI_PLUSE_600	Signal for 600 milliseconds
EPS_BI_PLUSE_700	Signal for 700 milliseconds
EPS_BI_PLUSE_800	Signal for 800 milliseconds

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiRingBuzzer

Sounds a buzzer.

Syntax

```
int BiRingBuzzer(int nHandle, BYTE bTone, BYTE bCount
, WORD wOnTime, WORD wOffTime)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

bTone: Specifies the buzzer tone.

Constant	Description
MF_BUZZER_TONE_HIGH	High-pitched sound
MF_BUZZER_TONE_MIDDLE	Middle-pitched sound
MF_BUZZER_TONE_LOW	Low-pitched sound

bCount: Specifies the number of buzzers. The valid setting value is 1 to 8. When a value exceeding 8 is set, it will be rounded to 8.

wOnTime: Specifies the buzzer tone duration. The valid setting value is 100 to 800 (unit: mm). The specification must be made in 100 mm unit. A value less than 100 is rounded off to the nearest hundred. When a value exceeding 800 is set, it will be rounded to 800.

wOffTime: Specifies the off time of buzzer tone. The valid setting value is 100 to 800 (unit: mm). The setting must be made in 100 mm unit. A value less than 100 is rounded off to the nearest hundred. A value outside the valid range is rounded to the approximate value that can be specified. 0 (zero) can be set as well as valid values for the setting. When 0 is specified, the buzzer keeps sounding during the duration calculated by the following expression: Ring duration x Number of buzzers. No value is rounded to 0.

Return value

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



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Setting all the parameters of bTone, bCount, wOnTime and wOffTime to 0 (zero) stops the buzzer.

This also applies to the buzzer specified by the BASE setting ([page 277](#)).

BiGetCounter

Acquires the maintenance counter value.

Syntax

```
int BiGetCounter(int nHandle, WORD readno, LPDWORD readcounter)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
 readno: Specifies the number of the acquired maintenance counter.
 readcounter: Returns the maintenance counter.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

4

Description

Refer to ["Maintenance Counter" on page 368](#) regarding the device counter and the acquired maintenance counters.

The maintenance counters include those that can be reset by the user and those that are not reset and count.

BiResetCounter

Acquires the maintenance counter value.

Syntax

```
int BiResetCounter(int nHandle, WORD writeno)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
writeno: Specifies the number of the reset maintenance counter.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_OFFLINE	-110	It was opened in the offline state, so it cannot be used until the online state is recovered.
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The maintenance counters include those that can be reset by the user and those that are not reset and count. Refer to ["Maintenance Counter" on page 368](#).

BiLoadAPISettings

Loads the ["API settings file" on page 391](#) to set the driver.

Syntax

```
int BiLoadAPISettings(int nHandle, char* pszSettingsFilePath)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

pszSettingsFilePath:

Specifies the path to the ["API settings file" on page 391](#).

Value	Description
Any string	The full path to the file must be specified.
APISETTINGS_DEFAULT	If APISETTINGS_DEFAULT is specified, it is assumed that APISettings.ini located in the same folder as the driver is specified.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_PARAM	-90	Parameter error
ERR_NOT_FOUND	-220	File not found
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetConfigure

Set enable/disable for 2nd receipt, machine tape, etc.

Syntax

```
int BiSetConfigure(int nHandle, unsigned long ulConfigID
, unsigned long ulConfigType)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

ulConfigID: Specify the function to set.

Value	Description
MF_CONFIG_ID_2ND_RECEIPT (Not supported in the TM-S2000II and the TM-S2000)	Settings for the 2nd receipt function
MF_CONFIG_ID_SCAN_LENGTH	Settings for scan paper length
MF_CONFIG_ID_MICR_RECOG	Settings for MICR type
MF_CONFIG_ID_ENDORSE_PRINT_POSITION	Settings for endorsement print position

ulConfigType: Settings for functions specified in ConfigID

Function	Value	Description
2nd Receipt (Not supported in the TM-S2000II and the TM-S2000)	MF_CONFIG_TYPE_2ND_RECEIPT_DISABLE	DISABLE
	MF_CONFIG_TYPE_2ND_RECEIPT_ENABLE	ENABLE
Machine tape	MF_CONFIG_TYPE_SCAN_LENGTH_NORMAL	Normal paper length
	MF_CONFIG_TYPE_SCAN_LENGTH_EXPAND	Expanded paper length (for machine tape)
MICR type	MF_CONFIG_TYPE_MICR_RECOG_ANSI	ANSI
	MF_CONFIG_TYPE_MICR_RECOG_CCCC	C&CCC

Function	Value	Description
Endorsement print position	MF_CONFIG_TYPE_ENDORSE_PRINT_POSITION_NORMAL	Print starting position is based on the leading edge of the paper
	MF_CONFIG_TYPE_ENDORSE_PRINT_POSITION_PAYEE_ENDORSE	<p>Measure the length of the paper and move the printing start position to the right.</p> <p>This allows the right side of the print data to be set along the right side of the paper when printing.</p> <p>This function does not right-align the print data.</p> <p>By combining with 90-degrees rotation printing, you can perform Payee endorsement printing.</p> <p>If the size of the print data is greater than the printable area for this product:</p> <ul style="list-style-type: none"> • Move the printing start position to the left side of the printable area. • Print data exceeding the printable area will not be printed. <p>Refer to this product's Technical Reference Guide for the size of the printable area.</p>

Return value


Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

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.



- Use this API when a recoverable error occurs. Refer to ["Device Status" on page 364](#) for details on recoverable errors.
- Cancels print jobs when this is called while printing.

Syntax


```
int BiResetPrinter(int nHandle)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	A timeout error occurred
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiCancelError

Restores recoverable device errors.

Syntax

```
int BiCancelError(int nHandle)
```

Argument

nHandle: Handle obtained with the returned value of the communication initialization API.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Invalid handle value
ERR_TIMEOUT	-70	A timeout error occurred
ERR_ACCESS	-80	Reading/writing with the device is not possible (printing in progress)



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiGetType

Acquires the device type ID.

Syntax

```
int BiGetType (int nHandle, LPBYTE typeID, LPBYTE font
               , LPBYTE exrom, LPBYTE special)
```

Argument

nHandle:	Handle obtained with the return value of the communication initialization API.
typeID:	Returns the device type ID.
font:	Not used
exrom:	Not used
special:	Not used

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_OFFLINE	-110	It was opened in the offline state, so it cannot be used until the online state is recovered.
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Refer to ["Type ID" on page 384](#).

BiGetOfflineCode

Acquires the cause of the device to go offline.



- This API is compatible with the TM-J9000/TM-S1000 API.
- This API acquires a value that is kept compatible with the offline factor for the previous models.
- Use "[BiGetOfflineCodeByIndex](#)" on page 274 to acquire a device-specific offline factor.

Syntax

```
int BiGetOfflineCode (int nHandle, LPBYTE lpOfflinecode)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

lpOfflinecode: Returns the 5-byte value indicating the reason for the device going offline.



When the cause of the device going offline is acquired while the device is online, zero will be written to the leading byte of lpOfflinecode.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_ACCESS	-80	Reading/writing with the device is not possible
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset



For a list of return values and troubleshooting actions, refer to "[Return value](#)" on page 61.

Description

Refer to "[BiGetOfflineCode](#)" on page 377.

BiGetOfflineCodeByIndex

Acquires the cause of the device to go offline from the leading byte.



- The cause for going offline is in the device set in the Status section of the API settings file. For more information, refer to ["API settings file" on page 391](#).
- Data obtainable by this API is a device-specific value and may differ from the data for the previous models.

Syntax

```
int BiGetOfflineCodeByIndex
    (int nHandle, int nIndex, LPBYTE lpOfflinecode)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.
 nIndex: Specifies bytes (1 to 10) of the acquired cause of the device going offline.
 lpOfflinecode: Returns the 5-byte value indicating the reason for the device going offline.



When the cause of the device going offline is acquired while the device is online, zero will be written to the leading byte of lpOfflinecode.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot be used because the other API is being executed
ERR_RESET	-400	Cannot be used because the device is being reset




For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

Refer to ["Offline Code" on page 370](#).

BiSCNGetClumpStatus

Acquires the clamp status of cut sheets.



This API is compatible with the TM-J9000 API. Since there is no clamp status information in the TM-S9000II, the TM-S2000II, the TM-S9000 and the TM-S2000, "01000000(0x40)" is always returned.

Syntax


int **BiSCNGetClumpStatus**(int nHandle, LPBYTE pStatus)

Argument

- nHandle: Handle obtained with the return value of the communication initialization API.
- pStatus: The clamp status(01000000 (0x40)) is set.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	The handle value that specifies the device is incorrect
ERR_PARAM	-90	Parameter error



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

BiSetPaperThickness

This function does not work with the TM-S9000II, the TM-S2000II, the TM-S9000 and the TM-S2000, because this function is just for compatibility with TM-S1000 API.

Syntax

```
int BiSetPaperThickness(int nHandle, WORD wThreshold)
```

Argument

nHandle: Handle obtained with the return value of the communication initialization API.

wThreshold: Specifies the double feed threshold. The valid specification range is 1 to 40 (0.01 mm to 0.40 mm). When "0" is specified, the threshold value when calling BiOpenMonPrinter is applied.

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 be used because the other API is being executed



For a list of return values and troubleshooting actions, refer to ["Return value" on page 61](#).

Description

The threshold set using this API ignores the thresholds defined for each paper type using the MF_PROCESS01 structure in advance (See ["MF_PROCESS01" on page 298](#)) and applies to all the paper types.

The thresholds set using this API are valid until BiCloseMonPrinter is invoked.

Structures

MF_BASE01

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

int iSize

Sets the size of the structure.

int iVersion

This is the structure version. Always specify MF_BASE_VERSION01. The MF_BASE_VERSION01 value differs for each driver version. The application uses the MF_BASE_VERSION01 for the supplied header file.

int iRet

The return value for this function is set. Regardless of its being synchronous or asynchronous, this is set when the function ends. ERR_PARAM is returned when the API is called by MF_EXEC / MF_xxxx_RETRANS without the MF_BASE01 structure being specified and when the API is called by MF_EXEC / MF_xxxx_RETRANS when only the MF_BASE01 structure is specified. Refer to ["MF_BASE01.iRet" on page 111](#) for details.

DWORD dwNotifyType

When MF_EXEC is specified in the 3rd parameter for this API, it indicates the operating method for the API. This API can be run synchronously or asynchronously. When the API is run synchronously, MF_BASE_MESSAGE_NO_MESSAGE is used, and when it is run asynchronously, another value is used. BiSCNMICRFunctionPostPrint and BiSCNMICRFunctionContinuously operate asynchronously regardless of the setting of dwNotifyType.

It is run in the condition where dwNotifyType is specified to MF_BASE_MESSAGE_NO_MESSAGE, when MF_xxxx_RETRANS is specified in the 3rd parameter for this API.

dwNotifyType (Constant)	Description
MF_BASE_MESSAGE_NO_MESSAGE	When MF_EXEC / MF_xxxx_RETRANS is used in the 3rd parameter and it is called, the function does not generate a new thread, and control returns after completion of the operation specified by the structure. (It is run synchronously.)
MF_BASE_MESSAGE_EVENT	When MF_EXEC is used in the 3rd parameter and it is called, the function runs a separate thread and immediately returns control. When the thread is started successfully, SUCCESS is returned. When there is a parameter error, invalid handle value or some other failure to meet prerequisite conditions for starting a thread, and error is returned according to the situation. When the value returned is SUCCESS, there is notification of completion of the function by setting the event handle specified by uNotifyHandle to signal status. Notification is carried out by the SetEvent function. The return value is set in iRet. When processing is canceled by BiSCNMICRCancelFunction, and event is generated. In this case, ERR_ABORT is set in iRet.
MF_BASE_MESSAGE_HWND	When MF_EXEC is used in the 3rd parameter and it is called, the function runs a separate thread and immediately returns control. When the thread is started successfully, SUCCESS is returned. When there is a parameter error, invalid handle value or some other failure to meet prerequisite conditions for starting a thread, an error is returned according to the situation. When the value returned is SUCCESS, there is notification of completion of the function in the window handle specified by uNotifyHandle. Notification is done by PostMessage, and regardless of whether it is normal or abnormal, WM_MF_DONE is sent when the function is completed. The return value is set in IParam. The return value is the same as for the API (SUCCESS, ERR_ACCESS, etc.). In addition, the same return value is set in iRet. When processing is canceled by BiSCNMICRCancelFunction, there is a WM_MF_DONE notification. In this case, ERR_ABORT is set IParam and in iRet.
MF_BASE_MESSAGE_BUTTON_CLICK	When MF_EXEC is used in the 3rd parameter and it is called, the function runs a separate thread and immediately returns control. When the thread is started successfully, SUCCESS is returned. When there is a parameter error, invalid handle value or some other failure to meet prerequisite conditions for starting a thread, an error is returned according to the situation. When the value returned is SUCCESS, there is notification of completion of the function in the button control window handle specified by uNotifyHandle using a PostMessage with WM_COMMAND (BN_CLICKED). There is WM_COMMAND (BN_CLICKED) notification when the function is completed regardless of whether it is normal or abnormal. The return value is obtained by referencing iRet. When processing is canceled by BiSCNMICRCancelFunction, there is notification with the same message. In this case, ERR_ABORT is set in iRet.

dwTimeout

The time to wait before paper insertion is specified in seconds. The values that can be specified are 0 to 300 (five minutes), and when 0 is specified, there is no timeout. When there is a timeout, `ERR_PAPER_INSERT_TIMEOUT` is returned. Timeouts other than paper insertion cannot be specified. However, it is possible for the application to specify a window handle valid for `hProgressWnd` and manage timeout times while watching the status of message notifications using `WM_MF_PROGRESS`. When independent timeouts are executed, running `BiSCNMICRCancelFunction` is recommended. (This is not done, `BiSCNMICRFunctionContinuously` will continue to run, and during that time a port will be occupied.) If `BiSCNMICRCancelFunction` is run, the `BiSCNMICRFunctionContinuously` process will be interrupted, and a port will be opened.

uNotifyHandle

This sets the pointer for the event handle or the window handle for notification of completion. Since API automatically generates and discards the event handle automatically, this is not done from the application.

hProgressWnd

This sets a window handle for notification of progress in processing. When each data block is read during scanning, there is notification of what percentage of the total amount has been read. In the message, `MF_PHASE_INIT`, `MF_PHASE_SCAN`, `MF_PHASE_MICR`, `MF_PHASE_PRINT` or `MF_PHASE_EXIT` is set in `wParam` by `WM_MF_PROGRESS`. The percentage value (0-100) is set in `lParam`. However, even if `lParam` is 100%, it does not mean that this function is complete. Completion of the function is always performed by `WM_MF_DONE` notification. There is no problem if the window handle set by `hProgressWnd` and `uNotifyHandle` are the same. When the progress notifications unnecessary, this parameter is set 0. See ["Progress Status Message List" on page 281](#) for each of the phases, the messages sent in each phase, and the message content acquisition macros.

WORD wErrorEject

This member is used for compatibility. Set the paper ejection method with ["MF_PROCESS01" on page 298](#).

BYTE bBuzzerHz[MF_BUZZER_TYPE_MAX]

This sets the frequency of the buzzer for MICR reading. This member is made up of three array elements, and `MF_BUZZER_TYPE_SUCCESS` is for the case where reading is successful, `MF_BUZZER_TYPE_ERROR` for the case where read error is occurred, and `MF_BUZZER_TYPE_WFEED` for the case where sending of multiple sheets is detected. One out of `MF_BUZZER_HZ_4000`, `MF_BUZZER_HZ_440`, and `MF_BUZZER_HZ_880` is specified for each of the array elements.

For example, if `MF_BASE01`. `bBuzzerHz[MF_BUZZER_TYPE_SUCCESS] = MF_BUZZER_HZ_440`; is set, a 440 Hz buzzer sounds when MICR reading is successful. Number of times the buzzer sounds can be set using `bBuzzerCount`. If a value other than the valid values is specified, `ERR_PARAM` is returned. Also, with `PhotoID`, this value is ignored.

bBuzzerHz (Constant)	Description
<code>MF_BUZZER_HZ_440</code>	Sounds the 440 Hz buzzer.
<code>MF_BUZZER_HZ_880</code>	Sounds the 880 Hz buzzer.
<code>MF_BUZZER_HZ_4000</code>	Sounds the 4000 Hz buzzer.

BYTE bBuzzerCount[MF_BUZZER_TYPE_MAX]

This sets the number of times the buzzer corresponding to each element in bBuzzerHz sounds.

The values that can be specified are 0 to 3.

For example, if

```
MF_BASE01.bBuzzerHz[MF_BUZZER_TYPE_ERROR] = MF_BUZZER_HZ_4000;
```

```
MF_BASE01.bBuzzerCount [MF_BUZZER_TYPE_ERROR] = 2;
```

is set, a 4000 Hz buzzer sounds two times when there is a failure in MICR reading. If a value other than the valid values is specified, ERR_PARAM is returned.

BYTE bUseNVMemory

Not used.

char cPortName[256]

This sets the port name. When ERR_HANDLE is occurred, nothing is set. (zero clear)

WORD wSuccessEject

This member is used for compatibility. Set the paper ejection method with ["MF_PROCESS01" on page 298](#).

Default Values of the MF_BASE01 Structure

Member of MF_BASE01 Structure	Default Value
MF_BASE01.dwNotifyType	MF_BASE_MESSAGE_HWND
MF_BASE01.dwTimeout	MF_BASE_TIMEOUT_DEFAULT
MF_BASE01.uNotifyHandle.hNotifyWnd	0
MF_BASE01.hProgressWnd	0
MF_BASE01.wErrorEject	MF_EXIT_ERROR_RELEASE
MF_BASE01.bBuzzerHz	MF_BUZZER_HZ_4000
MF_BASE01.bBuzzerCount	MF_BUZZER_DISABLE
MF_BASE01.bUseNVMemory	MF_BASE_NVMEMORY_NOT_USE
MF_BASE01.wSuccessEject	MF_EXIT_SUCCESS_DISCHARGE

Progress Status Message List

Message	wParam	lParam	Description
WM_MF_DONE	-	Return value	BiSCNMICRFunction was completed.
WM_MF_PROGRESS	MF_PHASE_INIT	MF_PROGRESS_START	Initialization was started.
		MF_PROGRESS_WAIT_PAPER	Waiting for paper insertion. This message will not be issued if paper is already set.
		MF_PROGRESS_CLUMP_PAPER	The paper is clamped and being transferred.
		MF_PROGRESS_PAPER_PILED	Double feed was detected.
		MF_PROGRESS_DONE	The device completed the SCAN/MICR data read operation.
	MF_PHASE_SCAN	Front data read progress (%)	MF_SCAN_FACE_FRONT is set for the lower BYTE of the higher WORD, whereas the percentage is set for the lower BYTE of the lower WORD. 0% is notified at the start, and 100% is notified at the end.
		Back data read progress (%)	MF_SCAN_FACE_BACK is set for the lower BYTE of the higher WORD, whereas the percentage is set for the lower BYTE of the lower WORD. 0% is notified at the start, and 100% is notified at the end.
	MF_PHASE_MICR	MF_PROGRESS_START	MIC_OCR reading was started.
		MF_PROGRESS_DONE	MIC_OCR reading was completed.
	MF_PHASE_BARCODE	Front data read progress (%)	MF_SCAN_FACE_FRONT is set for the lower BYTE of the higher WORD, whereas the percentage is set for the lower BYTE of the lower WORD. 0% is notified at the start, and 100% is notified at the end.
		Back data read progress (%)	MF_SCAN_FACE_BACK is set for the lower BYTE of the higher WORD, whereas the percentage is set for the lower BYTE of the lower WORD. 0% is notified at the start, and 100% is notified at the end.
WM_MF_PROGRESS	MF_PHASE_PRINT	MF_PROGRESS_START	The transaction printing was started.
		MF_PROGRESS_DONE	The transaction printing was completed.
	MF_PHASE_EXIT	MF_PROGRESS_START	Postprocessing, including paper ejection, was started.
		MF_PROGRESS_DONE	Postprocessing, including paper ejection, was completed.

- In case of the WM_MF_PROGRESS message, the read unit number is set for LOBYTE of HIWORD of wParam. The phase number (MF_PHASE_INIT, etc.) is set for LOBYTE of LOWORD. The read unit number for check sheet is EPS_BI_SCN_UNIT_CHECKPAPER.
- The content of wParam and lParam in the WM_MF_PROGRESS message received can be retrieved using the supplied macros below.

Macro	Description
MF_MACRO_GETPHASE(wParam)	Macro for retrieving the phase number
MF_MACRO_GETFACE(lParam)	Macro for retrieving the image face to be read
MF_MACRO_PERCENT(lParam)	Macro for retrieving the percent of image read

MF_SCAN

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    int iRet;                                 OUT
    WORD wImageID;                            IN
    short sResolution;                        IN
    BYTE bAddInfoDataSize;                   IN
    LPBYTE pAddInfoData;                     IN
    BYTE bStatus;                             OUT
    BYTE bDetail;                             OUT
    DWORD dwXSize;                           OUT
    DWORD dwYSize;                           OUT
    DWORD dwScanSize;                         OUT
    LPBYTE lpbScanData;                       OUT
} MF_SCAN, *LPMF_SCAN;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_SCAN_VERSION. Since the MF_SCAN_VERSION value is different for each driver version, the application must always use the supplied header file.

int iRet

The results for scanning are set. Refer to ["MF_SCAN.iRet" on page 111](#) for details.

WORD wImageID

This specifies the ID for an image that is read.

short sResolution

This specifies the resolution for an image that is read. Select one from the following preset parameters. Values other than these, return ERR_PARAM.

sResolution (Constant)	Description
MF_SCAN_DPI_DEFAULT	Scans at the default resolution for the device. (200 DPI)
MF_SCAN_DPI_200	Scans at 200 DPI.
MF_SCAN_DPI_300	Scans at 300 DPI.

Each of the following values can be specified independently only if EPS_BI_SCN_UNIT_CHECKPAPER is set to the unit when this structure is used. (Otherwise, it is considered to be not defined.)

sResolution (Constant)	Description
MF_SCAN_DPI_100	Scans at 100 DPI.
MF_SCAN_DPI_120	Scans at 120 DPI.
MF_SCAN_DPI_240	Scans at 240 DPI.



Small sized characters on check sheet will become hard to read when MF_SCAN_DPI_100 or MF_SCAN_DPI_120 is selected.

The following value can be specified independently only if EPS_BI_SCN_UNIT_CARD is set to the unit when this structure is used. (Otherwise, it is considered to be not defined.)

sResolution (Constant)	Description
MF_SCAN_DPI_600	Scans at 600 DPI.

BYTE bAddInfoDataSize; (IN)

This specifies the size of additional character data. When this value is 0, no characters are added even in cases when bAddInfoData is not NULL. The maximum value that can be specified is 255.

LPBYTE pAddInfoData; (IN)

This specifies the memory address where characters to be added are set. Even when this value is not NULL, no characters are added in cases when bAddInfoDataSize is 0.

BYTE bStatus

This sets the status when the scan is completed.

Bit	Status Contents	ON / OFF	Value	Status
0	---	---	0x0000	Reserved (Fixed to 0)
1	---	---	0x0000	Reserved (Fixed to 0)
2	---	---	0x0000	Reserved (Fixed to 0)
3	Scanning face	ON	0x0008	Back
		OFF	0x0000	Front
4	Rescanned	ON	0x0010	Disabled
		OFF	0x0000	-
5	Scanning results	ON	0x0020	Abnormal end
		OFF	0x0000	Success
6	Scanning data overflow	ON	0x0040	No overflow
		OFF	0x0000	Not (Fixed)
7	Scanning data translation error	ON	0x0080	Ends with an error
		OFF	0x0000	No error



For the status when scan is complete, the value when readout of the first block of the image data is successful is stored. For this reason, if an error occurs before reading of the first block of the image data is successful, the value will not be set as the status when scan is complete.

BYTE bDetail

This sets the detailed status when the scan is completed.

Value	Information
40h	Success
41h	No image reading result
42h	Cancellation of paper insertion waiting (Executing BiSCNMICRCancelFunction)
44h	Cancellation of image reading due to feed error
45h	Occurrence of double feed or insertion orientation error during image reading
46h	Detection of sending error before reading starts
48h	Detection of paper length error during reading



For the detailed status when scan is complete, the detailed status that has been stored in the first block of the image data that is read from the device, is stored. For this reason, if an error occurs before the first block of the image data is successfully read, the value will not be set as the status when scan is complete.

DWORD dwXSize

This sets the number of dots in the X direction for image reading.

DWORD dwYSize

This sets the number of dots in the Y direction for image reading.

DWORD dwScanSize

This sets the size of the image read.

LPBYTE lpbScanData

This sets the address of the image read. API automatically reserves and discards this memory.

Therefore, the application must discard it in a timely manner.

When discarding this memory on the application-side, specify this memory address for the Windows API GlobalFree function

Default Values of the MF_SCAN Structure

Member of MF_SCAN Structure	Default Value
MF_SCAN.wImageID	1
MF_SCAN.sResolution	MF_SCAN_DPI_DEFAULT
MF_SCAN.bAddInfoDataSize	0
MF_SCAN.pAddInfoData	NULL
MF_SCAN.bStatus	0
MF_SCAN.bDetail	0
MF_SCAN.dwXSize	0
MF_SCAN.dwYSize	0
MF_SCAN.dwScanSize	0
MF_SCAN.lpbScanData	NULL

MF_MICR01

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    int iRet;                                 OUT
    BYTE bFont;                               IN
    BYTE bMicOcrSelect;                      IN
    BOOL blParsing;                          IN
    BYTE bStatus;                            OUT
    BYTE bDetail;                            OUT
    CHAR szMicStr[MF_MICR_CHAR_MAX];         OUT
    MF_OCR_RELIABLE_INFO stOcrReliableInfo[MF_MICR_CHAR_MAX]; OUT
    CHAR szAccountNumber[MF_MICR_CHAR_MAX];  OUT
    CHAR szAmount[MF_MICR_CHAR_MAX];        OUT
    CHAR szBankNumber[MF_MICR_CHAR_MAX];     OUT
    CHAR szSerialNumber[MF_MICR_CHAR_MAX];   OUT
    CHAR szEPC[MF_MICR_CHAR_MAX];           OUT
    CHAR szTransitNumber[MF_MICR_CHAR_MAX];  OUT
    long lCheckType;                         OUT
    long lCountryCode;                      OUT
    CHAR szOnUSField[MF_MICR_CHAR_MAX];      OUT
    CHAR szAuxiliaryOnUsField[MF_MICR_CHAR_MAX]; OUT
} MF_MICR01, *LPMF_MICR01;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_MICR_VERSION01. Since the MF_MICR_VERSION01 value is different for each driver version, the application must always use the supplied header file.

int iRet

This sets the return values for running MICR OCR. Refer to "[MF_MICR01.iRet](#)" on page 112 for details.

BYTE 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 (CMC7 does not support acquisition of the OCR recognition result and Parsing.)

BYTE bMicOcrSelect

This member is compatible with the TM-S1000 API. With the TM-S9000/S2000 API, the settings for this member are ignored and “MF_MICR_USE_MICR | MF_MICR_USE_OCR” is always applied.

BOOL bIParsing

When TRUE is specified, parsing is carried out, and the character string is set in szAccountNumber, szAmount, szBankNumber, szSerialNumber, szEPC and szTransitNumber. The number is set in lCheck-Type, lCountryCode.

FALSE is specified, parsing is not carried out. When bFont is set to MF_MICR_FONT_CMC7, set False.

BYTE bStatus

The MICR reading status is set.

Bit	Status Contents	ON / OFF	Value	Status
0	Reading font	ON	0x0001	CMC7
		OFF	0x0000	E13B
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	OCR processing error	ON	0x0040	Yes
		OFF	0x0000	No
7	Readout data reception error	ON	0x0080	Yes
		OFF	0x0000	No



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

BYTE bDetail

Detailed MICR reading status is set.

Value	Information
40h	Success
41h	Check sheet reading has not ever been executed. (The BiSCNMICRFunction function has not been invoked.)
44h	A delivery error occurred in the processing before reading.
45h	A magnetic waveform cannot be detected.
46h	Characters that cannot be analyzed were detected in the analysis processing.
47h	A double-feeding error or an insertion direction error occurred during check sheet reading.
48h	An abnormality was detected in noise measurement.
49h	Check sheet reading was stopped due to a feeding error.
4Bh	In reading, an error of paper length too long occurred.

CHAR szMicrStr[MF_MICR_CHAR_MAX]

The read MICR data is set.

Refer to "[MICR Control Characters](#)" on page 289 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 "[MF_OCR_RELIABLE_INFO structure](#)" on page 293 for details.

CHAR szAccountNumber[MF_MICR_CHAR_MAX]

AccountNumber property value.

CHAR szAmount[MF_MICR_CHAR_MAX]

Amount property value.

CHAR szBankNumber[MF_MICR_CHAR_MAX]

BankNumber property value.

CHAR szSerialNumber[MF_MICR_CHAR_MAX]

SerialNumber property value.

CHAR szEPC[MF_MICR_CHAR_MAX]

EPC property value.

CHAR szTransitNumber[MF_MICR_CHAR_MAX]

TransitNumber property value.

long lCheckType

CheckType property value.

long lCountryCode

CountryCode property value.

CHAR szOnUSField[MF_MICR_CHAR_MAX];

The MICR On-US field is stored.

CHAR szAuxiliaryOnUsField[MF_MICR_CHAR_MAX];

The Auxiliary On-US field is stored.





Default Values of the MF_MICR01 Structure

Member of MF_MICR01 Structure	Default Value
MF_MICR01.bFont	MF_MICR_FONT_E13B
MF_MICR01.bMicOcrSelect	MF_MICR_USE_MICR
MF_MICR01.bIParsing	FALSE
MF_MICR01.bStatus	0
MF_MICR01.bDetail	0
MF_MICR01.szMicStr	Zero clear
MF_MICR01.stOcrReliableInfo	Zero clear
MF_MICR01.szAccountNumber	Zero clear
MF_MICR01.szAmount	Zero clear
MF_MICR01.szBankNumber	Zero clear
MF_MICR01.szSerialNumber	Zero clear
MF_MICR01.szEPC	Zero clear
MF_MICR01.szTransitNumber	Zero clear
MF_MICR01.lCheckType	0
MF_MICR01.lCountryCode	0
MF_MICR01.szOnUSField	Zero clear
MF_MICR01.szAuxiliaryOnUsField	Zero clear






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
	/
	#
	=
	>
	^

MF_OCR_AB

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    int iRet;                                 OUT
    BYTE bOcrType;                            IN
    BYTE bDirection;                         IN
    WORD wStartX;                             IN
    WORD wStartY;                             IN
    WORD wEndX;                               IN
    WORD wEndY;                               IN
    BYTE bSpaceHandling;                     IN
    CHAR szOcrStr[MF_OCR_AB_CHAR_MAX];        OUT
    MF_OCR_RELIABLE_INFO stOcrReliableInfo[MF_OCR_AB_CHAR_MAX]; OUT
} MF_OCR_AB, *LPMF_OCR_AB;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_OCR_AB_VERSION. Since the MF_OCR_AB_VERSION value is different for each driver version, the application must always use the supplied header file.

int iRet

Stores the return value of the OCR recognition processing. Refer to "[MF_MICR01.iRet](#)" on page 112 for details.

BYTE bOcrType

Specifies the font type. One of the following can be specified.

bOcrType (Constant)	Description
MF_OCR_FONT_OCRA_NUM	OCR-A font and numbers only
MF_OCR_FONT_OCRB_NUM	OCR-B font and numbers only
MF_OCR_FONT_OCRA_ALPHA	OCR-A font and alphabetic characters only
MF_OCR_FONT_OCRB_ALPHA	OCR-B font and alphabetic characters only
MF_OCR_FONT_OCRA_ALPHANUM	OCR-A font and alphanumeric characters
MF_OCR_FONT_OCRB_ALPHANUM	OCR-B font and alphanumeric characters
MF_OCR_FONT_OCRA_ALPHANUM_WOOH	OCR-A font and alphanumeric characters (except for OH.)
MF_OCR_FONT_OCRB_ALPHANUM_WOOH	OCR-B font and alphanumeric characters (except for OH.)
MF_OCR_FONT_OCRA_ALPHANUM_WOZERO	OCR-A font and alphanumeric characters (except for ZERO.)
MF_OCR_FONT_OCRB_ALPHANUM_WOZERO	OCR-B font and alphanumeric characters (except for ZERO.)

bOcrType (Constant)	Description
MF_OCR_FONT_OCRA_SYMNUM	OCR-A font, numbers, and symbols (excluding "+")
MF_OCR_FONT_OCRB_SYMNUM	OCR-B font, numbers, and symbols (including "+")

BYTE bDirection

Specifies the character direction for the area for which the OCR recognition is executed. One of the following values can be specified.

bDirection (Constant)	Description
MF_OCR_LEFTRIGHT	From left to right (normal direction)
MF_OCR_TOPBOTTOM	From top to bottom (90-clockwise rotation)
MF_OCR_RIGHTLEFT	From right to left (flip vertical)
MF_OCR_BOTTOMTOP	From bottom to top (90-counterclockwise rotation)

WORD wStartX

Specifies the starting point X-coordinate of the area for which the OCR recognition is executed (unit: mm). The available range is 0 to 254.

WORD wStartY

Specifies the starting point Y-coordinate of the area for which the OCR recognition is executed (unit: mm). The available range is 0 to 255.

WORD wEndX

Specifies the ending point X-coordinate of the area for which the OCR recognition is executed.

wEndX	Description
1 to 255 (unit: mm)	-
OCR_AREA_RIGHT	The right side of an image can be specified.
OCR_AREA_LEFT	The left side of an image can be specified.



In the following cases, when something outside OCR_AREA_RIGHT and OCR_AREA_LEFT is specified, an ERR_PARAM is returned.

- If $wStartX \geq wEndX$
- If the direction of the character string in the area subject to OCR is specified to MF_OCR_TOPBOTTOM or MF_OCR_BOTTOMTOP

WORD wEndY

Specifies the ending point Y-coordinate of the area for which the OCR recognition is executed.

wEndY	Description
1 to 256 (unit: mm)	-
OCR_AREA_BOTTOM	The bottom side of an image can be specified.
OCR_AREA_TOP	The top side of an image can be specified.



- When the ending point is specified to the origin, all the area for which the OCR recognition is executed can be specified. In this case, the character string is analyzed as one line.
- In the following cases, when something outside OCR_AREA_BOTTOM and OCR_AREA_TOP is specified, an ERR_PARAM is returned.
 - * If wStartY >= wEndY
 - * If the direction of the character string in the area subject to OCR is specified to MF_OCR_LEFTRIGHT or MF_OCR_RIGHTLEFT

BYTE bSpaceHandling

Specifies a handling method of space characters for the OCR recognition processing. One of the following values can be specified.

bSpaceHandling (Constant)	Description
OCR_SPACE_ENABLE	Space characters are included in the OCR recognition result.
OCR_SPACE_DISABLE	Space characters are not included in the OCR recognition result.

CHAR szOcrStr[MF_OCR_AB_CHAR_MAX]

Stores character strings acquired by the OCR recognition processing.

MF_OCR_RELIABLE_INFO stOcrReliableInfo[MF_OCR_AB_CHAR_MAX]

Sets the candidates and the reliability reliabilities for the first and the second characters acquired by the OCR recognition processing.

Refer to "[MF_OCR_RELIABLE_INFO structure](#)" on page 293 for details.



Character types and reliabilities for the first and the second candidates can be acquired at the same time as shown in the example below:

```
stFirstSelect.cRecogChar;      0
stFirstSelect.IPercentage;     80%
stSecondSelect.cRecogChar;    0
stSecondSelect.IPercentage;   20%
```

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;
```

long lPosition

Position (0 is left edge).

MF_OCR_RELIABILITY stFirstSelect

First recognition candidate. Refer to ["MF_OCR_RELIABILITY structure" on page 293](#) for details.

MF_OCR_RELIABILITY stSecondSelect

Second recognition candidate. Refer to ["MF_OCR_RELIABILITY structure" on page 293](#) for details.

MF_OCR_RELIABILITY structure

```
typedef struct {
    char cRecogChar;                                OUT
    long lPercentage;                                OUT
} MF_OCR_RELIABILITY *LPMF_OCR_RELIABILITY;
```

char cRecogChar

Recognized characters.

long lPercentage

Reliability (%).

Default Values of the MF_OCR_AB Structure

Member of MF_OCR_AB Structure	Default Value
MF_OCR_AB.bOcrType	
MF_OCR_AB.bDirection	
MF_OCR_AB.wStartX	
MF_OCR_AB.wStartY	
MF_OCR_AB.wEndX	
MF_OCR_AB.wEndY	
MF_OCR_AB.bSpaceHandling	
MF_OCR_AB.szOcrStr	Zero clear
MF_OCR_AB.stOcrReliableInfo	Zero clear

MF_PRINT01



This structure is used only for compatibility with the TM-J9000/TM-S1000 drivers. On this product, please print using the printing-related API (such as BiPrintText).

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    int iRet;                                 OUT
    BOOL bIDummy;                             IN
    LPSTR lpString[3];                        IN
    DWORD dwAttribute[3];                    IN
    WORD wFont[3];                           IN
    WORD wFontSize[3];                       IN
    BYTE bSpeed;                             IN
    BOOL bDirection;                         IN
    DWORD dwEndorseType;                     IN
} MF_PRINT01, *LPMF_PRINT01;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_PRINT_VERSION01. Since the MF_PRINT_VERSION01 value is different for each driver version, the application must always use the supplied header file.

int iRet

This sets the return values for validation printing. Refer to "[MF_PRINT01.iRet](#)" on page 112 for details.

BOOL bIDummy

Not used.

LPSTR lpString[3]

Specifies the address of the ASCII character string for electronic endorsement printing.

A pointer for the character string of the first line is specified to lpString[0], a pointer for the character string of the second line is specified to lpString[1], and a pointer for the character string of the 3rd line is specified to lpString[2]. If all are NULL pointers, ERR_PARAM is returned. "LF" may be added to the end of the last print line. When line-feeding and printing the first and second lines, the line-feeding code (CR+LF) needs to be added to the ends of the ASCII character strings.

DWORD dwAttribute[3]

Specifies the attribute of the character string of the first line to dwAttribute[0], the attribute of the character string of the second line to dwAttribute[1] and the attribute of the character string of the 3rd line to dwAttribute[2]. The attribute is specified with the following bits and multiple attributes can be specified.

dwAttribute (Constant)	Description
MF_PRINT_BOLD	Executes emphasized printing
MF_PRINT_UNDERLINE_1	Adds a 1-line width of UnderLine
MF_PRINT_UNDERLINE_2	Adds a two-line width of UnderLine
MF_PRINT_REVERSEVIDEO	Executes reverse printing
MF_PRINT_BLACK	Prints a character with the first color (usually it is black)
MF_PRINT_COLOR	<ul style="list-style-type: none"> Prints a character with the second color. Even if this value is set, text is printed using the 1st color.
MF_PRINT_MIXED	<ul style="list-style-type: none"> Prints a character with the first and second colors Even if this value is set, text is printed using the 1st color.
MF_PRINT_1ST_COLOR	Prints a character with the first color (usually it is black)
MF_PRINT_2ND_COLOR	Prints a character with the second color
MF_PRINT_NO_ATTRIBUTE	Not specifying the attribute



- Bits other than the above are ignored.
- When not specifying the attribute, specifies MF_PRINT_NO_ATTRIBUTE.
- The line width of the underlines developed with MF_PRINT_UNDERLINE_1 and MF_PRINT_UNDERLINE_2 is the same.
- Even if MF_PRINT_COLOR, MF_PRINT_MIXED is specified to dwAttribute of the MF_DECORATE structure, the character string printed is MF_PRINT_BLACK.

WORD wFont[3]

Specifies the text font.

wFont[0] specifies FONT of the character string of the first line, wFont[1] specifies FONT of the character string of the second line, and wFont[2] specifies FONT of the character string of the 3rd line. One of the following is set for the font.

wFont (Constant)	Description
MF_PRINT_FONT_A	Prints with device font A.
MF_PRINT_FONT_B	Prints with device font B.



- If a value other than the above is specified, ERR_PARAM is returned.
- The values above are used only for parameter checking. The fonts used for the electronic endorsement depend on the operating environments.

WORD wFontSize[3]

Specifies the font size.

wFontSize[0] specifies the FONT size of the character string of the first line, wFontSize[1] specifies the FONT size of the character string of the second line, and wFontSize[2] specifies the FONT size of the character string of the 3rd line. One of the following is set.

wFontSize (Constant)	Description
MF_PRINT_FONT_W1_H1	A font with 1 unit horizontal and 1 vertical
MF_PRINT_FONT_W1_H2	A font with 1 unit horizontal and 2 vertical
MF_PRINT_FONT_W2_H1	A font with 2 units horizontal and 1 vertical
MF_PRINT_FONT_W2_H2	A font with 2 units horizontal and 2 vertical



If a value other than the above is specified, ERR_PARAM is returned.

BYTE bSpeed

Not used.

BOOL bDirection

Not used.

DWORD dwEndorseType

This specifies the transaction printing process and the electronic endorsement process.

One of the following values can be specified.

dwEndorseType (Constant)	Description
MF_PRINT_TYPE_ELECTRIC_ENDORSE_ONLY	Executes electronic endorsement printing using the data specified with lpString.
MF_PRINT_TYPE_ELECTRIC_ENDORSE_EXTEND	When the electronic endorsement (MF_ST_E_ENDORSEMENT, MF_ST_E_ENDORSEMENT_BACK, MF_ST_E_ENDORSEMENT_FRONT) is specified with BiSetPrintStation, after registering the size of characters or images developed with BiSetPrintSize and specifying the position of characters or images developed with BiSetPrintPosition, the electronic endorsement printing can be executed by executing BiPrintText, BiPrintImage, BiPrintMemoryImage.
MF_PRINT_TYPE_ENDORSE_NORMAL	Performs both endorsement printing and electronic endorsement printing by using data specified by lpString.
MF_PRINT_TYPE_ENDORSE_EXTEND	Executes endorsement printing using the data specified with lpString.

Default Values of the MF_PRINT01 Structure

Member of MF_PRINT01 Structure	Default Value
MF_PRINT01.blDummy	FALSE
MF_PRINT01.lpString	NULL
MF_PRINT01.dwAttribute	MF_PRINT_NO_ATTRIBUTE
MF_PRINT01.wFont	MF_PRINT_FONT_A
MF_PRINT01.wFontSize	MF_PRINT_FONT_W1_H1
MF_PRINT01.bSpeed	MF_PRINT_SPEED_HIGH
MF_PRINT01.bDirection	MF_PRINT_DIRECTION_DOUBLE
MF_PRINT01.dwEndorseType	MF_PRINT_TYPE_ELECTRIC_ENDORSE_EXTEND

MF_PROCESS01

This structure is an option; therefore, this does not always need to be set.

When the MF_PROCESS01 structure is not set and reading is started, operates based on the default value of the MF_PROCESS01 structure; however, if the settings that correspond to the MF_BASE01 structure exist, the settings of the MF_BASE01 structure have priority.

```
typedef struct {
    int iSize;
    int iVersion;
    BYTE bActivationMode;
    BYTE bPaperType;
    DWORD dwStartWaitTime;
    BYTE bSuccessStamp;
    BYTE bPaperMisInsertionErrorSelect;
    BYTE bPaperMisInsertionErrorEject;
    BYTE bPaperMisInsertionStamp;
    BYTE bPaperMisInsertionCancel;
    BYTE bNoiseErrorSelect;
    BYTE bNoiseErrorEject;
    BYTE bNoiseStamp;
    BYTE bNoiseCancel;
    BYTE bDoubleFeedErrorSelect;
    BYTE bDoubleFeedErrorEject;
    BYTE bDoubleFeedStamp;
    BYTE bDoubleFeedCancel;
    BYTE bBaddataErrorSelect;
    BYTE bBaddataCount;
    BYTE bBaddataEject;
    BYTE bBaddataStamp;
    BYTE bBaddataCancel;
    BYTE bNodataErrorSelect;
    BYTE bNodataErrorEject;
    BYTE bNodataStamp;
    BYTE bNodataCancel;
    BYTE bNearFullSelect;
    BYTE bResultPartialData;
    BYTE bEndorsePrintMode;
    BYTE bPrnDataLenExceedErrorEject;
    BYTE bPrnDataLenExceedCancel;
    BYTE bPrnDataUnreceiveErrorEject;
    BYTE bPrnDataUnreceiveCancel;
} MF_PROCESS01, *LPMF_PROCESS01;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_PROCESS_VERSION01.

Since the MF_PROCESS_VERSION01 value is different for each driver version, the application must always use the supplied header file.

BYTE bActivationMode

This sets the activation mode for scanning. The valid commands are listed below.

bActivationMode (Constant)	Description
MF_ACTIVATE_MODE_HIGH_SPEED	High-speed scan mode
MF_ACTIVATE_MODE_CONFIRMATION	Confirmation scan mode



- When MF_ACTIVATE_MODE_HIGH_SPEED is specified, the driver carries out all the error verification at scanning. The driver deals with errors according to the pre-set actions.
- The actions at error occurrence are specified in this structure. Scan speed may slow down depending on the setting for the action at error occurrence.
- When MF_ACTIVATE_MODE_CONFIRMATION is specified, the application carries out all the error verification at scanning. For details of the error verification by the application, refer to the function "[BiSetBehaviorToScnResult](#)" on page 254.

BYTE bPaperType

This sets the paper type for a scan target. The valid commands are listed below.

bPaperType (Constant)	Description
MF_PAPER_TYPE_CHECK	Check sheets only
MF_PAPER_TYPE_OTHER	Papers other than check sheets included



This member is compatible with the TM-S1000 API. With the TM-S9000/S2000 API, the settings for this member are ignored.



When MF_PAPER_TYPE_CHECK is specified and a paper other than check sheets is scanned, a scan error may occur.

DWORD dwStartWaitTime

This sets the wait time that is taken before loading document.

The valid setting value is 0 to 6400 (unit: ms).

When a value exceeding 6400 is set, it will be rounded to 6400.



This sets the wait time that is taken from when the device has become ready for the insertion until the insertion starts.

BYTE bSuccessStamp

This sets whether to enable a Franker when scan completes successfully. The valid commands are listed below.

bSuccessStamp (Constant)	Description
MF_STAMP_ENABLE	Check sheets only
MF_STAMP_DISABLE	Papers other than check sheets included



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- In the High Speed mode, operates franker in accordance with this setting value.
- In the Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, operates franker in accordance with this setting value.
- Stamping operation cannot be shifted with Baddata, Nodata, or Success without slowing down the reading speed.

BYTE bPaperMisInsertionErrorSelect

This sets whether to detect the insertion direction error. The valid commands are listed below.

bPaperMisInsertionErrorSelect (Constant)	Description
MF_ERROR_SELECT_NODETECT	No error detected
MF_ERROR_SELECT_DETECT	Error detected



- When MF_ERROR_SELECT_NODETECT is specified, no error is detected even if the insertion direction is wrong.
- When MF_ERROR_SELECT_DETECT is specified, the action at error occurrence is taken according to the following.
 - * bPaperMisInsertionErrorEject
 - * bPaperMisInsertionStamp
 - * bPaperMisInsertionCancel

BYTE bPaperMisInsertionErrorEject

This sets the ejection method for when the insertion direction error is detected. The valid commands are listed below.

bPaperMisInsertionErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket
MF_EJECT_NOEJECT	Not ejected (completed with error)



- This setting takes place only when MF_ERROR_SELECT_DETECT is specified in bPaperMisInsertionErrorSelect.
- When MF_EJECT_NOEJECT is specified, the following values are ignored.
 - * bPaperMisInsertionStamp
 - * bPaperMisInsertionCancel

BYTE bPaperMisInsertionStamp

This sets the whether to enable a franker for when the insertion direction error is detected. The valid commands are listed below.

bPaperMisInsertionStamp (Constant)	Description
MF_STAMP_ENABLE	Franker enabled
MF_STAMP_DISABLE	Franker disabled



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- If MF_ERROR_SELECT_NODETECT is set to bPaperMisInsertionErrorSelect, the setting of bPaperMisInsertionStamp is ignored.
- In the High Speed mode, operates franker in accordance with this setting value.
- In the Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, operates franker in accordance with this setting value.

BYTE bPaperMisInsertionCancel

Sets whether to continue reading operation when an insertion direction incorrect error occurs.

bPaperMisInsertionCancel (Constant)	Description
MF_CANCEL_ENABLE	When MF_CANCEL_ENABLE is specified, does not cancel the reading process for the next check sheet.
MF_CANCEL_DISABLE	Reading process for the next check sheet is canceled



- When MF_ERROR_SELECT_NODETECT is set to bPaperMisInsertionErrorSelect, the setting of bPaperMisInsertionCancel is ignored.
- In the High Speed mode, operates in accordance with this setting value.
- In the Confirmation mode, the insertion direction incorrect error is notified with MF_ERROR_OCCURRED callback and the reading process is continued. If "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, operates in accordance with this setting value.

BYTE bNoiseErrorSelect

Specifies error detection enable/disable when an external noise error is detected.

bNoiseErrorSelect (Constant)	Description
MF_ERROR_SELECT_NODETECT	No error detected
MF_ERROR_SELECT_DETECT	Error detected



- When "MF_ERROR_SELECT_NODETECT" is set, the following settings are ignored.
- * bNoiseErrorEject
 - * bNoiseErrorStamp
 - * NoiseErrorCancel

BYTE bNoiseErrorEject

Specifies where to eject when an external noise error occurs.

Corresponds to wErrorEject of the MF_BASE01 structure. Even if the MF_BASE01 structure is set, the value of this structure has a priority.

bNoiseErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket
MF_EJECT_NOEJECT	Not ejected (completed with error)



- When MF_ERROR_SELECT_NODETECT is set to bNoiseErrorSelect, the setting of bNoiseErrorEject is ignored.
- In the High Speed mode, paper is ejected into a pocket in accordance with this setting value.
- In the Confirmation mode, if ["BiSetBehaviorToScnResult" on page 254](#) is not invoked in the MF_DATARECEIVE_DONE callback notification, paper is ejected into a pocket in accordance with this setting value.

BYTE bNoiseStamp

Specifies a franker processing when an external noise error occurs.

bNoiseStamp (Constant)	Description
MF_STAMP_ENABLE	Franker enabled
MF_STAMP_DISABLE	Franker disabled



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- When MF_ERROR_SELECT_NODETECT is specified to bNoiseErrorSelect, the setting of bNoiseStamp is ignored.
- In the High Speed mode, executes a franker operation in accordance with this setting value.
- In the Confirmation mode, if ["BiSetBehaviorToScnResult" on page 254](#) is not invoked in the MF_DATARECEIVE_DONE callback notification, executes a franker operation in accordance with this setting value.

BYTE bNoiseCancel

Sets whether to continue reading when an external noise error occurs.

bNoiseCancel (Constant)	Description
MF_CANCEL_ENABLE	Reading process for the next check sheet is canceled
MF_CANCEL_DISABLE	Reading process for the next check sheet is not canceled



- Corresponds to errorSelect of BiMICRSelectDataHandling. To have the compatibility with the TM-J9000 driver, when this structure is not set and the default value of this structure is not the same as the value of errorSelect, the value of errorSelect has a priority.
- When MF_ERROR_SELECT_NODETECT is specified to bNoiseErrorSelect, the setting of bNoiseCancel is ignored.
- In the High Speed mode, executes the operation in accordance with this setting value.
- In the Confirmation mode, an external noise error is notified in the MF_ERROR_OCCURRED callback and the reading process is continued. If ["BiSetBehaviorToScnResult" on page 254](#) is not invoked in the MF_DATARECEIVE_DONE callback notification, executes the operation in accordance with this setting value.

BYTE bDoubleFeedErrorSelect

This sets whether to detect the double feed error. The valid commands are listed below.

bDoubleFeedErrorSelect (Constant)	Description
MF_ERROR_SELECT_NODETECT	No error detected
MF_ERROR_SELECT_DETECT	Error detected



- When MF_ERROR_SELECT_NODETECT is specified, no error is detected even if a double feed occurs.
- When MF_ERROR_SELECT_DETECT is specified, the action at error occurrence is taken according to the following.
 - * bDoubleFeedErrorEject
 - * bDoubleFeedStamp
 - * bDoubleFeedCancel

BYTE bDoubleFeedErrorEject

This sets the ejection method for when the double feed error is detected. The valid commands are listed below.

bDoubleFeedErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket
MF_EJECT_NOEJECT	Not ejected (completed with error)



- This setting takes place only when MF_ERROR_SELECT_DETECT is specified with bDoubleFeedErrorSelect.
- When MF_EJECT_NOEJECT is specified, the following values are ignored.
 - * bDoubleFeedStamp
 - * bDoubleFeedCancel

BYTE bDoubleFeedStamp

This sets whether to enable franker when the double feed error is detected. The valid commands are listed below.

bDoubleFeedStamp (Constant)	Description
MF_STAMP_ENABLE	Franker enabled
MF_STAMP_DISABLE	Franker disabled



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- When MF_ERROR_SELECT_NODETECT is specified to bDoubleFeedErrorSelect, the setting of bDoubleFeedStamp is ignored.
- In the High Speed mode, a franker operation is executed in accordance with this setting value.
- In the Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, a franker operation is executed in accordance with this setting value.

BYTE bDoubleFeedCancel

Sets whether to continue reading when a double-feed error occurs.

bDoubleFeedCancel (Constant)	Description
MF_CANCEL_ENABLE	Reading process for the next check sheet is canceled
MF_CANCEL_DISABLE	Reading process for the next check sheet is not canceled



- Corresponds to errorSelect of BiMICRSelectDataHandling. To have the compatibility with the TM-J9000 driver, when this structure is not set and the default value of this structure is not the same as the value of errorSelect, the value of errorSelect has a priority.
- When MF_ERROR_SELECT_NODETECT is specified to bDoubleFeedErrorSelect, the setting of bDoubleFeedCancel is ignored.
- In the High Speed mode, the operation is executed in accordance with this setting value.
- In the Confirmation mode, the double-feed error is notified in the MF_ERROR_OCCURRED callback notification and the reading process is continued. If "[BiSetBehaviorToScnResult](#)" on [page 254](#) is not invoked in the MF_DATARECEIVE_DONE callback notification, executes the operation in accordance with this setting value.

BYTE bBaddataErrorSelect

Sets whether to detect unrecognizable MICR characters as errors.

When value of this member is MF_ERROR_SELECT_DETECT and number of unrecognizable characters is greater than the permissible value set with bBaddataCount, unrecognizable MICR characters are treated as an error. The valid commands are listed below.

bBaddataErrorSelect (Constant)	Description
MF_ERROR_SELECT_NODETECT	No error detected
MF_ERROR_SELECT_DETECT	Error detected



When "MF_ERROR_SELECT_NODETECT" is set, the following settings are ignored.

- * bBaddataCount
- * bBaddataErrorEject
- * bBaddataStamp
- * bBaddataCancel

BYTE bBaddataCount

Set the amount of unrecognizable characters permitted (the number below which is not determined as an error) when detecting unrecognizable MICR characters as errors.

When value of bBaddataErrorSelect is MF_ERROR_SELECT_DETECT and number of unrecognizable characters is greater than the permissible value set with this member, unrecognizable MICR characters are treated as an error. The valid setting value is 0 to 255.



When "MF_ERROR_SELECT_NODETECT" is set, the following settings are ignored.

- * bBaddataCount
- * bBaddataErrorEject
- * bBaddataStamp
- * bBaddataCancel

BYTE bBaddataErrorEject

This sets the ejection method for when the MICR character recognition error is detected and the number of characters detected exceeds the permissible number. The valid commands are listed below.

bBaddataErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket
MF_EJECT_NOEJECT	Not ejected (completed with error)



- When bBaddataErrorSelect is specified to MF_ERROR_SELECT_NODETECT or when the number of characters that cannot be analyzed is not over fewer than the permissible number specified with bBaddataCount, the setting of bBaddataErrorEject is ignored.
- In the High Speed mode, paper is ejected to the pocket corresponding to this setting value.
- In the Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, paper is ejected to the pocket corresponding to this setting value.
- When MF_ACTIVATE_MODE_HIGH_SPEED is set to bActivationMode, the scan speed will slow down if a value other than MF_EJECT_MAIN_POCKET is specified in this setting.

BYTE bBaddataStamp

This sets whether to enable a franker for when the MICR character recognition error is detected and the number of error characters exceeds the permissible number. The valid commands are listed below.

bBaddataStamp (Constant)	Description
MF_STAMP_ENABLE	Franker enabled
MF_STAMP_DISABLE	Franker disabled



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- This setting takes place only when MF_ERROR_SELECT_DETECT is specified in bBaddataErrorSelect and the number of unrecognized characters exceeds the permissible number set in bBaddataCount.
- When MF_ACTIVATE_MODE_HIGH_SPEED is set to bActivationMode, the scan speed will slow down if this setting is different from the following.
 - * bSuccessStamp
 - * bNodataStamp
- In the Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, a franker operation is executed in accordance with this setting value.

BYTE bBaddataCancel

This sets whether to cancel the action for when the MICR character recognition error is detected and the number of error characters exceeds the permissible number. The valid commands are listed below.

bBaddataCancel (Constant)	Description
MF_CANCEL_ENABLE	Reading process for the next check sheet is canceled
MF_CANCEL_DISABLE	Reading process for the next check sheet is not canceled



- Corresponds to errorSelect of BiMICRSelectDataHandling. To have the compatibility with the TM-J9000 driver, when this structure is not set and the default value of this structure is not the same as the value of errorSelect, the value of errorSelect has a priority.
- When bBaddataErrorSelect is specified to MF_ERROR_SELECT_NODETECT or when the number of characters that cannot be analyzed is fewer than the permissible number specified with bBaddataCount, the setting of bBaddataCancel is ignored.
- In the High Speed mode, executes the operation corresponding to this setting value.
- In the Confirmation mode, if the number of characters that cannot be analyzed is over the permissible number, it is notified in the MF_ERROR_OCCURRED callback and the reading process is continued.
- If "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, executes the operation corresponding to this setting value.
- When MF_ACTIVATE_MODE_HIGH_SPEED is set to bActivationMode, the scan speed will slow down if MF_CANCEL_ENABLE is specified in this setting.

BYTE bNodataErrorSelect

This sets whether to detect errors when magnetic waveform is not found. The valid commands are listed below.

bNodataErrorSelect (Constant)	Description
MF_ERROR_SELECT_NODETECT	No error detected
MF_ERROR_SELECT_DETECT	Error detected



- When MF_ERROR_SELECT_NODETECT is specified, no error is detected even if no MICR magnetic waveform is found.
- When MF_ERROR_SELECT_DETECT is specified, the action at error occurrence is taken according to the following.
 - * bNodataErrorEject
 - * bNodataStamp
 - * bNodataCancel

BYTE bNodataErrorEject

This sets the ejection method for when an error is detected because MICR magnetic waveform is not found. The valid setting values are as shown below.

bNodataErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket
MF_EJECT_NOEJECT	Not ejected (completed with error)



- This setting takes place only when MF_ERROR_SELECT_DETECT is specified in bNodataErrorSelect.
- When MF_EJECT_NOEJECT is specified, the following values are ignored.
 - * bNodataStamp
 - * bNodataCancel
- When MF_ACTIVATE_MODE_HIGH_SPEED is set to bActivationMode, the scan speed will slow down if a value other than MF_EJECT_MAIN_POCKET is specified in this setting.

BYTE bNodataStamp

This sets whether to enable a franker for when an error is detected because MICR magnetic waveform is not found. The valid commands are listed below.

bNodataStamp (Constant)	Description
MF_STAMP_ENABLE	Frunker enabled
MF_STAMP_DISABLE	Frunker disabled



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- This setting takes place only when MF_ERROR_SELECT_DETECT is specified in bNodataErrorSelect.
- When MF_ACTIVATE_MODE_HIGH_SPEED is set to bActivationMode, the scan speed will slow down if this setting is different from the following.
 - * bSuccessStamp
 - * bBaddataStamp
- In the Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, a franker operation is executed in accordance with this setting value.

BYTE bNodataCancel

This sets whether to cancel the action for when an error is detected because that MICR magnetic waveform is not found. The valid commands are listed below.

bNodataCancel (Constant)	Description
MF_CANCEL_ENABLE	Reading process for the next check sheet is canceled
MF_CANCEL_DISABLE	Reading process for the next check sheet is not canceled



- Corresponds to errorSelect of BiMICRSelectDataHandling. To have the compatibility with the TM-J9000 driver, when this structure is not set and the default value of this structure is not the same as the value of errorSelect, the value of errorSelect has a priority.
- When bNodataErrorSelect is specified to MF_ERROR_SELECT_NODETECT, the setting of bNodataCancel is ignored.
- In the High Speed mode, executes the operation corresponding to this setting value.
- In the Confirmation mode, an MICR magnetic waveform undetected error is notified in the MF_ERROR_OCCURRED callback, and the reading process is continued. If "[BiSetBehavior-ToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, executes the operation corresponding to this setting value.
- When MF_ACTIVATE_MODE_HIGH_SPEED is set to bActivationMode, the scan speed will slow down if MF_CANCEL_ENABLE is specified in this setting.

BYTE bNearFullSelect

This sets whether to permit scanning when the eject pocket is nearly full. The valid commands are listed below.

bNearFullSelect (Constant)	Description
MF_NEARFULL_PERMIT	Scanning is continued even if either or both of the main pocket and sub pocket are nearly full.
MF_NEARFULL_MAIN_PERMIT	Scanning is continued when the main pocket is nearly full and is stopped when the sub pocket is nearly full.
MF_NEARFULL_SUB_PERMIT	Scanning is continued when the sub pocket is nearly full and is stopped when the main pocket is nearly full.
MF_NEARFULL_NOT_PERMIT	Scanning is stopped when either or both of the main pocket and sub pocket are nearly full.



- When MF_NEARFULL_NOT_PERMIT is specified, scanning stops if the ejection pocket is found nearly full.
- Do not use MF_NEARFULL_MAIN_PERMIT and MF_NEARFULL_SUB_PERMIT when the Waterfall mode is executed.

BYTE bResultPartialData

This sets whether to acquire data that is being scanned when an error that does not interrupt any operation occurs in the middle of the scanning. The valid commands are listed below.

bResultPartialData (Constant)	Description
MF_RESULT_PARTIAL	Data being scanned can be acquired
MF_RESULT_NONE	Data being scanned is deleted

BYTE bEndorsePrintMode

Print method for endorsement printing. Each of the following values can be specified independently.

bEndorsePrintMode (Constant)	Description
MF_ENDORSEPRINT_MODE_HIGHSPEED	If print data creation has not been completed within the period of time in which data transfer is permitted, the process goes on without printing.
MF_ENDORSEPRINT_MODE_DATAWAITING	If print data creation has not been completed within the period of time in which data transfer is permitted, waits at the print start position until data creation becomes complete.



If a Print Data Unreceived Error occurs, configuring this setting to MF_ENDORSEPRINT_MODE_DATAWAITING is recommended.

BYTE bPrnDataLenExceedErrorEject

This sets the ejection method for when the printing content exceeds printable area. The valid commands are listed below.

bPrnDataLenExceedErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket



- In High Speed mode, executes the operation corresponding to this setting value.
- In Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not called up in the MF_DATARECEIVE_DONE callback notification, exit pocket follows MF_PROCESS01 settings.

BYTE bPrnDataLenExceedCancel

Sets whether to continue reading operation when a paper length error occurs. The valid commands are listed below.

bPrnDataLenExceedCancel (Constant)	Description
MF_CANCEL_ENABLE	Reading process for the next check sheet is canceled
MF_CANCEL_DISABLE	Reading process for the next check sheet is not canceled



- In the High Speed mode, executes the operation corresponding to this setting value.
- In the Confirmation mode, a paper length error is notified in the MF_ERROR_OCCURRED callback, and the reading process is continued. If "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, executes the operation corresponding to this setting value.

BYTE bPrnDataUnreceiveErrorEject

This sets the ejection method for when the print data unreceived error is occurred. The valid commands are listed below.

bPrnDataUnreceiveErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket



- In High Speed mode, executes the operation corresponding to this setting value.
- In Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not called up in the MF_DATARECEIVE_DONE callback notification, exit pocket follows MF_PROCESS01 settings.
- Even if physical endorsement is not executed, the print data unreceived error may occur. Set MF_EJECT_SUB_POCKET and it ejects to the sub-pocket.

BYTE bPrnDataUnreceiveCancel

Sets whether to continue reading operation when a print data unreceived error occurs. The valid commands are listed below.

bPrnDataLenExceedCancel (Constant)	Description
MF_CANCEL_ENABLE	Reading process for the next check sheet is canceled
MF_CANCEL_DISABLE	Reading process for the next check sheet is not canceled



- In the High Speed mode, executes the operation corresponding to this setting value.
- In the Confirmation mode, a print data unreceived error is notified in the MF_ERROR_OC-CURRED callback, and the reading process is continued. If "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, executes the operation corresponding to this setting value.
- Even if physical endorsement is not executed, the print data unreceived error may occur. Set MF_CANCEL_ENABLE, and the scan is cancelled.

Default Values of the MF_PROCESS01 Structure

Member of MF_PROCESS01 Structure	Default Value
MF_PROCESS01.bActivationMode	MF_ACTIVATE_MODE_HIGH_SPEED
MF_PROCESS01.bPaperType	MF_PAPER_TYPE_CHECK
MF_PROCESS01.dwStartWaitTime	0
MF_PROCESS01.bSuccessStamp	MF_STAMP_DISABLE
MF_PROCESS01.bPaperMisInsertionErrorSelect	MF_ERROR_SELECT_DETECT
MF_PROCESS01.bPaperMisInsertionErrorEject	MF_EJECT_MAIN_POCKET
MF_PROCESS01.bPaperMisInsertionStamp	MF_STAMP_DISABLE
MF_PROCESS01.bPaperMisInsertionCancel	MF_CANCEL_DISABLE
MF_PROCESS01.bNoiseErrorSelect	MF_ERROR_SELECT_DETECT
MF_PROCESS01.bNoiseErrorEject	MF_EJECT_MAIN_POCKET
MF_PROCESS01.bNoiseStamp	MF_STAMP_DISABLE
MF_PROCESS01.bNoiseCancel	MF_CANCEL_DISABLE
MF_PROCESS01.bDoubleFeedErrorSelect	MF_ERROR_SELECT_DETECT
MF_PROCESS01.bDoubleFeedErrorEject	MF_EJECT_MAIN_POCKET
MF_PROCESS01.bDoubleFeedStamp	MF_STAMP_DISABLE
MF_PROCESS01.bDoubleFeedCancel	MF_CANCEL_DISABLE
MF_PROCESS01.bBaddataErrorSelect	MF_ERROR_SELECT_DETECT
MF_PROCESS01.bBaddataCount	255
MF_PROCESS01.bBaddataErrorEject	MF_EJECT_MAIN_POCKET
MF_PROCESS01.bBaddataStamp	MF_STAMP_DISABLE
MF_PROCESS01.bBaddataCancel	MF_CANCEL_DISABLE
MF_PROCESS01.bNodataErrorSelect	MF_ERROR_SELECT_DETECT
MF_PROCESS01.bNodataErrorEject	MF_EJECT_MAIN_POCKET
MF_PROCESS01.bNodataStamp	MF_STAMP_DISABLE
MF_PROCESS01.bNodataCancel	MF_CANCEL_DISABLE
MF_PROCESS01.bNearFullSelect	MF_NEARFULL_PERMIT
MF_PROCESS01.bResultPartialData	MF_RESULT_NONE
MF_PROCESS01.bEndorsePrintMode	MF_ENDORSEPRINT_MODE_HIGHSPEED
MF_PROCESS01.bPrnDataLenExceedErrorEject	MF_EJECT_MAIN_POCKET
MF_PROCESS01.bPrnDataLenExceedCancel	MF_CANCEL_DISABLE
MF_PROCESS01.bPrnDataUnreceiveErrorEject	MF_EJECT_MAIN_POCKET
MF_PROCESS01.bPrnDataUnreceiveCancel	MF_CANCEL_ENABLE

MF_IQA

```
typedef struct {
    int iSize;
    int iVersion;
    BYTE bErrorSelect;
    BYTE bErrorEject;
    BYTE bStamp;
    BYTE bCancel;
    BYTE bImageFormat;
    BYTE bColorDepth;
    CHAR bThreshold;
    BYTE bColor;
    BYTE bExOption;
    short sResolution;
    BYTE bUndersize;
    BYTE bOversize;
    BYTE bMincompressed;
    BYTE bMaxcompressed;
    BYTE bFront_rear;
    BYTE bToolight;
    BYTE bToodark;
    BYTE bStreaks;
    BYTE bNoise;
    BYTE bFocus;
    BYTE bCorners;
    BYTE bEdges;
    BYTE bFraming;
    BYTE bSkew;
    BYTE bCarbon;
    BYTE bPiggyback;
} MF_IQA, *LPMF_IQA;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_IQA_VERSION.

Since the MF_IQA_VERSION value is different for each driver version, the application must always use the supplied header file.

BYTE bErrorSelect

This sets whether to detect the image quality defects (IQA error). The valid commands are listed below.

bErrorSelect (Constant)	Description
MF_ERROR_SELECT_NODETECT	No error detected
MF_ERROR_SELECT_DETECT	Error detected



- When MF_ERROR_SELECT_NODETECT is specified, no error is detected even if the insertion direction is wrong.
- When MF_ERROR_SELECT_DETECT is specified, the action at error occurrence is taken according to the following. (For details, refer to the explanation of each element.)
 - * bErrorEject
 - * bStamp
 - * bCancel

BYTE bErrorEject

This sets the ejection method for when IQA error is detected. The valid commands are listed below.

bErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket
MF_EJECT_NOEJECT	Not ejected (completed with error)



- This setting takes place only when MF_ERROR_SELECT_DETECT is specified in bErrorSelect.
- This setting has a priority even when wErrorEject of MF_BASE01 structure is set.
- In the High Speed mode, paper is ejected to the pocket corresponding to this setting value.
- In the Confirmation mode, when ["BiSetBehaviorToScnResult" on page 254](#) is not called in the MF_DATARECEIVE_DONE callback notification, paper is ejected to the pocket corresponding to this setting value.
- When reading is processed in the High Speed mode, if this setting is set to other than MF_EJECT_MAIN_POCKET, reading speed slows down.

BYTE bStamp

This sets the whether to enable a franker for when the IQA error is detected. The valid commands are listed below.

bStamp (Constant)	Description
MF_STAMP_ENABLE	Franker enabled
MF_STAMP_DISABLE	Franker disabled



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- If MF_ERROR_SELECT_NODETECT is set to bErrorSelect, the setting of bStamp is ignored.
- In the High Speed mode, operates franker in accordance with this setting value.
- In the Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, operates franker in accordance with this setting value.
- If the setting for bSuccessStamp of MF_PROCESS01 structure differs from this setting when reading is processed in the High Speed mode, reading speed slows down.

BYTE bCancel

Sets whether to continue reading operation when the IQA error occurs.

bCancel (Constant)	Description
MF_CANCEL_DISABLE	Does not cancel the reading process for the next check sheet.
MF_CANCEL_ENABLE	Cancels the reading process for the next check sheet.



- When MF_ERROR_SELECT_NODETECT is set to bErrorSelect, the setting of bCancel is ignored.
- In the High Speed mode, operates in accordance with this setting value.
- In the Confirmation mode, the insertion direction incorrect error is notified with MF_ERROR_OCCURRED callback and the reading process is continued. If "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, operates in accordance with this setting value.
- When reading is processed in the High Speed mode, if this setting is set to other than MF_CANCEL_ENABLE, reading speed slows down.

BYTE bImageFormat

This sets an image format at the IQA validation. The valid commands are listed below.

bImageFormat (Constant)	Description
EPS_BI_SCN_TIFF	TIFF format CCITT (Group 4) compressed data
EPS_BI_SCN_RASTER	Raster format uncompressed data
EPS_BI_SCN_BITMAP	Bitmap format uncompressed data
EPS_BI_SCN_TIFF256	TIFF format uncompressed data
EPS_BI_SCN_JPEGHIGH	JPEG format high compression (size priority) data
EPS_BI_SCN_JPEGNORMAL	JPEG format normal compression data
EPS_BI_SCN_JPEGLow	JPEG format low compression (quality priority) data
EPS_BI_SCN_JTIFF	TIFF format JPEG compressed data

BYTE bColorDepth

This sets the gradation (bits per pixel) at the IQA validation. The valid commands are listed below.

bColorDepth (Constant)	Description
EPS_BI_SCN_1BIT	Black and White (1 bit)
EPS_BI_SCN_8BIT	256 grayscale (8 bit)

CHAR bThreshold

This sets the density threshold at the IQA validation.

Enabled when bColorDepth is set to EPS_BI_SCN_1BIT, and bExOption is set to EPS_BI_SCN_MANUAL. The valid value is -128 to 127.



- The value -128 to 127 corresponds to 0 to 255 of the brightness.
- When "0" is specified, the intermediate brightness "128" is applied.

BYTE bColor

This sets color at the IQA validation. The valid commands are listed below.

bColor (Constant)	Description
EPS_BI_SCN_MONOCHROME	Monochrome
EPS_BI_SCN_COLOR	Color



Even if you set EPS_BI_SCN_COLOR, the operation is the same as with EPS_BI_SCN_MONOCHROME.

BYTE bExOption

This sets the variety of density adjustment at the IQA validation. The valid commands are listed below.

bExOption (Constant)	Description
EPS_BI_SCN_MANUAL	Applies the value of bThreshold for Black and White.
EPS_BI_SCN_SHARP	Sharpening
EPS_BI_SCN_SHARP_CUSTOM	Custom sharpening
EPS_BI_SCN_SHARP_CUSTOM2	Combination of sharpening and custom sharpening

short sResolution

This sets the resolution at the IQA validation. The valid commands are listed below.

sResolution (Constant)	Description
MF_SCAN_DPI_DEFAULT	Scans at the default resolution for the device. (200 DPI)
MF_SCAN_DPI_200	Scans at 200 DPI.
MF_SCAN_DPI_300	Scans at 300 DPI.

Each of the following values can be specified independently only if EPS_BI_SCN_UNIT_CHECKPAPER is set to the unit when this structure is used. (Otherwise, it is considered to be not defined.)

sResolution (Constant)	Description
MF_SCAN_DPI_100	Scans at 100 DPI.
MF_SCAN_DPI_120	Scans at 120 DPI.
MF_SCAN_DPI_240	Scans at 240 DPI.

The following value can be specified independently only if EPS_BI_SCN_UNIT_CARD is set to the unit when this structure is used. (Otherwise, it is considered to be not defined.)

sResolution (Constant)	Description
MF_SCAN_DPI_600	Scans at 600 DPI.

BYTE bUndersize

This sets the execution of UndersizeImage validation. The valid commands are listed below.

bUndersize (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bOversize

This sets the execution of OversizeImage validation. The valid commands are listed below.

bOversize (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bMincompressed

This sets the execution of MinCompressedImageSize validation. The valid commands are listed below.

bMincompressed (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.



When the combination of bColorDepth and blmageFormat is other than the ones listed below, MinCompressedImageSize validation is not executed even if this is set to MF_IQA_TEST_ENABLE.

bColorDepth	blmageFormat
EPS_BI_SCN_1BIT	EPS_BI_SCN_TIFF
	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF
EPS_BI_SCN_8BIT	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF

BYTE bMaxcompressed

This sets the execution of MaxCompressedImageSize validation. The valid commands are listed below.

bMaxcompressed (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.



When the combination of bColorDepth and blmageFormat is other than the ones listed below, MaxCompressedImageSize validation is not executed even if this is set to MF_IQA_TEST_ENABLE.

bColorDepth	blmageFormat
EPS_BI_SCN_1BIT	EPS_BI_SCN_TIFF
	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF
EPS_BI_SCN_8BIT	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF

BYTE bFront_rear

This sets the execution of FrontRearImageMismatch validation. The valid commands are listed below.

bFront_rear (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bToolight

This sets the execution of ImageTooLight validation. The valid commands are listed below.

bToolight (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bToodark

This sets the execution of ImageTooDark validation. The valid commands are listed below.

bToodark (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bStreaks

This sets the execution of HorizontalStreaksPresent validation. The valid commands are listed below.

bStreaks (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bNoise

This sets the execution of ExcessiveSpotNoise validation. The valid commands are listed below.

bNoise (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.



When bColorDepth is set to EPS_BI_SCN_8BIT, ExcessiveSpotNoise validation is not executed even if this is set to MF_IQA_TEST_ENABLE.

BYTE bFocus

This sets the execution of ImageOutOfFocus validation. The valid commands are listed below.

bFocus (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.



When bColorDepth is set to EPS_BI_SCN_1BIT, ImageOutOfFocus validation is not executed even if this is set to MF_IQA_TEST_ENABLE.

BYTE bCorners

This sets the execution of FoldedTornDocCorners validation. The valid commands are listed below.

bCorners (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bEdges

This sets the execution of FoldedTornDocEdges validation. The valid commands are listed below.

bEdges (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bFraming

This sets the execution of DocFramingError validation. The valid commands are listed below.

bFraming (Constant)	Description
MF_IQA_TEST_DISABLE (Default)	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bSkew

This sets the execution of ExcessiveDocSkew validation. The valid commands are listed below.

bSkew (Constant)	Description
MF_IQA_TEST_DISABLE (Default)	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bCarbon

This sets the execution of CarbonStripDetection validation. The valid commands are listed below.

bCarbon (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bPiggyback

This sets the execution of Piggyback validation. The valid commands are listed below.

bPiggyback (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

Default Values of the MF_IQA Structure

Member of MF_IQA Structure	Default Value
MF_IQA.bErrorSelect	MF_ERROR_SELECT_NODETECT
MF_IQA.bErrorEject	MF_EJECT_MAIN_POCKET
MF_IQA.bStamp	MF_STAMP_DISABLE
MF_IQA.bCancel	MF_CANCEL_DISABLE
MF_IQA.bImageFormat	EPS_BI_SCN_TIFF
MF_IQA.bColorDepth	EPS_BI_SCN_1BIT
MF_IQA.bThreshold	0
MF_IQA.bColor	EPS_BI_SCN_MONOCHROME
MF_IQA.bExOption	EPS_BI_SCN_SHARP_CUSTOM2
MF_IQA.sResolution	MF_SCAN_DPI_DEFAULT
MF_IQA.bUndersize	MF_IQA_TEST_DISABLE
MF_IQA.bOversize	MF_IQA_TEST_DISABLE
MF_IQA.bMincompressed	MF_IQA_TEST_DISABLE
MF_IQA.bMaxcompressed	MF_IQA_TEST_DISABLE
MF_IQA.bFront_rear	MF_IQA_TEST_DISABLE
MF_IQA.bToolight	MF_IQA_TEST_DISABLE
MF_IQA.bToodark	MF_IQA_TEST_DISABLE
MF_IQA.bStreaks	MF_IQA_TEST_DISABLE
MF_IQA.bNoise	MF_IQA_TEST_DISABLE
MF_IQA.bFocus	MF_IQA_TEST_DISABLE
MF_IQA.bCorners	MF_IQA_TEST_DISABLE
MF_IQA.bEdges	MF_IQA_TEST_DISABLE
MF_IQA.bFraming	MF_IQA_TEST_DISABLE
MF_IQA.bSkew	MF_IQA_TEST_DISABLE
MF_IQA.bCarbon	MF_IQA_TEST_DISABLE
MF_IQA.bPiggyback	MF_IQA_TEST_DISABLE

MF_IQA01



IQA, Image Quality Assurance defect metrics stated by CTS2010 in India is a little bit different from that of FSTC in the USA. To support CTS2010's IQA, application should use ["MF_IQA01" on page 321](#) instead of ["MF_IQA" on page 312](#) in Win32/64. Result of IQA metrics is stored ["MF_IQA_RESULT01" on page 341](#) structure.

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    BYTE bErrorSelect;                        IN
    BYTE bErrorEject;                        IN
    BYTE bStamp;                             IN
    BYTE bCancel;                             IN
    BYTE bTestImage;                         IN
    BYTE bImageFormat;                       IN
    BYTE bColorDepth;                       IN
    CHAR bThreshold;                        IN
    BYTE bColor;                             IN
    BYTE bExOption;                         IN
    short sResolution;                      IN
    BYTE bUndersize;                        IN
    BYTE bOversize;                        IN
    BYTE bMincompressed;                   IN
    BYTE bMaxcompressed;                   IN
    BYTE bFront_rear;                      IN
    BYTE bToolight;                        IN
    BYTE bToodark;                        IN
    BYTE bStreaks;                         IN
    BYTE bNoise;                           IN
    BYTE bFocus;                           IN
    BYTE bCorners;                         IN
    BYTE bEdges;                           IN
    BYTE bFraming;                         IN
    BYTE bSkew;                             IN
    BYTE bCarbon;                           IN
    BYTE bPiggyback;                       IN
    BYTE bPartialImage;                   IN
} MF_IQA01, *LPMF_IQA01;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_IQA_VERSION.

Since the MF_IQA_VERSION value is different for each driver version, the application must always use the supplied header file.

BYTE bErrorSelect

This sets whether to detect the image quality defects (IQA error). The valid commands are listed below.

bErrorSelect (Constant)	Description
MF_ERROR_SELECT_NODETECT	No error detected
MF_ERROR_SELECT_DETECT	Error detected



- When MF_ERROR_SELECT_NODETECT is specified, no error is detected even if the insertion direction is wrong.
- When MF_ERROR_SELECT_DETECT is specified, the action at error occurrence is taken according to the following. (For details, refer to the explanation of each element.)
 - * bErrorEject
 - * bStamp
 - * bCancel

BYTE bErrorEject

This sets the ejection method for when IQA error is detected. The valid commands are listed below.

bErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket
MF_EJECT_NOEJECT	Not ejected (completed with error)



- This setting takes place only when MF_ERROR_SELECT_DETECT is specified in bErrorSelect.
- This setting has a priority even when wErrorEject of MF_BASE01 structure is set.
- In the High Speed mode, paper is ejected to the pocket corresponding to this setting value.
- In the Confirmation mode, when ["BiSetBehaviorToScnResult" on page 254](#) is not called in the MF_DATARECEIVE_DONE callback notification, paper is ejected to the pocket corresponding to this setting value.
- When reading is processed in the High Speed mode, if this setting is set to other than MF_EJECT_MAIN_POCKET, reading speed slows down.

BYTE bStamp

This sets the whether to enable a franker for when the IQA error is detected. The valid commands are listed below.

bStamp (Constant)	Description
MF_STAMP_ENABLE	Franker enabled
MF_STAMP_DISABLE	Franker disabled



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- If MF_ERROR_SELECT_NODETECT is set to bErrorSelect, the setting of bStamp is ignored.
- In the High Speed mode, operates franker in accordance with this setting value.
- In the Confirmation mode, if ["BiSetBehaviorToScnResult" on page 254](#) is not invoked in the MF_DATARECEIVE_DONE callback notification, operates franker in accordance with this setting value.
- If the setting for bSuccessStamp of MF_PROCESS01 structure differs from this setting when reading is processed in the High Speed mode, reading speed slows down.

BYTE bCancel

Sets whether to continue reading operation when the IQA error occurs.

bCancel (Constant)	Description
MF_CANCEL_DISABLE	Does not cancel the reading process for the next check sheet.
MF_CANCEL_ENABLE	Cancels the reading process for the next check sheet.



- When MF_ERROR_SELECT_NODETECT is set to bErrorSelect, the setting of bCancel is ignored.
- In the High Speed mode, operates in accordance with this setting value.
- In the Confirmation mode, the insertion direction incorrect error is notified with MF_ERROR_OCCURRED callback and the reading process is continued. If ["BiSetBehaviorToScnResult" on page 254](#) is not invoked in the MF_DATARECEIVE_DONE callback notification, operates in accordance with this setting value.
- When reading is processed in the High Speed mode, if this setting is set to other than MF_CANCEL_ENABLE, reading speed slows down.

BYTE bTestImage

Target image condition to be tested using IQA metrics. The following option can be specified.

bTestImage (Constant)	Description
MF_IQA_TESTIMAGE_BY_STRUCTURE	Image condition is specified using the following members of "MF_IQA01" on page 321 structure. <ul style="list-style-type: none"> * bImageFormat * bColorDepth * bThreshold * bColor * bExOption * sResolution
MF_IQA_TESTIMAGE_BY_FUNCTION	Target image is the one that is passed using "BiPrintImage" on page 192 before "BiGetIQAResult" on page 230 is called.

BYTE bImageFormat

This sets an image format at the IQA validation. The valid commands are listed below.

bImageFormat (Constant)	Description
EPS_BI_SCN_TIFF	TIFF format CCITT (Group 4) compressed data
EPS_BI_SCN_RASTER	Raster format uncompressed data
EPS_BI_SCN_BITMAP	Bitmap format uncompressed data
EPS_BI_SCN_TIFF256	TIFF format uncompressed data
EPS_BI_SCN_JPEGHIGH	JPEG format high compression (size priority) data
EPS_BI_SCN_JPEGNORMAL	JPEG format normal compression data
EPS_BI_SCN_JPEGLOW	JPEG format low compression (quality priority) data
EPS_BI_SCN_JTIFF	TIFF format JPEG compressed data

BYTE bColorDepth

This sets the gradation (bits per pixel) at the IQA validation. The valid commands are listed below.

bColorDepth (Constant)	Description
EPS_BI_SCN_1BIT	Black and White (1 bit)
EPS_BI_SCN_8BIT	256 grayscale (8 bit)

CHAR bThreshold

This sets the density threshold at the IQA validation.

Enabled when bColorDepth is set to EPS_BI_SCN_1BIT, and bExOption is set to EPS_BI_SCN_MANUAL. The valid value is -128 to 127.



- The value -128 to 127 corresponds to 0 to 255 of the brightness.
- When "0" is specified, the intermediate brightness "128" is applied.

BYTE bColor

This sets color at the IQA validation. The valid commands are listed below.

bColor (Constant)	Description
EPS_BI_SCN_MONOCHROME	Monochrome
EPS_BI_SCN_COLOR	Color



Even if you set EPS_BI_SCN_COLOR, the operation is the same as with EPS_BI_SCN_MONOCHROME.

BYTE bExOption

This sets the variety of density adjustment at the IQA validation. The valid commands are listed below.

bExOption (Constant)	Description
EPS_BI_SCN_MANUAL	Applies the value of bThreshold for Black and White.
EPS_BI_SCN_SHARP	Sharpening
EPS_BI_SCN_SHARP_CUSTOM	Custom sharpening
EPS_BI_SCN_SHARP_CUSTOM2	Combination of sharpening and custom sharpening

short sResolution

This sets the resolution at the IQA validation. The valid commands are listed below.

sResolution (Constant)	Description
MF_SCAN_DPI_DEFAULT	Scans at the default resolution for the device. (200 DPI)
MF_SCAN_DPI_200	Scans at 200 DPI.
MF_SCAN_DPI_300	Scans at 300 DPI.

Each of the following values can be specified independently only if EPS_BI_SCN_UNIT_CHECKPAPER is set to the unit when this structure is used. (Otherwise, it is considered to be not defined.)

sResolution (Constant)	Description
MF_SCAN_DPI_100	Scans at 100 DPI.
MF_SCAN_DPI_120	Scans at 120 DPI.
MF_SCAN_DPI_240	Scans at 240 DPI.

The following value can be specified independently only if EPS_BI_SCN_UNIT_CARD is set to the unit when this structure is used. (Otherwise, it is considered to be not defined.)

sResolution (Constant)	Description
MF_SCAN_DPI_600	Scans at 600 DPI.

BYTE bUndersize

This sets the execution of UndersizeImage validation. The valid commands are listed below.

bUndersize (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bOversize

This sets the execution of OversizeImage validation. The valid commands are listed below.

bOversize (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bMincompressed

This sets the execution of MinCompressedImageSize validation. The valid commands are listed below.

bMincompressed (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.



When the combination of bColorDepth and blmageFormat is other than the ones listed below, MinCompressedImageSize validation is not executed even if this is set to MF_IQA_TEST_ENABLE.

bColorDepth	blmageFormat
EPS_BI_SCN_1BIT	EPS_BI_SCN_TIFF
	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF
EPS_BI_SCN_8BIT	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF

BYTE bMaxcompressed

This sets the execution of MaxCompressedImageSize validation. The valid commands are listed below.

bMaxcompressed (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.



When the combination of bColorDepth and blmageFormat is other than the ones listed below, MaxCompressedImageSize validation is not executed even if this is set to MF_IQA_TEST_ENABLE.

bColorDepth	blmageFormat
EPS_BI_SCN_1BIT	EPS_BI_SCN_TIFF
	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF
EPS_BI_SCN_8BIT	EPS_BI_SCN_JPEGHIGH
	EPS_BI_SCN_JPEGNORMAL
	EPS_BI_SCN_JPEGLOW
	EPS_BI_SCN_JTIFF

BYTE bFront_rear

This sets the execution of FrontRearImageMismatch validation. The valid commands are listed below.

bFront_rear (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bToolight

This sets the execution of ImageTooLight validation. The valid commands are listed below.

bToolight (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bToodark

This sets the execution of ImageTooDark validation. The valid commands are listed below.

bToodark (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bStreaks

This sets the execution of HorizontalStreaksPresent validation. The valid commands are listed below.

bStreaks (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bNoise

This sets the execution of ExcessiveSpotNoise validation. The valid commands are listed below.

bNoise (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.



When bColorDepth is set to EPS_BI_SCN_8BIT, ExcessiveSpotNoise validation is not executed even if this is set to MF_IQA_TEST_ENABLE.

BYTE bFocus

This sets the execution of ImageOutOfFocus validation. The valid commands are listed below.

bFocus (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.



When bColorDepth is set to EPS_BI_SCN_1BIT, ImageOutOfFocus validation is not executed even if this is set to MF_IQA_TEST_ENABLE.

BYTE bCorners

This sets the execution of FoldedTornDocCorners validation. The valid commands are listed below.

bCorners (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bEdges

This sets the execution of FoldedTornDocEdges validation. The valid commands are listed below.

bEdges (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bFraming

This sets the execution of DocFramingError validation. The valid commands are listed below.

bFraming (Constant)	Description
MF_IQA_TEST_DISABLE (Default)	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bSkew

This sets the execution of ExcessiveDocSkew validation. The valid commands are listed below.

bSkew (Constant)	Description
MF_IQA_TEST_DISABLE (Default)	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bCarbon

This sets the execution of CarbonStripDetection validation. The valid commands are listed below.

bCarbon (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bPiggyback

This sets the execution of Piggyback validation. The valid commands are listed below.

bPiggyback (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

BYTE bPartialImage

This sets the execution of PartialImage validation. The valid commands are listed below.

bPartialImage (Constant)	Description
MF_IQA_TEST_DISABLE	Validation is not executed.
MF_IQA_TEST_ENABLE	Validation is executed.

Default Values of the MF_IQA01 Structure

Member of MF_IQA01 Structure	Default Value
MF_IQA01.bErrorSelect	MF_ERROR_SELECT_NODETECT
MF_IQA01.bErrorEject	MF_EJECT_MAIN_POCKET
MF_IQA01.bStamp	MF_STAMP_DISABLE
MF_IQA01.bCancel	MF_CANCEL_DISABLE
MF_IQA01.bTestImage	MF_IQA_TESTIMAGE_BY_STRUCTURE
MF_IQA01.bImageFormat	EPS_BI_SCN_TIFF
MF_IQA01.bColorDepth	EPS_BI_SCN_1BIT
MF_IQA01.bThreshold	0
MF_IQA01.bColor	EPS_BI_SCN_MONOCHROME
MF_IQA01.bExOption	EPS_BI_SCN_SHARP_CUSTOM2
MF_IQA01.sResolution	MF_SCAN_DPI_DEFAULT
MF_IQA01.bUndersize	MF_IQA_TEST_DISABLE
MF_IQA01.bOversize	MF_IQA_TEST_DISABLE
MF_IQA01.bMincompressed	MF_IQA_TEST_DISABLE
MF_IQA01.bMaxcompressed	MF_IQA_TEST_DISABLE
MF_IQA01.bFront_rear	MF_IQA_TEST_DISABLE
MF_IQA01.bToolight	MF_IQA_TEST_DISABLE
MF_IQA01.bToodark	MF_IQA_TEST_DISABLE
MF_IQA01.bStreaks	MF_IQA_TEST_DISABLE
MF_IQA01.bNoise	MF_IQA_TEST_DISABLE
MF_IQA01.bFocus	MF_IQA_TEST_DISABLE
MF_IQA01.bCorners	MF_IQA_TEST_DISABLE
MF_IQA01.bEdges	MF_IQA_TEST_DISABLE
MF_IQA01.bFraming	MF_IQA_TEST_DISABLE
MF_IQA01.bSkew	MF_IQA_TEST_DISABLE
MF_IQA01.bCarbon	MF_IQA_TEST_DISABLE
MF_IQA01.bPiggyback	MF_IQA_TEST_DISABLE
MF_IQA01.bPartialImage	MF_IQA_TEST_DISABLE

MF_IQA_RESULT

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    int iRet;                                 OUT
    IQARESULT_UNDERSIZE_IMAGE stUnderSize;    OUT
    IQARESULT_OVERSIZE_IMAGE stOverSize;     OUT
    IQARESULT_MIN_COMPRESSED_IMAGE_SIZE stMinCompressedImageSize; OUT
    IQARESULT_MAX_COMPRESSED_IMAGE_SIZE stMaxCompressedImageSize; OUT
    IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatch; OUT
    IQARESULT_IMAGE_TOO_LIGHT stImageTooLight; OUT
    IQARESULT_IMAGE_TOO_DARK stImageTooDark; OUT
    IQARESULT_HORIZONTAL_STREAKS_PRESENT stHorizontalStreaksPresent; OUT
    IQARESULT_EXCESSIVE_SPOT_NOISE stExcessiveSpotNoise; OUT
    IQARESULT_IMAGE_OUT_OF_FOCUS stImageOutOfFocus; OUT
    IQARESULT_FOLDED_TORN_DOC_CORNERS stFoldedTornDocCorners; OUT
    IQARESULT_FOLDED_TORN_DOC_EDGES stFoldedTornDocEdges; OUT
    IQARESULT_DOC_FRAMING_ERROR stDocFramingError; OUT
    IQARESULT_EXCESSIVE_DOC_SKEW stExcessiveDocSkew; OUT
    IQARESULT_CARBON_STRIP_DETECTION stCarbonStripDetection; OUT
    IQARESULT_PIGGYBACK stPiggyBack;         OUT
} MF_IQA_RESULT, *LPMF_IQA_RESULT;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_IQARESULT_VERSION.

Since the MF_IQARESULT_VERSION value is different for each driver version, the application must always use the supplied header file.

int iRet

This sets the return values for running IQA.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_SCAN	-450	Device failed in image scanning
ERR_SCN_IQA	-1120	Error is detected by the IQA validation

IQARESULT_UNDERSIZE_IMAGE stUnderSize

UndersizeImage validation result is set.

```
typedef struct tag_IQA_UNDERSIZE_IMAGE {  
    BYTE bResult;                                OUT  
    int iWidth;                                  OUT  
    int iHeight;                                 OUT  
} IQARESULT_UNDERSIZE_IMAGE, *LPIQARESULT_UNDERSIZE_IMAGE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iWidth
Image width (unit: 0.1 inch)
- int iHeight
Image height (unit: 0.1 inch)

IQARESULT_OVERSIZE_IMAGE stOverSize

OversizeImage validation result is set.

```
typedef struct tag_IQA_OVERSIZE_IMAGE {  
    BYTE bResult;                                OUT  
    int iWidth;                                  OUT  
    int iHeight;                                 OUT  
} IQARESULT_OVERSIZE_IMAGE, *LPIQARESULT_OVERSIZE_IMAGE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iWidth
Image width (unit: 0.1 inch)
- int iHeight
Image height (unit: 0.1 inch)

IQARESULT_MIN_COMPRESSED_IMAGE_SIZE stMinCompressedImageSize

MinCompressedImageSize validation result is set.

```
typedef struct tag_IQA_MIN_COMPRESSED_IMAGE_SIZE {
    BYTE bResult;                                OUT
    int iSize;                                    OUT
} IQARESULT_MIN_COMPRESSED_IMAGE_SIZE, *LPIQARESULT_MIN_COMPRESSED_IMAGE_SIZE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iSize

Compressed image size in bytes.

IQARESULT_MAX_COMPRESSED_IMAGE_SIZE stMaxCompressedImageSize

MaxCompressedImageSize validation result is set.

```
typedef struct tag_IQA_MAX_COMPRESSED_IMAGE_SIZE {
    BYTE bResult;                                OUT
    int iSize;                                    OUT
} IQARESULT_MAX_COMPRESSED_IMAGE_SIZE, *LPIQARESULT_MAX_COMPRESSED_IMAGE_SIZE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iSize

Compressed image size in bytes.

IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatch

FrontRearImageMismatch validation result is set.

```
typedef struct tag_IQA_FRONT_REAR_IMAGE_MISMATCH {
    BYTE bResult;                                OUT
    int iAbsWidthDiff;                            OUT
    int iAbsHeightDiff;                          OUT
} IQARESULT_FRONT_REAR_IMAGE_MISMATCH, *LPIQARESULT_FRONT_REAR_IMAGE_MISMATCH;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iWidth

Width difference between the front-side and back-side images (unit: 0.1 inch)

- int iHeight

Height difference between the front-side and back-side images (unit: 0.1 inch)

IQARESULT_IMAGE_TOO_LIGHT stImageTooLight;

ImageTooLight validation result is set.

```
typedef struct tag_IQA_IMAGE_TOO_LIGHT {
    BYTE bResult;                                OUT
    int iBlackPixels;                            OUT
    int iBrightness;                            OUT
    int iContrast;                              OUT
} IQARESULT_IMAGE_TOO_LIGHT, *LPIQARESULT_IMAGE_TOO_LIGHT;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iBlackPixels

Percentage of black pixels in the image (unit: 0.1%)

- int iBrightness

Image brightness (unit: 0.1%)

- int iContrast

Image contrast (unit: 0.1%)

IQARESULT_IMAGE_TOO_DARK stImageTooDark;

ImageTooDark validation result is set.

```
typedef struct tag_IQA_IMAGE_TOO_DARK {
    BYTE bResult;                                OUT
    int iBlackPixels;                            OUT
    int iBrightness;                             OUT
} IQARESULT_IMAGE_TOO_DARK, *LPIQARESULT_IMAGE_TOO_DARK;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iBlackPixels
Percentage of black pixels in the image (unit: 0.1%)
- int iBrightness
Image brightness (unit: 0.1%)

IQARESULT_HORIZONTAL_STREAKS_PRESENT stHorizontalStreaksPresent

HorizontalStreaksPresent validation result is set.

```
typedef struct tag_IQA_HORIZONTAL_STREAKS_PRESENT {
    BYTE bResult;                                OUT
    int iStreakCount;                            OUT
    int iStreakHeight;                           OUT
} IQARESULT_HORIZONTAL_STREAKS_PRESENT, *LPIQARESULT_HORIZONTAL_STREAKS_PRESENT;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iStreakCount
Number of black stripes present in binary images
- int iStreakHeight
Width of the thickest black stripe

IQARESULT_EXCESSIVE_SPOT_NOISE stExcessiveSpotNoise

ExcessiveSpotNoise validation result is set.

```
typedef struct tag_IQA_EXCESSIVE_SPOT_NOISE {
    BYTE bResult;                                OUT
    int iCount;                                  OUT
} IQARESULT_EXCESSIVE_SPOT_NOISE, *LPIQARESULT_EXCESSIVE_SPOT_NOISE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iCount

Mean number of spots (noise) per square inch

IQARESULT_IMAGE_OUT_OF_FOCUS stImageOutOfFocus

ImageOutOfFocus validation result is set.

```
typedef struct tag_IQA_IMAGE_OUT_OF_FOCUS {
    BYTE bResult;                                OUT
    int iImageFocusScore;                        OUT
} IQARESULT_IMAGE_OUT_OF_FOCUS, *LPIQARESULT_IMAGE_OUT_OF_FOCUS;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iImageFocusScore

Score estimated by (max video gradient) / (grey level dynamic range) * (pixel pitch)

IQARESULT_FOLDED_TORN_DOC_CORNERS stFoldedTornDocCorners

FoldedTornDocCorners validation result is set.

```
typedef struct tag_IQA_FOLDED_TORN_DOC_CORNERS {
    BYTE bResult;
    int iTopLeftWidth;
    int iTopLeftHeight;
    int iTopRightWidth;
    int iTopRightHeight;
    int iBottomLeftWidth;
    int iBottomLeftHeight;
    int iBottomRightWidth;
    int iBottomRightHeight;
} IQARESULT_FOLDED_TORN_DOC_CORNERS, *LPIQARESULT_FOLDED_TORN_DOC_CORNERS;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iTopLeftWidth
Width of a tear/bend at the upper left corner of the image (unit: 0.1 inch)
- int iTopLeftHeight
Height of a tear/bend at the upper left corner of the image (unit: 0.1 inch)
- int iTopRightWidth
Width of a tear/bend at the upper right corner of the image (unit: 0.1 inch)
- int iTopRightHeight
Height of a tear/bend at the upper right corner of the image (unit: 0.1 inch)
- int iBottomLeftWidth
Width of a tear/bend at the lower left corner of the image (unit: 0.1 inch)
- int iBottomLeftHeight
Height of a tear/bend at the lower left corner of the image (unit: 0.1 inch)
- int iBottomRightWidth
Width of a tear/bend at the lower right corner of the image (unit: 0.1 inch)
- int iBottomRightHeight
Height of a tear/bend at the lower right corner of the image (unit: 0.1 inch)

IQARESULT_FOLDED_TORN_DOC_EDGES stFoldedTornDocEdges

FoldedTornDocEdges validation result is set.

```
typedef struct tag_IQA_FOLDED_TORN_DOC_EDGES {
    BYTE bResult;
    int iTopWidth;
    int iTopHeight;
    int iLeftWidth;
    int iLeftHeight;
    int iRightWidth;
    int iRightHeight;
    int iBottomWidth;
    int iBottomHeight;
} IQARESULT_FOLDED_TORN_DOC_EDGES, *LPIQARESULT_FOLDED_TORN_DOC_EDGES;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- iTopWidth
Top width of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iTopHeight
Top height of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iLeftWidth
Left width of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iLeftHeight
Left height of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iRightWidth
Right width of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iRightHeight
Right height of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iBottomWidth
Bottom width of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iBottomHeight
Bottom height of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)

IQARESULT_DOC_FRAMING_ERROR stDocFramingError

DocFramingError validation result is set.

```
typedef struct tag_IQA_DOC_FRAMING_ERROR {
    BYTE bResult;                                OUT
    int iTop;                                    OUT
    int iLeft;                                   OUT
    int iRight;                                 OUT
    int iBottom;                                OUT
} IQARESULT_DOC_FRAMING_ERROR, *LPIQARESULT_DOC_FRAMING_ERROR;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iTop
Width of the top margin of the image (unit: 0.1 inch)
- int iLeft
Width of the left margin of the image (unit: 0.1 inch)
- int iRight
Width of the right margin of the image (unit: 0.1 inch)
- int iBottom
Width of the bottom margin of the image (unit: 0.1 inch)

IQARESULT_EXCESSIVE_DOC_SKEW stExcessiveDocSkew

ExcessiveDocSkew validation result is set.

```
typedef struct tag_PIQA_EXCESSIVE_DOC_SKEW {
    BYTE bResult;                                OUT
    int iAngle;                                  OUT
    int iRange;                                  OUT
} IQARESULT_EXCESSIVE_DOC_SKEW, *LPIQARESULT_EXCESSIVE_DOC_SKEW;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iAngle
Tilt angle (unit: 0.1 degree)
- int iRange
Fixed to 0: Value indicating that iAngle is in the range from -90 to +90

IQARESULT_CARBON_STRIP_DETECTION stCarbonStripDetection

CarbonStripDetection validation result is set.

```
typedef struct tag_IQA_CARBON_STRIP_DETECTION {
    BYTE bResult;                                OUT
    int iStripHeight;                            OUT
} IQARESULT_CARBON_STRIP_DETECTION, *LPIQARESULT_CARBON_STRIP_DETECTION;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iStripHeight

Length of a carbon strip (unit: 0.1 inch)

IQARESULT_PIGGYBACK stPiggyBack

Piggyback validation result is set.

```
typedef struct tag_IQA_PIGGYBACK {
    BYTE bResult;                                OUT
} IQARESULT_PIGGYBACK, *LPIQARESULT_PIGGYBACK;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

Regarding the result for both sides

The MF_IQA_RESULT structure only returns the result of the front side or the back side. Regarding items for verification of both sides, under the following rules, it returns the result for the front side or the back side.

Verification item of both sides	Result (Front side)	Result (Back side)	Returns result
Compressed Image Too Small (page 332)	OK	OK	Results of the back side
		NG	Results of the back side
	NG	OK	Results of the front side
		NG	Results of the front side
Compressed Image Too big (page 332)	OK	OK	Results of the back side
		NG	Results of the back side
	NG	OK	Results of the front side
		NG	Results of the front side
Too Light (page 333)	OK	OK	Results of the back side
		NG	Results of the back side
	NG	OK	Results of the front side
		NG	Results of the front side
Too Dark (page 334)	OK	OK	Results of the back side
		NG	Results of the back side
	NG	OK	Results of the front side
		NG	Results of the front side
Horizontal Streaks (page 334)	OK	OK	Results of the back side
		NG	Results of the back side
	NG	OK	Results of the front side
		NG	Results of the front side

MF_IQA_RESULT01

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    int iRet;                                 OUT
    IQARESULT_UNDERSIZE_IMAGE stUnderSize;    OUT
    IQARESULT_UNDERSIZE_IMAGE stUnderSizeHorizontal;    OUT
    IQARESULT_UNDERSIZE_IMAGE stUnderSizeVertical;    OUT
    IQARESULT_OVERSIZE_IMAGE stOverSize;    OUT
    IQARESULT_OVERSIZE_IMAGE stOverSizeHorizontal;    OUT
    IQARESULT_OVERSIZE_IMAGE stOverSizeVertical;    OUT
    IQARESULT_MIN_COMPRESSED_IMAGE_SIZE stMinCompressedImageSize;    OUT
    IQARESULT_MAX_COMPRESSED_IMAGE_SIZE stMaxCompressedImageSize;    OUT
    IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatch;    OUT
    IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatchHorizontal;    OUT
    IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatchVertical;    OUT
    IQARESULT_IMAGE_TOO_LIGHT stImageTooLight;    OUT
    IQARESULT_IMAGE_TOO_DARK stImageTooDark;    OUT
    IQARESULT_HORIZONTAL_STREAKS_PRESENT stHorizontalStreaksPresent;    OUT
    IQARESULT_EXCESSIVE_SPOT_NOISE stExcessiveSpotNoise;    OUT
    IQARESULT_IMAGE_OUT_OF_FOCUS stImageOutOfFocus;    OUT
    IQARESULT_FOLDED_TORN_DOC_CORNERS stFoldedTornDocCorners;    OUT
    IQARESULT_FOLDED_TORN_DOC_CORNERS stFoldedDocCorners;    OUT
    IQARESULT_FOLDED_TORN_DOC_CORNERS stTornDocCorners;    OUT
    IQARESULT_FOLDED_TORN_DOC_EDGES stFoldedTornDocEdges;    OUT
    IQARESULT_DOC_FRAMING_ERROR stDocFramingError;    OUT
    IQARESULT_EXCESSIVE_DOC_SKEW stExcessiveDocSkew;    OUT
    IQARESULT_CARBON_STRIP_DETECTION stCarbonStripDetection;    OUT
    IQARESULT_PIGGYBACK stPiggyBack;    OUT
    IQARESULT_PARTIAL_IMAGE stPartialImage;    OUT
} MF_IQA_RESULT01, *LPMF_IQA_RESULT01;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_IQARESULT_VERSION.

Since the MF_IQARESULT_VERSION value is different for each driver version, the application must always use the supplied header file.

int iRet

This sets the return values for running IQA.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_SCAN	-450	Device failed in image scanning
ERR_SCN_IQA	-1120	Error is detected by the IQA validation

IQARESULT_UNDERSIZE_IMAGE stUnderSize

UndersizeImage validation result is set.

```
typedef struct tag_IQA_UNDERSIZE_IMAGE {
    BYTE bResult;                                OUT
    int iWidth;                                  OUT
    int iHeight;                                 OUT
} IQARESULT_UNDERSIZE_IMAGE, *LPIQARESULT_UNDERSIZE_IMAGE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iWidth
Image width (unit: 0.1 inch)
- int iHeight
Image height (unit: 0.1 inch)

IQARESULT_UNDERSIZE_IMAGE stUnderSizeHorizontal

Below Minimum Image Length test result is set to bResult of ["IQARESULT_UNDERSIZE_IMAGE stUnderSize"](#) on page 342.

IQARESULT_UNDERSIZE_IMAGE stUnderSizeVertical

Below Minimum Image Height test result is set to bResult of ["IQARESULT_UNDERSIZE_IMAGE stUnderSize"](#) on page 342.

IQARESULT_OVERSIZE_IMAGE stOverSize

OversizeImage validation result is set.

```
typedef struct tag_IQA_OVERSIZE_IMAGE {  
    BYTE bResult;                                OUT  
    int iWidth;                                  OUT  
    int iHeight;                                 OUT  
} IQARESULT_OVERSIZE_IMAGE, *LPIQARESULT_OVERSIZE_IMAGE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iWidth
Image width (unit: 0.1 inch)
- int iHeight
Image height (unit: 0.1 inch)

IQARESULT_OVERSIZE_IMAGE stOverSizeHorizontal

Exceeds Maximum Image Length test result is set to bResult of "[IQARESULT_OVERSIZE_IMAGE stOverSize](#)" on page 343.

IQARESULT_OVERSIZE_IMAGE stOverSizeVertical

Exceeds Maximum Image Height test result is set to bResult of "[IQARESULT_OVERSIZE_IMAGE stOverSize](#)" on page 343.

IQARESULT_MIN_COMPRESSED_IMAGE_SIZE stMinCompressedImageSize

MinCompressedImageSize validation result is set.

```
typedef struct tag_IQA_MIN_COMPRESSED_IMAGE_SIZE {
    BYTE bResult;                                OUT
    int iSize;                                    OUT
} IQARESULT_MIN_COMPRESSED_IMAGE_SIZE, *LPIQARESULT_MIN_COMPRESSED_IMAGE_SIZE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iSize

Compressed image size in bytes.

IQARESULT_MAX_COMPRESSED_IMAGE_SIZE stMaxCompressedImageSize

MaxCompressedImageSize validation result is set.

```
typedef struct tag_IQA_MAX_COMPRESSED_IMAGE_SIZE {
    BYTE bResult;                                OUT
    int iSize;                                    OUT
} IQARESULT_MAX_COMPRESSED_IMAGE_SIZE, *LPIQARESULT_MAX_COMPRESSED_IMAGE_SIZE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iSize

Compressed image size in bytes.

IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatch

FrontRearImageMismatch validation result is set.

```
typedef struct tag_IQA_FRONT_REAR_IMAGE_MISMATCH {  
    BYTE bResult;                                OUT  
    int iAbsWidthDiff;                            OUT  
    int iAbsHeightDiff;                          OUT  
} IQARESULT_FRONT_REAR_IMAGE_MISMATCH, *LPIQARESULT_FRONT_REAR_IMAGE_MISMATCH;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iWidth
Width difference between the front-side and back-side images (unit: 0.1 inch)
- int iHeight
Height difference between the front-side and back-side images (unit: 0.1 inch)

IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatchHorizontal

Image Length Mismatch test result is set to bResult of "[IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatch](#)" on page 345.

IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatchVertical

Image Height Mismatch test result is set to bResult of "[IQARESULT_FRONT_REAR_IMAGE_MISMATCH stFrontRearImageMismatch](#)" on page 345.

IQARESULT_IMAGE_TOO_LIGHT stImageTooLight

ImageTooLight validation result is set.

```
typedef struct tag_IQA_IMAGE_TOO_LIGHT {
    BYTE bResult;                                OUT
    int iBlackPixels;                            OUT
    int iBrightness;                             OUT
} IQARESULT_IMAGE_TOO_LIGHT, *LPIQARESULT_IMAGE_TOO_LIGHT;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iBlackPixels
Percentage of black pixels in the image (unit: 0.1%)
- int iBrightness
Image brightness (unit: 0.1%)

IQARESULT_IMAGE_TOO_DARK stImageTooDark

ImageTooDark validation result is set.

```
typedef struct tag_IQA_IMAGE_TOO_DARK {
    BYTE bResult;                                OUT
    int iBlackPixels;                            OUT
    int iBrightness;                             OUT
    int iContrast;                               OUT
} IQARESULT_IMAGE_TOO_DARK, *LPIQARESULT_IMAGE_TOO_DARK;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iBlackPixels
Percentage of black pixels in the image (unit: 0.1%)
- int iBrightness
Image brightness (unit: 0.1%)
- int iContrast
Image contrast (unit: 0.1%)

IQARESULT_HORIZONTAL_STREAKS_PRESENT stHorizontalStreaksPresent

HorizontalStreaksPresent validation result is set.

```
typedef struct tag_IQA_HORIZONTAL_STREAKS_PRESENT {
    BYTE bResult;                                OUT
    int iStreakCount;                            OUT
    int iStreakHeight;                          OUT
} IQARESULT_HORIZONTAL_STREAKS_PRESENT, *LPIQARESULT_HORIZONTAL_STREAKS_PRESENT;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iStreakCount
Number of black stripes present in binary images
- int iStreakHeight
Width of the thickest black stripe

IQARESULT_EXCESSIVE_SPOT_NOISE stExcessiveSpotNoise

ExcessiveSpotNoise validation result is set.

```
typedef struct tag_IQA_EXCESSIVE_SPOT_NOISE {
    BYTE bResult;                                OUT
    int iCount;                                OUT
} IQARESULT_EXCESSIVE_SPOT_NOISE, *LPIQARESULT_EXCESSIVE_SPOT_NOISE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iCount
Mean number of spots (noise) per square inch

IQARESULT_IMAGE_OUT_OF_FOCUS stImageOutOfFocus

ImageOutOfFocus validation result is set.

```
typedef struct tag_IQA_IMAGE_OUT_OF_FOCUS {
    BYTE bResult;
    int iImageFocusScore;
} IQARESULT_IMAGE_OUT_OF_FOCUS, *LPIQARESULT_IMAGE_OUT_OF_FOCUS;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iImageFocusScore

Score estimated by (max video gradient) / (grey level dynamic range) * (pixel pitch)

IQARESULT_FOLDED_TORN_DOC_CORNERS stFoldedTornDocCorners

FoldedTornDocCorners validation result is set.

```
typedef struct tag_IQA_FOLDED_TORN_DOC_CORNERS {
    BYTE bResult;
    int iTopLeftWidth;
    int iTopLeftHeight;
    int iTopRightWidth;
    int iTopRightHeight;
    int iBottomLeftWidth;
    int iBottomLeftHeight;
    int iBottomRightWidth;
    int iBottomRightHeight;
} IQARESULT_FOLDED_TORN_DOC_CORNERS, *LPIQARESULT_FOLDED_TORN_DOC_CORNERS;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iTopLeftWidth

Width of a tear/bend at the upper left corner of the image (unit: 0.1 inch)

- int iTopLeftHeight

Height of a tear/bend at the upper left corner of the image (unit: 0.1 inch)

- int iTopRightWidth

Width of a tear/bend at the upper right corner of the image (unit: 0.1 inch)

- `int iTopRightHeight`
Height of a tear/bend at the upper right corner of the image (unit: 0.1 inch)
- `int iBottomLeftWidth`
Width of a tear/bend at the lower left corner of the image (unit: 0.1 inch)
- `int iBottomLeftHeight`
Height of a tear/bend at the lower left corner of the image (unit: 0.1 inch)
- `int iBottomRightWidth`
Width of a tear/bend at the lower right corner of the image (unit: 0.1 inch)
- `int iBottomRightHeight`
Height of a tear/bend at the lower right corner of the image (unit: 0.1 inch)

IQARESULT_FOLDED_TORN_DOC_CORNERS stFoldedDocCorners

Bent Corner test result is set to `bResult` of "[IQARESULT_FOLDED_TORN_DOC_CORNERS stFolded-TornDocCorners](#)" on page 348.

IQARESULT_FOLDED_TORN_DOC_CORNERS stTornDocCorners

Torn Corner test result is set to `bResult` of "[IQARESULT_FOLDED_TORN_DOC_CORNERS stFolded-TornDocCorners](#)" on page 348.

IQARESULT_FOLDED_TORN_DOC_EDGES stFoldedTornDocEdges

`FoldedTornDocEdges` validation result is set.

```
typedef struct tag_IQA_FOLDED_TORN_DOC_EDGES {
    BYTE bResult;                                OUT
    int iTopWidth;                                OUT
    int iTopHeight;                               OUT
    int iLeftWidth;                               OUT
    int iLeftHeight;                              OUT
    int iRightWidth;                              OUT
    int iRightHeight;                             OUT
    int iBottomWidth;                             OUT
    int iBottomHeight;                            OUT
} IQARESULT_FOLDED_TORN_DOC_EDGES, *LPIQARESULT_FOLDED_TORN_DOC_EDGES;
```

- `BYTE bResult`

Test results:

bResult (Constant)	Description
<code>IQARESULT_NOT_TESTED</code>	The test has not been performed.
<code>IQARESULT_PASS</code>	The test was passed.
<code>IQARESULT_NOT_PASS</code>	The test was failed.

- `iTopWidth`
Top width of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- `iTopHeight`
Top height of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)

- iLeftWidth
Left width of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iLeftHeight
Left height of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iRightWidth
Right width of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iRightHeight
Right height of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iBottomWidth
Bottom width of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)
- iBottomHeight
Bottom height of circumscribing rectangle of torn/folded edge (unit: 0.1 inch)

IQARESULT_DOC_FRAMING_ERROR stDocFramingError

DocFramingError validation result is set.

```
typedef struct tag_IQA_DOC_FRAMING_ERROR {  
    BYTE bResult;                                OUT  
    int iTop;                                    OUT  
    int iLeft;                                   OUT  
    int iRight;                                  OUT  
    int iBottom;                                 OUT  
} IQARESULT_DOC_FRAMING_ERROR, *LPIQARESULT_DOC_FRAMING_ERROR;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iTop
Width of the top margin of the image (unit: 0.1 inch)
- int iLeft
Width of the left margin of the image (unit: 0.1 inch)
- int iRight
Width of the right margin of the image (unit: 0.1 inch)
- int iBottom
Width of the bottom margin of the image (unit: 0.1 inch)

IQARESULT_EXCESSIVE_DOC_SKEW stExcessiveDocSkew

ExcessiveDocSkew validation result is set.

```
typedef struct tag_PIQA_EXCESSIVE_DOC_SKEW {
    BYTE bResult;                                OUT
    int iAngle;                                  OUT
    int iRange;                                  OUT
} IQARESULT_EXCESSIVE_DOC_SKEW, *LPIQARESULT_EXCESSIVE_DOC_SKEW;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iAngle

Tilt angle (unit: 0.1 degree)

- int iRange

Fixed to 0: Value indicating that iAngle is in the range from -90 to +90

IQARESULT_CARBON_STRIP_DETECTION stCarbonStripDetection

CarbonStripDetection validation result is set.

```
typedef struct tag_IQA_CARBON_STRIP_DETECTION {
    BYTE bResult;                                OUT
    int iStripHeight;                            OUT
} IQARESULT_CARBON_STRIP_DETECTION, *LPIQARESULT_CARBON_STRIP_DETECTION;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iStripHeight

Length of a carbon strip (unit: 0.1 inch)

IQARESULT_PIGGYBACK stPiggyBack

Piggyback validation result is set.

```
typedef struct tag_IQA_PIGGYBACK {
    BYTE bResult;
} IQARESULT_PIGGYBACK, *LPIQARESULT_PIGGYBACK;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

IQARESULT_PARTIAL_IMAGE stPartialImage

Partial Image test result is set to bResult of IQARESULT_PARTIAL_IMAGE.

```
typedef struct {
    BYTE bResult;
    int iLostPercentage;
} IQARESULT_PARTIAL_IMAGE, *LPIQARESULT_PARTIAL_IMAGE;
```

- BYTE bResult

Test results:

bResult (Constant)	Description
IQARESULT_NOT_TESTED	The test has not been performed.
IQARESULT_PASS	The test was passed.
IQARESULT_NOT_PASS	The test was failed.

- int iLostPercentage

Percentage of area of the image detected to be missing (unit: %)

MF_BARCODE

```
typedef struct {
    int iSize;
    int iVersion;
    int iRet;
    BYTE bErrorSelect;
    BYTE bErrorEject;
    BYTE bStamp;
    BYTE bCancel;
    DWORD dwTargetColor;
    short sResolution;
    DWORD dwInfoMode;
    BARCODE_INFO stInfo[5];
    BYTE bDataCount;
    LPVOID lpData;
} MF_BARCODE, *LPMF_BARCODE;
```

int iSize

This is the size of this structure.

int iVersion

This is the structure version. Always specify MF_BARCODE_VERSION.
Since the MF_BARCODE_VERSION value is different for each driver version, the application must always use the supplied header file.

int iRet

This sets the return values for barcode decoding. Refer to ["MF_PRINT01.iRet" on page 112](#) for details.

BYTE bErrorSelect

Specifies whether to detect the barcode decode error. The valid commands are listed below.

bErrorSelect (Constant)	Description
MF_ERROR_SELECT_NODETECT	No error detected
MF_ERROR_SELECT_DETECT	Error detected



When MF_ERROR_SELECT_NODETECT is specified, the follow settings is ignored.

- * bErrorEject
- * bStamp
- * bCancel

BYTE bErrorEject

Specifies where to eject when the barcode decode error is detected. The valid commands are listed below.

bErrorEject (Constant)	Description
MF_EJECT_MAIN_POCKET	Ejected to the main pocket
MF_EJECT_SUB_POCKET	Ejected to the sub pocket
MF_EJECT_NOEJECT	Not ejected (completed with error)



- If MF_ERROR_SELECT_NODETECT is set to bErrorSelect, the setting of bErrorEject is ignored.
- In the Confirmation mode, this setting is applied when "[BiSetBehaviorToScnResult](#)" on page 254 cannot be called back in the MF_DATARECEIVE_DONE callback process. Reading speed decreases if bActivationMode of MF_PROCESS01 structure is set, and this setting is set to other than MF_EJECT_MAIN_POCKET.
- When reading is processed in the High Speed mode, if this setting is set to other than MF_EJECT_MAIN_POCKET, reading speed slows down.

BYTE bStamp

Specifies whether to do the franking process to the sheet when the barcode decode error is detected. The valid commands are listed below.

bStamp (Constant)	Description
MF_STAMP_ENABLE	Franker enabled
MF_STAMP_DISABLE	Franker disabled



For TM-S9000/S2000 API, regardless of this value, the action is always not to stamp.



- If MF_ERROR_SELECT_NODETECT is set to bErrorSelect, the setting of bStamp is ignored.
- In the High Speed mode, operates franker in accordance with this setting value.
- In the Confirmation mode, if "[BiSetBehaviorToScnResult](#)" on page 254 is not invoked in the MF_DATARECEIVE_DONE callback notification, operates franker in accordance with this setting value.
- If the setting for bSuccessStamp of MF_PROCESS01 structure differs from this setting when reading is processed in the High Speed mode, reading speed slows down.

BYTE bCancel

Specifies whether to continue the reading process when the barcode decode error is detected. The valid commands are listed below.

bCancel (Constant)	Description
MF_CANCEL_DISABLE	Does not cancel the reading process for the next check sheet.
MF_CANCEL_ENABLE	Cancels the reading process for the next check sheet.



- When MF_ERROR_SELECT_NODETECT is set to bErrorSelect, the setting of bCancel is ignored.
- In the High Speed mode, operates in accordance with this setting value.
- In the Confirmation mode, the insertion direction incorrect error is notified with MF_ERROR_OCCURRED callback and the reading process is continued. If ["BiSetBehaviorToScnResult" on page 254](#) is not invoked in the MF_DATARECEIVE_DONE callback notification, operates in accordance with this setting value.
- When reading is processed in the High Speed mode, if this setting is set to other than MF_CANCEL_ENABLE, reading speed slows down.

DWORD dwTargetColor

Specifies the target color for barcode decoding. The valid commands are listed below.

dwTargetColor (Constant)	Description
BARCODE_TARGET_COLOR_GRAY	Decode as grayscale

short sResolution

Specifies the image resolution of image data. The valid commands are listed below.

sResolution (Constant)	Description
MF_SCAN_DPI_DEFAULT	Scans at the default resolution for the device. (200 DPI)
MF_SCAN_DPI_200	Scans at 200 DPI.

DWORD dwInfoMode

Specify the stInfo sequence.

dwInfoMode	Description
0	Regions are not specified according to stInfo. If 0 is specified, the only stInfo[0] elements referred to are dwSymbolMask and bDirection.
1 to 5	Specification of regions according to stInfo is enabled. By specifying in dwInfoMode the number of items in the BARCODE_INFO structure to enable, it is possible to make multiple settings related to decoding. <Example> If 3 is specified in dwInfoMode, stInfo[0], stInfo[1], and stInfo[2] are referenced, and stInfo[3] and stInfo[4] are ignored.

BARCODE_INFO stInfo[5]

This is the structure where the decode setting and the status of decode result are made. Five arrangements are available. Refer to ["BARCODE_INFO structure" on page 356](#) for details.

BYTE bDataCount

The number of scanned barcodes is set.

LPVOID lpData

The result of barcode decoding (BARCODE_DATA structure) is set.

The BARCODE_DATA structure is an element of arrangement. The number of elements is the number of read barcodes (bDataCount). If there is no decode result of barcode, NULL is set. Refer to ["MF_PRINT01.iRet" on page 112](#) for details.

BARCODE_INFO structure

```
typedef struct {
    int iRet;                                OUT
    BYTE bStatus;                            OUT
    BYTE bDetail;                            OUT
    DWORD dwSymbolMask;                      IN
    BYTE bDirection;                         IN
    BYTE bOrigin;                            IN
    WORD wStartX;                            IN
    WORD wStartY;                            IN
    WORD wEndX;                              IN
    WORD wEndY;                              IN
} BARCODE_INFO, *LPBARCODE_INFO;
```

int iRet

The barcode decode execution result is set. Refer to ["MF_PRINT01.iRet" on page 112](#) for details.

BYTE bStatus

The status of decoding result is set.

Bit	Status Contents	ON / OFF	Value	Status
0	---	---	0x0000	Reserved (Fixed to 0)
1	---	---	0x0000	Reserved (Fixed to 0)
2	---	---	0x0000	Reserved (Fixed to 0)
3	Detailed information	ON	0x0008	Added
		OFF	0x0000	-
4	---	---	0x0000	Reserved (Fixed to 0)
5	Reading result	ON	0x0020	Failure end
		OFF	0x0000	Success
6	---	---	0x0000	Reserved (Fixed to 0)
7	---	---	0x0000	Reserved (Fixed to 0)

BYTE bDetail

The detailed status of decoding result is set.

Value	Information
40h	Success
45h	Barcode cannot be detected.

DWORD dwSymbolMask

Specifies the type of barcode symbol to decode. Make sure to set this because the default value is not set for this setting.

dwSymbolMask (Constant)	Barcode Type
MF_BARCODE_SYMBOL_CODABAR	Codabar
MF_BARCODE_SYMBOL_CODE128	Code128
MF_BARCODE_SYMBOL_CODE39	Code39
MF_BARCODE_SYMBOL_ITF	ITF
MF_BARCODE_SYMBOL_EAN_JAN	JAN13(EAN), JAN8(EAN)
MF_BARCODE_SYMBOL_UPC_A	UPC-A
MF_BARCODE_SYMBOL_UPC_E	UPC-E

A combination of multiple barcode symbols can be set.

<Example: When specifying Codabar and ITF>

```
MF_BARCODE.stInfo[0].dwSymbolMask = (MF_BARCODE_SYMBOL_CODABAR | MF_BARCODE_SYMBOL_ITF);
```

BYTE bDirection

Sets the barcode decode direction. The valid commands are listed below.

bDirection (Constant)	Description
MF_BARCODE_DIRECTION_ALL	Decodes it in both directions from left to right and top to bottom
MF_BARCODE_DIRECTION_LEFTRIGHT	Decodes it from left to right
MF_BARCODE_DIRECTION_TOPBOTTOM	Decodes it from top down
MF_BARCODE_DIRECTION_RIGHTLEFT	Decodes it from right to left
MF_BARCODE_DIRECTION_BOTTOMTOP	Decodes it from bottom up

BYTE bOrigin

Specifies the origin to specify the decode area. The valid commands are listed below.

bOrigin (Constant)	Description
MF_BARCODE_ORIGIN_TOP_LEFT	Sets the origin to the top left corner in relation to the document insertion direction.
MF_BARCODE_ORIGIN_BOTTOM_LEFT	Sets the origin to the bottom left corner in relation to the document insertion point.
MF_BARCODE_ORIGIN_TOP_RIGHT	Sets the origin to the top right corner in relation to the document insertion point.
MF_BARCODE_ORIGIN_BOTTOM_RIGHT	Sets the origin to the bottom right corner in relation to the document insertion point.

WORD wStartX

Specifies the decode region (starting X coordinate). Specified in increments of 0.1 mm.



If MF_BARCODE.dwInfoMode is anything other than 0, and a value larger than wEndX is set, a parameter error (ERR_PARAM) occurs.

WORD wStartY

Specifies the decode region (starting Y coordinate). Specified in increments of 0.1 mm.



If MF_BARCODE.dwInfoMode is anything other than 0, and a value larger than wEndY is set, a parameter error (ERR_PARAM) occurs.

WORD wEndX

Specifies the decode region (ending X coordinate). Specifies a value larger than wStartX (unit: 0.1 mm).

WORD wEndY

Specifies the decode region (ending Y coordinate). Specifies a value larger than wStartY (unit: 0.1 mm).

BARCODE_DATA structure

```
typedef struct {
    DWORD dwSymbol;           OUT
    BYTE bDirection;         OUT
    WORD wStartX;             OUT
    WORD wStartY;             OUT
    WORD wEndX;               OUT
    WORD wEndY;               OUT
    DWORD dwDataSize;         OUT
    LPVOID pData;             OUT
} BARCODE_DATA, *LPBARCODE_DATA;
```

DWORD dwSymbol

The scanned barcode symbol is set.

dwSymbolMask (Constant)	Barcode Type
MF_BARCODE_SYMBOL_CODABAR	Codabar
MF_BARCODE_SYMBOL_CODE128	Code128
MF_BARCODE_SYMBOL_CODE39	Code39
MF_BARCODE_SYMBOL_ITF	ITF
MF_BARCODE_SYMBOL_EAN_JAN	JAN13(EAN), JAN8(EAN)
MF_BARCODE_SYMBOL_UPC_A	UPC-A
MF_BARCODE_SYMBOL_UPC_E	UPC-E

BYTE bDirection

The direction of decoded barcode is set.

bDirection (Constant)	Description
MF_BARCODE_DIRECTION_LEFTRIGHT	Decodes it from left to right
MF_BARCODE_DIRECTION_TOPBOTTOM	Decodes it from top down
MF_BARCODE_DIRECTION_RIGHTLEFT	Decodes it from right to left
MF_BARCODE_DIRECTION_BOTTOMTOP	Decodes it from bottom up

WORD wStartX

The starting of decoding in the x-coordinate (dot unit) of scanned barcode is set.

WORD wStartY

The starting of decoding in the y-coordinate (dot unit) of scanned barcode is set.

WORD wEndX

The end of decoding in the x-coordinate (dot unit) of scanned barcode is set.

WORD wEndY

The end of decoding in the y-coordinate (dot unit) of scanned barcode is set.

DWORD dwDataSize

The size of the scanned barcode's binary data is set.

The size including the NULL at the end is set if the set data(pData) is a character string.

LPVOID pData

The scanned barcode's binary data is set.

The size including the NULL at the end is set if the set result is a character string.

Free the reference memory with GlobalFree.

Default Values of the MF_BARCODE Structure

Member of MF_BARCODE Structure	Default Value
MF_BARCODE.bErrorSelect	MF_ERROR_SELECT_NODETECT
MF_BARCODE.bErrorEject	MF_EJECT_MAIN_POCKET
MF_BARCODE.bStamp	MF_STAMP_DISABLE
MF_BARCODE.bCancel	MF_CANCEL_DISABLE
MF_BARCODE.dwTargetColor	MF_BARCODE_TARGET_COLOR_GRAY
MF_BARCODE.sResolution	MF_SCAN_DPI_DEFAULT
MF_BARCODE.dwInfoMode	0
MF_BARCODE.stInfo.dwSymbolMask	0
MF_BARCODE.stInfo.bDirection	MF_BARCODE_DIRECTION_ALL
MF_BARCODE.stInfo.bOrigin	MF_BARCODE_ORIGIN_TOP_LEFT
MF_BARCODE.stInfo.wStartX	0
MF_BARCODE.stInfo.wStartY	0
MF_BARCODE.stInfo.wEndX	65535
MF_BARCODE.stInfo.wEndY	65535

MF_DECORATE

```
typedef struct {
    DWORD dwAttribute;           IN
    WORD wFont;                 IN
    LPSTR szFontName;           IN
    WORD wFontSize;             IN
} MF_DECORATE, *LPMF_DECORATE;
```

DWORD dwAttribute

Specifies the text attributes. The following values can be set individually or in multiples.

dwAttribute (Constant)	Description
MF_PRINT_BOLD	Executes emphasized printing
MF_PRINT_UNDERLINE_1	Adds a 1-line width of UnderLine
MF_PRINT_UNDERLINE_2	Adds a two-line width of UnderLine
MF_PRINT_REVERSEVIDEO	Executes reverse printing
MF_PRINT_BLACK	Prints a character with the first color (usually it is black)
MF_PRINT_COLOR	<ul style="list-style-type: none"> Prints a character with the second color. Even if this value is set, text is printed using the 1st color.
MF_PRINT_MIXED	<ul style="list-style-type: none"> Prints a character with the first and second colors Even if this value is set, text is printed using the 1st color.
MF_PRINT_1ST_COLOR	Prints a character with the first color (usually it is black)
MF_PRINT_2ND_COLOR	Prints a character with the second color
MF_PRINT_NO_ATTRIBUTE	Not specifying the attribute



Even if MF_PRINT_COLOR, MF_PRINT_MIXED is specified to dwAttribute of the MF_DECORATE structure, the character string printed is MF_PRINT_BLACK.

WORD wFont

Specifies the type of font. The selectable value is as follows.

wFont (Constant)	Description
MF_PRINT_SYSTEMFONT	Prints with the system font
MF_PRINT_FONT_A	Prints with the FontA
MF_PRINT_FONT_B	Prints with the FontB

LPSTR szFontName

When MF_PRINT_SYSTEMFONT is specified with wFont, any system font can be used for printing by specifying the font name in this parameter. If the specified font is not present, printing is performed using the system default font.

WORD wFontSize

Specifies the font size.



- When MF_PRINT_SYSTEMFONT is specified in wFont, point units can be specified. Many system fonts support 8 to 72 pt.
- If the specified size does not agree, printing is performed using the nearest size font.
- Small sized characters are hard to read at 100/120 dpi. Arial font, size 15 point or larger is recommended.

MF_MICR

This structure is used for compatibility. For more information, refer to ["MF_MICR01" on page 286](#).

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    int iRet;                                 OUT
    BYTE bFont;                               IN
    BYTE bMicOcrSelect;                       IN
    BOOL bIParsing;                           IN
    BYTE bStatus;                             OUT
    BYTE bDetail;                             OUT
    CHAR szMicStr[MF_MICR_CHAR_MAX];          OUT
    MF_OCR_RELIABLE_INFO stOcrReliableInfo[MF_MICR_CHAR_MAX]; OUT
    CHAR szAccountNumber[MF_MICR_CHAR_MAX];   OUT
    CHAR szAmount[MF_MICR_CHAR_MAX];          OUT
    CHAR szBankNumber[MF_MICR_CHAR_MAX];      OUT
    CHAR szSerialNumber[MF_MICR_CHAR_MAX];    OUT
    CHAR szEPC[MF_MICR_CHAR_MAX];             OUT
    CHAR szTransitNumber[MF_MICR_CHAR_MAX];   OUT
    long lCheckType;                          OUT
    long lCountryCode;                        OUT
} MF_MICR, *LPMF_MICR;
```

MF_PROCESS

This structure is used for compatibility. For more information, refer to ["MF_PROCESS01" on page 298](#).

```
typedef struct {
    int iSize;                                IN
    int iVersion;                             IN
    BYTE bActivationMode;                     IN
    BYTE bPaperType;                          IN
    DWORD dwStartWaitTime;                    IN
    BYTE bSuccessStamp;                       IN
    BYTE bPaperMisInsertionErrorSelect;       IN
    BYTE bPaperMisInsertionErrorEject;        IN
    BYTE bPaperMisInsertionStamp;             IN
    BYTE bPaperMisInsertionCancel;            IN
    BYTE bNoiseErrorSelect;                   IN
    BYTE bNoiseErrorEject;                    IN
    BYTE bNoiseStamp;                         IN
    BYTE bNoiseCancel;                        IN
    BYTE bDoubleFeedErrorSelect;              IN
    BYTE bDoubleFeedErrorEject;               IN
    BYTE bDoubleFeedStamp;                    IN
    BYTE bDoubleFeedCancel;                   IN
    BYTE bBaddataErrorSelect;                 IN
    BYTE bBaddataCount;                       IN
    BYTE bBaddataEject;                       IN
    BYTE bBaddataStamp;                       IN
    BYTE bBaddataCancel;                      IN
    BYTE bNodataErrorSelect;                  IN
    BYTE bNodataErrorEject;                   IN
    BYTE bNodataStamp;                        IN
    BYTE bNodataCancel;                       IN
    BYTE bNearFullSelect;                     IN
    BYTE bResultPartialData;                  IN
} MF_PROCESS, *LPMF_PROCESS;
```

Device Information

Device Status

* Not Supported in the TM-S2000II and the TM-S2000.

Macro Definition (Constant)	ON / OFF	Value	Status
ASB_NO_RESPONSE	ON	0x00000001	Device not responding
	OFF	0x00000000	Device responding
ASB_PRINT_SUCCESS	ON	0x00000002	Print complete
	OFF	0x00000000	-
ASB_DRAWER_KICK *	ON	0x00000004	Status of drawer kick connector 3rd pin became high
	OFF	0x00000000	-
ASB_OFF_LINE	ON	0x00000008	Offline status
	OFF	0x00000000	Online status
ASB_MAIN_POCKET_NEAR_FULL	ON	0x00000010	The main pocket is nearly full
	OFF	0x00000000	The main pocket is not nearly full
ASB_COVER_OPEN	ON	0x00000020	Cover is open
	OFF	0x00000000	Cover is closed
ASB_PAPER_FEED *	ON	0x00000040	Paper is being fed with the Paper feed switch or the button 2
	OFF	0x00000000	The printer Feed Switch or Button 2 is not pressed.
ASB_SUB_POCKET_NEAR_FULL	ON	0x00000080	The sub pocket is nearly full
	OFF	0x00000000	The sub pocket is not nearly full
ASB_WAIT_PEPRT_EJECT	ON	0x00000100	Waiting for cut sheet ejection
	OFF	0x00000000	-
ASB_PANEL_SWITCH *	ON	0x00000200	Paper feed switch or Button 2 has been pressed (ON)
	OFF	0x00000000	Paper feed switch or Button 2 has not been pressed (OFF)
ASB_MECHANICAL_ERR	ON	0x00000400	Recoverable errors occurred
	OFF	0x00000000	Recoverable errors not occurred
ASB_AUTOCUTTER_ERR *	ON	0x00000800	Auto-cutter error occurred
	OFF	0x00000000	Auto-cutter error not occurred
ASB_UNRECOVER_ERR	ON	0x00002000	Unrecoverable error occurred
	OFF	0x00000000	Unrecoverable error not occurred
ASB_AUTORECOVER_ERR	ON	0x00004000	Auto-recovery error occurred
	OFF	0x00000000	Auto-recovery error not occurred
ASB_NOT_CARD_INSERT	ON	0x00010000	The card is not inserted into the PhotoID detector
	OFF	0x00000000	-
ASB_RECEIPT_NEAR_END *	ON	0x00020000	No paper in the roll paper near end detector
	OFF	0x00000000	Paper in the roll paper near end detector
ASB_EJECT_SENSOR_NO_PAPER	ON	0x00040000	No paper in the eject detector
	OFF	0x00000000	Paper in the eject detector

* Not Supported in the TM-S2000II and the TM-S2000.

Macro Definition (Constant)	ON / OFF	Value	Status
ASB_RECEIPT_END *	ON	0x00080000	No paper in the roll paper end detector
	OFF	0x00000000	Paper in the roll paper end detector
ASB_PAPER_INTERMEDIATE	ON	0x00200000	No paper in the intermediate detector
	OFF	0x00000000	Paper in the intermediate detector
ASB_SLIP_TOF (TM-J9000 Compatibility)	ON	0x00200000	No paper in TOF detector
	OFF	0x00000000	Paper in TOF detector
ASB_ASF_PAPER	ON	0x00400000	No paper in the ASF detector
	OFF	0x00000000	Paper in the ASF detector
ASB_SLIP_SELECTED	ON	0x01000000	Cut sheet is not selected
	OFF	0x00000000	Cut sheet is selected
ASB_PRINT_SLIP	ON	0x02000000	Printing is disabled for cut sheet
	OFF	0x00000000	Printing is enabled for cut sheet
ASB_VALIDATION_SELECTED	ON	0x04000000	Validation is not selected
	OFF	0x00000000	Validation is selected
ASB_PRINT_VALIDATION	ON	0x08000000	Printing is disabled for validation
	OFF	0x00000000	Printing is enabled for validation
ASB_WAIT_INSERT	ON	0x20000000	Waiting for cut sheet insertion
	OFF	0x00000000	-
ASB_SLIP_PAPER_SIZE	ON	0x40000000	No paper in the paper length detector
	OFF	0x00000000	Paper in the paper length detector
ASB_VALIDATION_NO_PAPER (TM-J9000 Compatibility)	ON	0x40000000	No paper in the validation detector
	OFF	0x00000000	Paper in the validation detector

Device Status (for TM-S1000)

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 status
	OFF	0x00000000	Online status
ASB_COVER_OPEN	ON	0x00000020	Cover is open
	OFF	0x00000000	Cover is closed
ASB_WAIT_PEPRT_EJECT	ON	0x00000100	Waiting for cut sheet ejection
	OFF	0x00000000	-
ASB_MECHANICAL_ERR	ON	0x00000400	Recoverable errors occurred
	OFF	0x00000000	Recoverable errors not occurred
ASB_UNRECOVER_ERR	ON	0x00002000	Unrecoverable error occurred
	OFF	0x00000000	Unrecoverable error not occurred
ASB_PAPER_INTERMEDIATE	ON	0x00010000	No paper in the intermediate detector
	OFF	0x00000000	Paper in the intermediate detector
ASB_MAIN_NEAR_FULL	ON	0x00020000	The main pocket is not nearly full
	OFF	0x00000000	The main pocket is nearly full
ASB_EJECT_SENSOR_NO_PAPER	ON	0x00040000	No paper in the eject detector
	OFF	0x00000000	Paper in the eject detector
ASB_SUB_NEAR_FULL	ON	0x00080000	The sub pocket is not nearly full
	OFF	0x00000000	The sub pocket is nearly full
ASB_SLIP_PAPER_SIZE	ON	0x00200000	No paper in the paper length detector
	OFF	0x00000000	Paper in the paper length detector
ASB_ASF_PAPER	ON	0x00400000	No paper in the ASF detector
	OFF	0x00000000	Paper in the ASF detector
ASB_STAMP_EXIST	ON	0x02000000	The franking cartridge is not installed
	OFF	0x00000000	The franking cartridge is installed
ASB_WAIT_INSERT	ON	0x20000000	Waiting for cut sheet insertion
	OFF	0x00000000	-
ASB_FRANKING_SENSOR	ON	0x40000000	No paper in the franking detector
	OFF	0x00000000	Paper in the franking detector

Ink Status

Macro Definition (Constant)	ON / OFF	Value	Status
INK_ASB_OK	ON	0x0000	
INK_ASB_NEAR_END	ON	0x0001	Remaining ink level is low
	OFF	0x0000	-
INK_ASB_END	ON	0x0002	Exchange the ink cartridge
	OFF	0x0000	-
INK_ASB_NO_CARTRIDGE	ON	0x0004	No ink cartridge
	OFF	0x0000	-
INK_ASB_NO_CARTRIDGE2 *	ON	0x0008	No ink cartridge (2nd colors)
	OFF	0x0000	Ink cartridge present
INK_ASB_CLEANING	ON	0x0020	Cleaning in progress
	OFF	0x0000	Cleaning is not in progress
INK_ASB_NEAR_END2 *	ON	0x0100	Remaining ink level is low (2nd colors)
	OFF	0x0000	-
INK_ASB_END2 *	ON	0x0200	Exchange the ink cartridge (2nd colors)
	OFF	0x0000	-

*: Defined for compatibility with TM-J9000. Normally OFF for this product.

Maintenance Counter

* Not Supported in the TM-S2000II and the TM-S2000.

Reset ability	Counter Number	Counter	Unit
Resettable	21 (15H) *	Number of times head timing pulse (for roll paper)	Times
	22 (16H) *	Number of lines fed by thermal head	Lines
	30 (1EH) *	Number of lines fed for roll paper	Lines
	31 (1FH)	Number of IJ head shots (Column A)	1000 shots
	32 (20H)	Number of IJ head shots (Column B)	1000 shots
	34 (22H)	Count of pump motor operations	Count
	50 (32H) *	Count of autocutter drive	Count
	60 (3CH)	Count of magnetic ink character read	Count
	61 (3DH)	Count of check paper scanning	Count
	62 (3EH)	Count of card scanning	Count
	67 (43H)	Count of check paper feeding	Count
	70 (46H)	Duration of product operation	Hours
	80 (50H)	Count of hopper open/close	Count
	82 (52H)	Count of pocket switch	Count
Cumulative	149 (95H) *	Number of times head timing pulse (for roll paper)	Times
	150 (96H) *	Number of lines fed by thermal head	Lines
	158 (9EH) *	Number of lines fed for roll paper	Lines
	159 (9FH)	Number of IJ head shots (Column A)	1000 shots
	160 (A0H)	Number of IJ head shots (Column B)	1000 shots
	162 (A2H)	Count of pump motor operations	Count
	178 (B2H) *	Count of autocutter drive	Count
	188 (BCH)	Count of magnetic ink character read	Count
	189 (BDH)	Count of check paper scanning	Count
	190 (BEH)	Count of card scanning	Count
	195 (C3H)	Count of check paper feeding	Count
	198 (C6H)	Duration of product operation	Hours
	208 (D0H)	Count of hopper open/close	Count
	210 (D2H)	Count of pocket switch	Count

MICR Status

Bit	ON/OFF	Value	Status
0	-	0x00	Fixed to 0
1	-	0x02	Fixed to 1
2	ON	0x04	MICR function non-selection
	OFF	0x00	MICR function selection
3	ON	0x08	Waiting for check sheet/cleaning sheet insertion
	OFF	0x00	-
4	-	0x10	Fixed to 1
5	ON	0x20	Middle sensor senses no paper
	OFF	0x00	Middle sensor senses that there is paper
6	ON	0x40	ASF sensor senses no paper
	OFF	0x00	ASF sensor senses that there is paper
7	-	0x00	Fixed to 0

Offline Code

BiGetOfflineCodeByIndex (for the TM-S9000II, the TM-S2000II, the TM-S9000 and the TM-S2000)**First Byte (Unrecoverable error)**

Bit	ON/ OFF	Value	Status
0	-	0x00	Fixed to 0
1	-	0x00	Fixed to 0
2	-	0x00	Fixed to 0
3	-	0x00	Fixed to 0
4	OFF	0x00	Power retrigger error not occurred (Fixed to 0)
5	-	0x00	Fixed to 0
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Second Byte (Unrecoverable error)

Bit	ON/ OFF	Value	Status
0	ON	0x01	High voltage error occurred
	OFF	0x00	High voltage error not occurred
1	ON	0x02	Low voltage error occurred
	OFF	0x00	Low voltage error not occurred
2	-	-	Undefined
3	ON	0x08	FPGA CONFIG error occurred
	OFF	0x00	FPGA CONFIG error not occurred
4	ON	0x10	Ink-jet mechanical error occurred
	OFF	0x00	Ink-jet mechanical error not occurred
5	-	-	Undefined
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Third Byte (Unrecoverable error)

* Undefined in the TM-S2000II and the TM-S2000.

Bit	ON/ OFF	Value	Status
0 *	ON	0x01	Thermal head thermistor error occurred
	OFF	0x00	Thermal head thermistor error not occurred
1	ON	0x02	Ink-jet head thermistor error occurred
	OFF	0x00	Ink-jet head thermistor error not occurred
2	ON	0x04	Waste ink absorption volume error or ink information writing error occurred
	OFF	0x00	Waste ink absorption volume error or ink information writing error not occurred
3	ON	0x08	Motor driver thermistor error occurred
	OFF	0x00	Motor driver thermistor error not occurred
4	ON	0x10	Ink-jet head driver thermistor error occurred
	OFF	0x00	Ink-jet head driver thermistor error not occurred
5	ON	0x20	Pump drive frequency error occurred
	OFF	0x00	Pump drive frequency error not occurred
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Fourth Byte (Unrecoverable error)

Bit	ON/ OFF	Value	Status
0	-	-	Undefined
1	-	-	Undefined
2	-	-	Undefined
3	-	-	Undefined
4	-	-	Undefined
5	-	-	Undefined
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Fifth Byte (Recoverable Error: Mechanism position error)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Hopper position error occurred
	OFF	0x00	Hopper position error not occurred
1	ON	0x02	Switching plate position error occurred
	OFF	0x00	Switching plate position error not occurred
2	-	-	Undefined
3	-	-	Undefined
4	-	-	Undefined
5	ON	0x20	Endorsement printing data not received error occurred
	OFF	0x00	Endorsement printing data not received error not occurred
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Sixth Byte (Recoverable Error: Roll paper)

* Undefined in the TM-S2000II and the TM-S2000.

Bit	ON/ OFF	Value	Status
0 *	ON	0x01	Auto-cutter error occurred
	OFF	0x00	Auto-cutter error not occurred
1 *	ON	0x02	Roll paper cover open error occurred (when specifying recovery error)
	OFF	0x00	Roll paper cover open error not occurred (when specifying recovery error)
2	-	-	Undefined
3	-	-	Undefined
4	-	-	Undefined
5	-	-	Undefined
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Seventh Byte (Recoverable Error: Cut sheet paper)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Cut sheet paper stuck error occurred
	OFF	0x00	Cut sheet paper stuck error not occurred
1	ON	0x02	Cut sheet paper feed error occurred
	OFF	0x00	Cut sheet paper feed error not occurred
2	ON	0x04	Cut sheet paper length error occurred
	OFF	0x00	Cut sheet paper length error not occurred
3	ON	0x08	Cut sheet paper transfer error occurred
	OFF	0x00	Cut sheet paper transfer error not occurred
4	ON	0x10	Paper path cover open error occurred, when cut sheet paper is printed
	OFF	0x00	Paper path cover open error not occurred, when cut sheet paper is printed
5	-	-	Undefined
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Eighth Byte (Recoverable Error: Other)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Cut sheet paper double feed error occurred
	OFF	0x00	Cut sheet paper double feed error not occurred
1	ON	0x02	Cut sheet paper insertion direction error occurred
	OFF	0x00	Cut sheet paper insertion direction error not occurred
2	ON	0x04	External noise error occurred
	OFF	0x00	External noise error not occurred
3	ON	0x08	Error in the read processing that was designated by a command
	OFF	0x00	No error in the read processing that was designated by a command
4	ON	0x10	Print specification error caused by specifying the printing content exceeds printable area
	OFF	0x00	Print specification error not caused by specifying the printing content exceeds printable area.
5	ON	0x20	No magnetically read characters error occurred
	OFF	0x00	No magnetically read characters error not occurred
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Ninth Byte (Auto-Recovery Error)

* Undefined in the TM-S2000II and the TM-S2000.

Bit	ON/ OFF	Value	Status
0 *	ON	0x01	Roll paper cover open error occurred (when specifying auto-recovery error)
	OFF	0x00	Roll paper cover open error not occurred (when specifying auto-recovery error)
1	ON	0x02	Thermal head/ink-jet head driver temporary high-temperature error occurred
	OFF	0x00	Thermal head/ink-jet head driver temporary high-temperature error not occurred
2 *	ON	0x04	Motor driver IC temporary high-temperature error occurred
	OFF	0x00	Motor driver IC temporary high-temperature error not occurred
3	ON	0x08	Ink carriage cover is open
	OFF	0x00	Ink carriage cover is closed
4	ON	0x10	Paper path cover is open
	OFF	0x00	Paper path cover is closed
5 *	ON	0x20	Roll paper cover is open
	OFF	0x00	Roll paper cover is closed
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Tenth Byte (Auto-Recovery Error)

* Undefined in the TM-S2000II and the TM-S2000.

Bit	ON/ OFF	Value	Status
0 *	ON	0x01	Thermal head temporary low-voltage error occurred
	OFF	0x00	Thermal head temporary low-voltage error not occurred
1	-	0x00	Fixed to 0
2	-	0x00	Fixed to 0
3	-	0x00	Fixed to 0
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 0
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

BiGetOfflineCodeByIndex (for TM-S1000)

First Byte

Bit	ON/ OFF	Value	Status
0	-	0x00	Fixed to 0
1	ON	0x02	CPU execution error occurred
	OFF	0x00	CPU execution error not occurred
2	ON	0x04	ROM error occurred
	OFF	0x00	ROM error not occurred
3	-	0x00	Fixed to 0
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 0
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Second Byte

Bit	ON/ OFF	Value	Status
0	ON	0x01	High voltage error occurred
	OFF	0x00	High voltage error not occurred
1	ON	0x02	Low voltage error occurred
	OFF	0x00	Low voltage error not occurred
2	ON	0x04	CIS error occurred
	OFF	0x00	CIS error not occurred
3	-	0x00	Fixed to 0
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 0
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Third Byte

Bit	ON/ OFF	Value	Status
0	ON	0x01	Hopper position error occurred
	OFF	0x00	Hopper position error not occurred
1	ON	0x02	Stamp position error occurred
	OFF	0x00	Stamp position error not occurred
2	ON	0x04	Switching plate position error occurred
	OFF	0x00	Switching plate position error not occurred
3	-	0x00	Fixed to 0
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 0
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Fourth Byte

Bit	ON/ OFF	Value	Status
0	ON	0x01	Cut sheet paper stuck error occurred
	OFF	0x00	Cut sheet paper stuck error not occurred
1	ON	0x02	Cut sheet paper feed error occurred
	OFF	0x00	Cut sheet paper feed error not occurred
2	ON	0x04	Cut sheet paper length error occurred
	OFF	0x00	Cut sheet paper length error not occurred
3	ON	0x08	Cut sheet paper transfer error occurred
	OFF	0x00	Cut sheet paper transfer error not occurred
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 0
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Fifth Byte

Bit	ON/ OFF	Value	Status
0	ON	0x01	Cut sheet paper double feed error occurred
	OFF	0x00	Cut sheet paper double feed error not occurred
1	ON	0x02	Cut sheet paper insertion direction error occurred
	OFF	0x00	Cut sheet paper insertion direction error not occurred
2	ON	0x04	External noise error occurred
	OFF	0x00	External noise error not occurred
3	ON	0x08	Error in the read processing that was designated by a command
	OFF	0x00	No error in the read processing that was designated by a command
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 1
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

BiGetOfflineCode

First Byte (Unrecoverable error)

Bit	ON/ OFF	Value	Status
0	-	0x00	Fixed to 0
1	-	0x00	Fixed to 0
2	-	0x00	Fixed to 0
3	-	0x00	Fixed to 0
4	OFF	0x00	Power retrigger error not occurred (Fixed to 0)
5	-	0x00	Fixed to 0
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Second Byte (Unrecoverable error)

Bit	ON/ OFF	Value	Status
0	ON	0x01	High voltage error occurred
	OFF	0x00	High voltage error not occurred
1	ON	0x02	Low voltage error occurred
	OFF	0x00	Low voltage error not occurred
2	-	-	Undefined
3	ON	0x08	FPGA CONFIG error occurred
	OFF	0x00	FPGA CONFIG error not occurred
4	ON	0x10	Ink-jet mechanical error occurred
	OFF	0x00	Ink-jet mechanical error not occurred
5	-	-	Undefined
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Third Byte (Unrecoverable error)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Thermal head thermistor error occurred
	OFF	0x00	Thermal head thermistor error not occurred
1	ON	0x02	Ink-jet head thermistor error occurred
	OFF	0x00	Ink-jet head thermistor error not occurred
2	ON	0x04	Waste ink absorption volume error or ink information writing error occurred
	OFF	0x00	Waste ink absorption volume error or ink information writing error not occurred
3	ON	0x08	Motor driver thermistor error occurred
	OFF	0x00	Motor driver thermistor error not occurred
4	ON	0x10	Ink-jet head driver thermistor error occurred
	OFF	0x00	Ink-jet head driver thermistor error not occurred
5	ON	0x20	Pump drive frequency error occurred
	OFF	0x00	Pump drive frequency error not occurred
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Fourth Byte (Unrecoverable error)

Bit	ON/ OFF	Value	Status
0	-	-	Undefined
1	-	-	Undefined
2	-	-	Undefined
3	-	-	Undefined
4	-	-	Undefined
5	-	-	Undefined
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Fifth Byte (Recoverable Error: Mechanism position error)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Hopper position error occurred
	OFF	0x00	Hopper position error not occurred
1	ON	0x02	Switching plate position error occurred
	OFF	0x00	Switching plate position error not occurred
2	-	-	Undefined
3	-	-	Undefined
4	-	-	Undefined
5	ON	0x20	Endorsement printing data not received error occurred
	OFF	0x00	Endorsement printing data not received error not occurred
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Sixth Byte (Recoverable Error: Roll paper)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Auto-cutter error occurred
	OFF	0x00	Auto-cutter error not occurred
1	ON	0x02	Roll paper cover open error occurred (when specifying recovery error)
	OFF	0x00	Roll paper cover open error not occurred (when specifying recovery error)
2	-	-	Undefined
3	-	-	Undefined
4	-	-	Undefined
5	-	-	Undefined
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Seventh Byte (Recoverable Error: Cut sheet paper)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Cut sheet paper stuck error occurred
	OFF	0x00	Cut sheet paper stuck error not occurred
1	ON	0x02	Cut sheet paper feed error occurred
	OFF	0x00	Cut sheet paper feed error not occurred
2	ON	0x04	Cut sheet paper length error occurred
	OFF	0x00	Cut sheet paper length error not occurred
3	ON	0x08	Cut sheet paper transfer error occurred
	OFF	0x00	Cut sheet paper transfer error not occurred
4	ON	0x10	Paper path cover open error occurred, when cut sheet paper is printed
	OFF	0x00	Paper path cover open error not occurred, when cut sheet paper is printed
5	-	-	Undefined
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Eighth Byte (Recoverable Error: Other)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Cut sheet paper double feed error occurred
	OFF	0x00	Cut sheet paper double feed error not occurred
1	ON	0x02	Cut sheet paper insertion direction error occurred
	OFF	0x00	Cut sheet paper insertion direction error not occurred
2	ON	0x04	External noise error occurred
	OFF	0x00	External noise error not occurred
3	ON	0x08	Error in the read processing that was designated by a command
	OFF	0x00	No error in the read processing that was designated by a command

Bit	ON/ OFF	Value	Status
4	ON	0x10	Print specification error caused by specifying the printing content exceeds printable area
	OFF	0x00	Print specification error not caused by specifying the printing content exceeds printable area.
5	ON	0x20	No magnetically read characters error occurred
	OFF	0x00	No magnetically read characters error not occurred
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Ninth Byte (Auto-Recovery Error)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Roll paper cover open error occurred (when specifying auto-recovery error)
	OFF	0x00	Roll paper cover open error not occurred (when specifying auto-recovery error)
1	ON	0x02	Thermal head/ink-jet head driver temporary high-temperature error occurred
	OFF	0x00	Thermal head/ink-jet head driver temporary high-temperature error not occurred
2	ON	0x04	Motor driver IC temporary high-temperature error occurred
	OFF	0x00	Motor driver IC temporary high-temperature error not occurred
3	ON	0x08	Ink carriage cover is open
	OFF	0x00	Ink carriage cover is closed
4	ON	0x10	Paper path cover is open
	OFF	0x00	Paper path cover is closed
5	ON	0x20	Roll paper cover is open
	OFF	0x00	Roll paper cover is closed
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Tenth Byte (Auto-Recovery Error)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Thermal head temporary low-voltage error occurred
	OFF	0x00	Thermal head temporary low-voltage error not occurred
1	-	0x00	Fixed to 0
2	-	0x00	Fixed to 0
3	-	0x00	Fixed to 0
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 0
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Device ID

Device ID	Type of Device ID	Status
1	Product ID	<ul style="list-style-type: none"> 131(83H) for the TM-S9000II and the TM-S9000. 132(84H) for the TM-S2000II and the TM-S2000.
2	Type ID (A)	1-byte data. See "Type ID (A)" on page 381 .
33	Type information	2-byte data. See "Type Information" on page 382 .
65	Version of firmware	String data. The obtained data varies depending on the firmware version. Example: "1.00 ESC/POS"
66	Manufacture name	Fixed to "EPSON".
67	Product name	Fixed to "TM-S9000MJ" for the TM-S9000II and the TM-S9000. Fixed to "TM-S2000MJ" for the TM-S2000II and the TM-S2000.
68	Serial number	String data. The obtained data varies depending on the device. Example: "DEKK000015"
112	Type ID (B)	1-byte data. See "Type ID (B)" on page 383 .

Type ID (A)

Bit	ON/OFF	Value	Status
0	ON	0x01	Multi-byte characters are supported.
	OFF	0x00	Multi-byte characters are not supported.
1	ON	0x02	Auto-cutter is installed. Fixed to 1 for the TM-S9000II and the TM-S9000. Fixed to 0 for the TM-S2000II and the TM-S2000.
2	OFF	0x00	DM-D is not connected. (Fixed to 0)
3	ON	0x08	MICR reader is installed. (Fixed to 1)
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 0
6	ON	0x40	Endorsement printer(E/P) is installed. (Fixed to 1)
7	-	0x00	Fixed to 0

Type Information

First Byte

Bit	ON/ OFF	Value	Status
0	ON	0x01	Multi-byte characters are supported.
	OFF	0x00	Multi-byte characters are not supported.
1	ON	0x02	Auto-cutter is installed. Fixed to 1 for the TM-S9000II and the TM-S9000. Fixed to 0 for the TM-S2000II and the TM-S2000.
2	OFF	0x00	DM-D is not connected. (Fixed to 0)
3	ON	0x08	MICR reader is installed. (Fixed to 1)
4	ON	0x10	Image scanner is installed. (Fixed to 1)
5	ON	0x20	Endorsement printer(E/P) is installed. (Fixed to 1)
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Second Byte

Bit	ON/ OFF	Value	Status
0	OFF	0x00	No NV memory for saving image scanning results.
1	ON	0x02	Grayscale reading is supported. (Fixed to 1)
2	ON	0x04	Image scanner unit is installed. (Fixed to 1)
3	ON	0x08	Automatic sheet feeder is installed. (Fixed to 1)
4	-	0x00	Fixed to 0
5	ON	0x20	Validation mechanism is installed. (Fixed to 1)
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Type ID (B)

Bit	ON/ OFF	Value	Status
0	ON	0x01	Magnetic stripe reader is installed.
	OFF	0x00	Magnetic stripe reader is not installed.
1	ON	0x02	2 pockets
	OFF	0x00	1 pocket
2	ON	0x04	UV light source is not equipped.
	OFF	0x00	UV light source is equipped.
3	Maximum speed setting status (See "Maximum speed setting status" on page 383)		
4			
5	ON	0x20	1 pass Photo ID scan with visible and 2nd light source can be enabled.
	OFF	0x00	1 pass Photo ID scan with visible and 2nd light source is disabled.
6	-	0x40	Fixed to 1
7	-	0x00	Fixed to 0

Maximum speed setting status

Printer Model	3 Bit	4 Bit
TM-S9000II/ TM-S2000II: 130 DPM	ON (0x01)	OFF (0x00)
TM-S9000/ TM-S2000: 110 DPM		
TM-S9000II/ TM-S2000II: 225 DPM	OFF (0x00)	ON (0x01)
TM-S9000/ TM-S2000: 200 DPM		

Type ID

Bit	ON/ OFF	Value	Status
0	ON	0x01	Multi-byte characters are supported.
	OFF	0x00	Multi-byte characters are not supported.
1	ON	0x02	Auto-cutter is installed. Fixed to 1 for the TM-S9000II and the TM-S9000. Fixed to 0 for the TM-S2000II and the TM-S2000.
2	OFF	0x00	DM-D is not connected. (Fixed to 0)
3	ON	0x08	MICR reader is installed. (Fixed to 1)
4	-	0x00	Fixed to 0
5	-	0x00	Fixed to 0
6	ON	0x40	Endorsement printer(E/P) is installed. (Fixed to 1)
7	-	0x00	Fixed to 0

Compatible Information

This chapter provides a description of the compatibility information on TM-J9000/TM-S1000 API.

API

OK: API that is compatible with the TM-J9000/TM-S1000 API. Use as it is.

Change: API that has the same name, but that has a different input or output.

New: Added new API.

* Not supported in the TM-S2000II and the TM-S2000.

API	Compatibility	Detail
BiOpenMonPrinter	Change	The value set in the parameters changes. <TM-S9000II and TM-S9000> TYPE_PORT: USB3 TYPE_PRINTER: TMS9000U <TM-S2000II and TM-S2000 > TYPE_PORT: USB4 TYPE_PRINTER: TMS2000U
BiCloseMonPrinter	OK	-
BiSCNMICRSetStatusBackFunction	OK	-
BiSCNMICRSetStatusBackWnd	OK	-
BiSCNMICRSetStatusBackWndEx	New	-
BiSCNMICRCancelStatusBack	OK	-
BiStartEndorsementSetStatusBackFunction	New	-
BiStartEndorsementSetStatusBackWnd	New	-
BiStartEndorsementCancelStatusBack	New	-
BiEndEndorsementSetStatusBackFunction	New	-
BiEndEndorsementSetStatusBackWnd	New	-
BiEndEndorsementCancelStatusBack	New	-
BiGetStatus	Change	The device status that can be acquired changes.
BiGetRealStatus	Change	The device status that can be acquired changes.
BiSetStatusBackFunction	Change	The device status that can be acquired changes.
BiSetStatusBackFunctionEx	Change	The device status that can be acquired changes.
BiSetStatusBackWnd	Change	The device status that can be acquired changes.
BiSetStatusBackWndEx	New	-
BiCancelStatusBack	OK	-
BiGetInkStatus	Change	The ink status that can be acquired changes.
BiSetInkStatusBackFunction	Change	The ink status that can be acquired changes.
BiSetInkStatusBackFunctionEx	Change	The ink status that can be acquired changes.
BiSetInkStatusBackWnd	Change	The ink status that can be acquired changes.
BiSetInkStatusBackWndEx	New	-
BiCancelInkStatusBack	OK	-
BiMICRSelectDataHandling	OK	-

OK: API that is compatible with the TM-J9000/TM-S1000 API. Use as it is.

Change: API that has the same name, but that has a different input or output.

New: Added new API.

* Not supported in the TM-S2000II and the TM-S2000.

API	Compatibility	Detail
BiSCNSelectScanUnit	OK	-
BiSCNSetImageTypeOption	New	-
BiSCNGetImageTypeOption	New	-
BiSCNMICRFunctionContinuously	Change	<ul style="list-style-type: none"> The version of the MF_MICR/MF_PROCESS structure is updated. For more information, refer to "Structures" on page 390. It can perform a card scan.
BiSCNMICRFunctionPostPrint	Change	<ul style="list-style-type: none"> The version of the MF_MICR/MF_PROCESS structure is updated. For more information, refer to "Structures" on page 390. It can perform a card scan.
BiSCNMICRFunction	Change	The version of the MF_MICR/MF_PROCESS structure is updated. For more information, refer to "Structures" on page 390 .
BiSCNMICRCancelFunction	OK	-
BiSCNDeleteCroppingArea	OK	-
BiSCNSelectScanFace	OK	-
BiSCNSetImageFormat	OK	-
BiSCNGetImageFormat	OK	-
BiSCNSetScanArea	Change	The range of the area that can be specified changes.
BiSCNGetScanArea	OK	-
BiSCNSetImageQuality	Change	It is possible to specify color (24-bit).
BiSCNGetImageQuality	OK	-
BiSCNSelectScanImage	New	-
BiSCNSetCroppingArea	Change	The range of the area that can be specified changes.
BiSCNGetCroppingArea	OK	-
BiSCNSetImageAdjustment	New	-
BiSCNGetImageAdjustment	New	-
BiMICRClearSpaces	OK	-
BiSetOcrABAreaOrigin	OK	-
BiGetMicrText	OK	-
BiGetOcrABText	OK	-
BiGetScanImage	OK	-
BiGetScanImageSize	OK	-
BiGetBarcodeData	OK	-
BiDecodeBarcode	OK	-
BiDecodeBarcodeMemory	OK	-
BiSetTransactionNumber	OK	-
BiGetTransactionNumber	OK	-

OK: API that is compatible with the TM-J9000/TM-S1000 API. Use as it is.

Change: API that has the same name, but that has a different input or output.

New: Added new API.

* Not supported in the TM-S2000II and the TM-S2000.

API	Compatibility	Detail
BiSetTransactionNumberWithIncremental	OK	-
BiSetPrintStation	Change	Added print stations that can be specified.
BiGetPrintStation	OK	-
BiSetPrintPosition	Change	If there is buffering at BiTemplatePrint, even if this is executed, nothing will be performed.
BiGetPrintPosition	OK	-
BiSetEndorseDirection	OK	-
BiSetPrintSize	Change	<ul style="list-style-type: none"> If there is buffering at BiTemplatePrint, even if this is executed, nothing will be performed. If the print station uses roll paper, the upper limit for width that can be specified is 72.
BiGetPrintSize	OK	-
BiSetPrintControl	Change	Has compatibility and can be used, but the result of execution will not be reflected.
BiGetPrintControl	OK	-
BiSetPrintImageMethod	New	-
BiGetPrintImageMethod	New	-
BiSetPrintAlignment	Change	<ul style="list-style-type: none"> Cannot operate at printing stations that do not use roll paper. The upper limit for width that can be specified is 72.
BiGetPrintAlignment	OK	-
BiPrintText	OK	-
BiSCNPrintText	New	-
BiPrintBarCode *	OK	-
BiPrintImage	OK	-
BiPrintMemoryImage	OK	-
BiSCNPrintMemoryImage	New	-
BiPrintMultipleToneImage *	New	-
BiBufferedPrint	OK	-
BiUpdateEndorseText	OK	-
BiSetPrintCutSheetSettings	New	-
BiGetPrintCutSheetSettings	New	-
BiPrintCutSheet	New	-
BiLoadTemplatePrintArea	New	-
BiSetTemplatePrintArea	New	-
BiTemplatePrint	New	-
BiClearTemplatePrintData	New	-
BiInsertValidation	Change	Has compatibility and can be used normally, but with this product, it is used for cut sheet printing.

OK: API that is compatible with the TM-J9000/TM-S1000 API. Use as it is.

Change: API that has the same name, but that has a different input or output.

New: Added new API.

* Not supported in the TM-S2000II and the TM-S2000.

API	Compatibility	Detail
BiRemoveValidation	Change	Has compatibility and can be used normally, but with this product, it is used for cut sheet printing.
BiAutoCutRollPaper *	New	-
BiSetWaterfallMode	OK	-
BiSetNumberOfDocuments	OK	-
BiSetBehaviorToScnResult	Change	Has compatibility and can be used normally, but because this product does not have franking equipped, the settings for franking will be ignored even if they are enabled.
BiSelectErrorEjectAtContinuously	OK	-
BiMICRGetStatus	Change	The value that can be acquired changes.
BiConfirmBufferedData	OK	-
BiGetIQAResult	OK	-
BiGetVersion	Change	The value that can be acquired changes.
BiESCNEnable	OK	-
BiESCNSetAutoSize	OK	-
BiESCNGetAutoSize	OK	-
BiESCNSetCutSize	OK	-
BiESCNGetCutSize	OK	-
BiESCNSetRotate	OK	-
BiESCNGetRotate	OK	-
BiESCNSetDeSkew	OK	-
BiESCNGetDeSkew	OK	-
BiESCNSetDocumentSize	OK	-
BiESCNGetDocumentSize	OK	-
BiESCNDefineCropArea	OK	-
BiESCNGetMaxCropAreas	OK	-
BiESCNSStoreImage	OK	-
BiESCNClearImage	OK	-
BiESCNRRetrieveImage	OK	-
BiESCNGetRemainingImages	OK	-
BiResetPrinter	OK	-
BiCancelError	OK	-
BiMICRCleaning	OK	-
BiInkHeadCleaning	New	-
BiOpenDrawer *	New	-
BiRingBuzzer	OK	-
BiGetCounter	Change	The value that can be acquired changes.
BiResetCounter	OK	-
BiGetPrnCapability	Change	The value that can be acquired changes.

OK: API that is compatible with the TM-J9000/TM-S1000 API. Use as it is.

Change: API that has the same name, but that has a different input or output.

New: Added new API.

* Not supported in the TM-S2000II and the TM-S2000.

API	Compatibility	Detail
BiGetType	Change	The value that can be acquired changes.
BiGetOfflineCodeByIndex	Change	The value that can be acquired changes.
BiGetOfflineCode	Change	The value that can be acquired changes.
BiLoadAPISettings	New	-
BiSetConfigure	New	-
BiSetMonInterval	OK	-
BiSCNGetClumpStatus	Change	Always returns 01000000 (0x40).
BiSelectJamDetect	Change	Has compatibility and can be used, but nothing is executed.
BiSetPaperThickness	Change	Has compatibility and can be used, but nothing is executed.

Structures

OK: Structures that is compatible with the TM-J9000/TM-S1000 API. Use as it is.

Change: Structures that has the same name, but that has a different input or output.

New: Added new Structures.

Structures	Compatibility	Detail
MF_BASE01	OK	-
MF_SCAN	Change	Added the resolution that can be specified.
MF_MICR	OK	Has compatibility and be used normally, but with the TM-S9000II, the TM-S2000II, the TM-S9000 and the TM-S2000, the MF_MICR01 structure is recommended.
MF_MICR01	New	Added the MICR On-US and the Auxiliaty On-US fields to the MF_MICR structure
MF_PROCESS	Change	<ul style="list-style-type: none"> Has compatibility and be used normally, but with the TM-S9000II, the TM-S2000II, the TM-S9000 and the TM-S2000, the MF_PROCESS01 structure is recommended. Because this product does not have franking equipped, the settings for franking will be ignored even if they are enabled.
MF_PROCESS01	New	Added settings for endorsement printing to the MF_PROCESS structure.
MF_IQA	Change	Has compatibility and can be used normally, but because this product does not have franking equipped, the settings for franking will be ignored even if they are enabled.
MF_IQA_RESULT	OK	-
MF_PRINT01	Change	Has compatibility and can be used normally, but is not recommended for this product. For printing, instead of the structure, using APIs such as BiPrintText is recommended.
MF_DECORATE	OK	-
PRINTAREAINFO	New	-
MF_OCR_AB	Change	The range of the area that performs OCR changes.
MF_OCR_RELIABILITY	OK	-
MF_OCR_RELIABLE_INFO	OK	-
MF_BARCODE	Change	Has compatibility and can be used normally, but because this product does not have franking equipped, the settings for franking will be ignored even if they are enabled.
BARCODE_INFO	Change	<ul style="list-style-type: none"> The distinction limit on the decoding direction was removed. Can specify the decode region.
BARCODE_DATA	OK	-
VERSION_INFO	OK	-

Setting File

This chapter provides a description of a configuration file for the TM-S9000/S2000 driver.

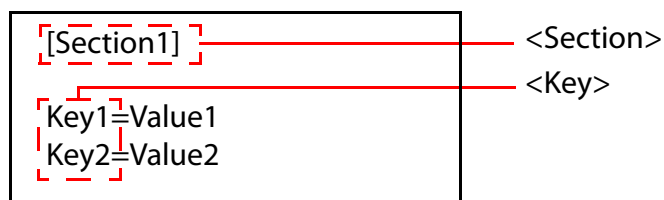


Please refer to ["Log Function" on page 406](#) regarding the setting file for the log function (log setting file).

Method for setting the setting file

Each setting file is in the ini file format.

The file has a "section", which is enclosed in [] and shows the kind of setting, and a "key", which is an element of each setting. Use "=" to set a value to the "key". Refer to the following during setting.



API settings file

The file that specifies the initial settings for the TM-S9000/S2000 API.

API settings file location
C:\Program Files\EPSON\TMSPDriver\bin (If both 32 bit/64 bit are in a 64 bit OS, the 64 bit version is stored in the x64bin folder.)
API settings file name
APISettings.ini



Normally this is set with the APISetting.ini, which is stored in the location shown above, but it can be set and read to any other file name with the ["BiLoadAPISettings" on page 267](#)

Settings for API settings files

Configuration

Section name	
Config	
Key name	Description
2ndReceipt (TM-S9000II and TM-S9000)	Sets the issuing of the second receipt. If it has not been set, it operates the same as when 0 is specified. <Setting value> 0: Disable. 1: Enable.
MICRRecog	Sets the MICR type. If it has not been set, it operates the same as when 0 is specified. <Setting value> 0: ANSI 1: C&CCC

Common Settings

Section name	
Common	
Key name	Description
UseNewErrorCode	Specifies error code compatibility. <Setting value> 0: Compatible error codes with TM-S1000 driver. 1: New error code in the TM-S9000II, the TM-S2000II, the TM-S9000 and the TM-S2000 (ERR_PRINT_DATA_UNRECEIVE or ERR_PRINT_DATA_LENGTH_EXCEED) is returned in corresponding condition. 2: In addition to #1, ERR_INK_STATUS is returned if ink cartridge is not installed or ink is low. (Default setting in Ver.2.10 or later)
OpenOnOffline	Specifies BiOpenMonPrinter return code compatibility. <Setting value> 0: ERR_ACCESS is returned if an off-line error has occurred before "BiOpenMon-Printer" on page 68 is called. 1: ERR_ACCESS is returned only if USB cable is disconnected or the device is powered off. (Default setting in Ver.2.10 or later)

Scan settings

<Common>

Section name	
ScanSettings	

Key name	Description
Waterfall	<p>Sets the initial value for "BiSetWaterfallMode" on page 228.</p> <p>If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0: Disables Waterfall.</p> <p>1: Prioritizes ejecting to the main pocket.</p> <p>2: Ejects equally to the main and sub pockets.</p>
ScanUnit	<p>Sets the initial value for "BiSCNSelectScanUnit" on page 100.</p> <p>If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>48: Makes the check scanner the target scanner.</p> <p>49: Makes the ID card scanner the target scanner.</p>

<Separated by scan units (check paper / card)>

Section name	Description
ScanSettingsCheck	Settings for the check paper unit
ScanSettingsCard	Settings for the card unit

Key name	Description
ScanFace	<p>Sets the initial value for "BiSCNSelectScanFace" on page 134.</p> <p>If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0: Makes the front side the surface to be scanned.</p> <p>1: Makes the back side the surface to be scanned.</p>
ScanAreaBeginX	<p>Sets the initial value for the starting X coordinate of the scanning area for "BiSCNSetScanArea" on page 140.</p> <p>If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0 to 108 (unit: mm)</p>
ScanAreaBeginY	<p>Sets the initial value for the starting Y coordinate of the scanning area for "BiSCNSetScanArea" on page 140.</p> <p>If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0 to 252 (unit: mm)</p>
ScanAreaEndX	<p>Sets the initial value for the ending X coordinate of the scanning area for "BiSCNSetScanArea" on page 140.</p> <p>If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0 to 110, and larger than ScanAreaBeginX (unit: mm)</p>

Key name	Description
ScanAreaEndY	<p>Sets the initial value for the ending Y coordinate of the scanning area for "BiSCNSetScanArea" on page 140.</p> <p>If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0 to 254, and larger than ScanAreaBeginY (unit: mm)</p>
AutoSizeEnable	<p>Sets the initial value for "BiESCNSetAutoSize" on page 235.</p> <p>If it has not been set, it operates the same as when 1 is specified.</p> <p><Setting value></p> <p>0: Disables removal of black margins</p> <p>1: Enables removal of black margins</p>
CutSize	<p>Sets the initial value for "BiESCNSetCutSize" on page 237.</p> <p>If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0 to 1500 (unit: 0.1 mm)</p>
Deskew	<p>Sets the initial value for "BiESCNSetDeSkew" on page 241.</p> <p>If it has not been set, it operates the same as when 150 is specified.</p> <p><Setting value></p> <p>0: Enables skew correction</p> <p>65535: Disables skew correction</p> <p>1 to 8999: Performs skew correction at the specified threshold (unit: 0.01 degree)</p>
RotateEnable	<p>Sets the initial value for "BiESCNSetRotate" on page 239.</p> <p>If it has not been set, it operates the same as when 1 is specified.</p> <p><Setting value></p> <p>0: Disables rotation of the scanned image to a viewable direction.</p> <p>1: Enables rotation of the scanned image to a viewable direction.</p>

<Separated by scan units (check paper/card) and surfaces (front/back)>

Section name	Description
ScanSettingsCheckFront	Settings for the front side of the check paper unit
ScanSettingsCheckBack	Settings for the back side of the check paper unit
ScanSettingsCardFront	Settings for the front side of the card unit
ScanSettingsCardBack	Settings for the back side of the card unit

Key name	Description
ImageFormat	<p>Sets the initial value for "BiSCNSetImageFormat" on page 137. If it has not been set, it operates the same as when 1 is specified.</p> <p><Setting value></p> <p>0: CCITT(Group 4) compressed data in TIFF format 1: Uncompressed data in raster format 2: Uncompressed data in bitmap format 3: Uncompressed data in TIFF format 4: Highly compressed data in JPEG format (priority on size) 5: Standard compressed data in JPEG format 6: Low compressed data in JPEG format (priority on quality) 7: JPEG compressed data in TIFF format</p>
ColorDepth	<p>Sets the initial gradation value for "BiSCNSetImageQuality" on page 144. If it has not been set, it operates the same as when 8 is specified.</p> <p><Setting value></p> <p>1: 1 bit color depth (2-tone) 8: 8 bit color depth (256-tone) 24: 24 bit color depth (RGB 24bit)</p>
Threshold	<p>Sets the initial concentration threshold value for "BiSCNSetImageQuality" on page 144. If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>-128 to 127</p>
Color	<p>Sets the initial color value for "BiSCNSetImageQuality" on page 144. If it has not been set, it operates the same as when 48 is specified. (If ColorDepth is 24, the only value that can be specified is 49.)</p> <p><Setting value></p> <p>48: Monochrome 49: Color</p>
ExOption	<p>Sets the initial expansion function value for "BiSCNSetImageQuality" on page 144. If it has not been set, it operates the same as when 52 is specified.</p> <p><Setting value></p> <p>49: Manual adjustment of concentration 50: Sharpening 51: Custom sharpening 52: Sharpening and custom sharpening 53: Edge-preserving smoothing (fast)</p>

Key name	Description
ScanChannel	<p>Sets the initial value for "BiSCNSetImageTypeOption" on page 101. If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0: RGB color 1: Infrared 2: Red of RGB color 3: Green of RGB color 4: Blue of RGB color 8: RGB color and infrared (The card unit cannot be used.) 9: RGB grayscale 10: RGB grayscale and infrared (The card unit cannot be used.) 11: Red of RGB color and infrared (The card unit cannot be used.) 12: Green of RGB color and infrared (The card unit cannot be used.) 13: Blue of RGB color and infrared (The card unit cannot be used.)</p>
Brightness	<p>Sets the initial brightness adjustment value for "BiSCNSetImageAdjustment" on page 152. If it has not been set, it operates the same as when 0 is specified. (If 0 is specified, brightness is not imposed.)</p> <p><Setting value></p> <p>-100 to 100</p>
Contrast	<p>Sets the initial contrast adjustment value for "BiSCNSetImageAdjustment" on page 152. If it has not been set, it operates the same as when 0 is specified. (If 0 is specified, contrast is not imposed.)</p> <p><Setting value></p> <p>-100 to 100</p>
Gamma	<p>Sets the initial gamma correction value for "BiSCNSetImageAdjustment" on page 152. If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0: Does not perform gamma correction. 10: Does not perform gamma correction. 18: Makes the correction value 1.8 22: Makes the correction value 2.2</p>

Method for ejection during an error

Section name	
ErrorHandling	

Key name	Description
DoubleFeedPocket	<p>This sets the ejection method for when the double feed error is detected. If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0: Ejects to the main pocket 1: Ejects to the sub pocket</p>
MICRNoData	<p>This sets the ejection method for when the magnetic waveform detection error is detected. If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0: Ejects to the main pocket 1: Ejects to the sub pocket</p>
MICRBadData	<p>This sets the ejection method for when the unanalyzable character detection error is detected. If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0: Ejects to the main pocket 1: Ejects to the sub pocket</p>
Noise	<p>This sets the ejection method for when the external noise error is detected. If it has not been set, it operates the same as when 0 is specified.</p> <p><Setting value></p> <p>0: Ejects to the main pocket 1: Ejects to the sub pocket</p>
charSelect	<p>Sets the behavior for when the MICR magnetic waveform could not be recognized. If it has not been set, it operates the same as when 1 is specified.</p> <p><Setting value></p> <p>0: Cancels the MICR magnetic waveform recognition processing, and does not add a recognition result. 1: Replaces the MICR magnetic waveform parts that could not be recognized with a "?" and continues recognition processing.</p>
errorSelect	<p>Sets the initial value for scan processing behavior when an error is detected (double feed / magnetic waveform detection error / detection of letters that could not be analyzed / noise error). If it has not been set, it operates the same as when 1 is specified. For more information, refer to "errorSelect settings" on page 398.</p>

errorSelect settings

Set- ting value	Scan processing behavior during an error			
	Double feed	Magnetic waveform detection error	Detection of letters that could not be analyzed	Noise error
0	Cancelled	Cancelled	Cancelled	Cancelled
1	Continued	Continued	Continued	Continued
2	Continued	Cancelled	Cancelled	Cancelled
3				
4	Cancelled	Continued	Cancelled	Cancelled
5				
6	Continued	Continued	Cancelled	Cancelled
7				
8	Cancelled	Cancelled	Continued	Cancelled
9				
10	Continued	Cancelled	Continued	Cancelled
11				
12	Cancelled	Continued	Continued	Cancelled
13				
14	Continued	Continued	Continued	Cancelled
15				
16	Cancelled	Cancelled	Cancelled	Continued
17				
18	Continued	Cancelled	Cancelled	Continued
19				
20	Cancelled	Continued	Cancelled	Continued
21				
22	Continued	Continued	Cancelled	Continued
23				
24	Cancelled	Cancelled	Continued	Continued
25				
26	Continued	Cancelled	Continued	Continued
27				
28	Cancelled	Continued	Continued	Continued
29				
30	Continued	Continued	Continued	Continued
31				

Print settings

Section name	
PrintSettings	

Key name	Description
PrintStation	<p>Sets the initial value for "BiSetPrintStation" on page 174.</p> <p>If it has not been set, it operates the same as when 4 is specified.</p> <p><Setting value></p> <p>1: Sets the print station to roll paper (TM-S9000II and TM-S9000)</p> <p>2: Sets the print station to validation</p> <p>4: Sets the print station to the electronic endorsement back side.</p> <p>8: Sets the print station to the electronic endorsement front side.</p> <p>16: Sets the print station to the physical endorsement.</p> <p>32: Sets the print station to the feeder.</p>
CutSheetPrintAreaX	<p>Sets the initial value for the width for the possible printing region when printing a cut sheet.</p> <p>If it has not been set, it operates the same as when 2150 is specified.</p> <p><Setting value></p> <p>0 to 2150 (unit: 0.1 mm)</p>
CutSheetPrintAreaY	<p>Sets the initial value for the height for the possible printing region when printing a cut sheet.</p> <p>If it has not been set, it operates the same as when 508 is specified.</p> <p><Setting value></p> <p>0 to 508 (unit: 0.1 mm)</p>

Print settings (Roll paper) (TM-S9000II and TM-S9000)

Section name	
PrintSettingsRollpaper	

Key name	Description
Alignment	<p>Specifies the initial printing alignment setting.</p> <p>If it has not been set, it operates the same as when -1 is specified.</p> <p><Setting value></p> <p>-1: Align on the left</p> <p>-2: Center</p> <p>-3: Align on the right</p> <p>0 to 72: Print location shifts a specified amount from the left</p>
PrintSizeX	<p>Sets the initial print size value (width)</p> <p>If nothing is set, 0 is specified.</p> <p><Setting value></p> <p>0 to 72 (unit: mm)</p>
PrintSizeY	<p>Sets the initial print size value (height)</p> <p>If nothing is set, 0 is specified.</p> <p><Setting value></p> <p>Any whole number value (unit: mm)</p>

Print settings(Validation)

Section name	
PrintSettingsValidation	
Key name	Description
Alignment	<p>Sets the initial setting for print alignment. If it has not been set, it operates the same as when -1 is specified.</p> <p><Setting value></p> <p>-1: Align on the left</p> <p>-2: Center</p> <p>-3: Align on the left</p> <p>0 to 104: Print location shifts a specified amount from the left</p>
PrintSizeX	<p>Sets the initial print size value (width) If nothing is set, 0 is specified.</p> <p><Setting value></p> <p>0 to 104 (unit: mm)</p>
PrintSizeY	<p>Sets the initial print size value (height) If nothing is set, 0 is specified.</p> <p><Setting value></p> <p>0 to 27 (unit: mm)</p>

Print settings(Feeder)

Section name	
PrintSettingsFeeder	
Key name	Description
PrintSizeX	<p>Sets the initial print size value (width) If nothing is set, 0 is specified.</p> <p><Setting value></p> <p>Whole number from 0 to the "CutSheetPrintAreaX" on page 399 value/10.</p>
PrintSizeY	<p>Sets the initial print size value (height) If nothing is set, 0 is specified.</p> <p><Setting value></p> <p>Whole number from 0 to the "CutSheetPrintAreaY" on page 399 value/10.</p>

Print settings(Electronic Endorsement front side/back side)

Section name	Description
PrintSettingsEEndorseFront	Set the print settings for the electronic endorsement front side
PrintSettingsEEndorseBack	Set the print settings for the electronic endorsement back side

Key name	Description
PrintPositionX	Sets the initial value for print starting position (Y coordinate). If nothing is set, 0 is specified. <Setting value> Any whole number value (unit: mm)
PrintPositionY	Sets the initial value for print starting position (X coordinate). If nothing is set, 0 is specified. <Setting value> Any whole number value (unit: mm)
PrintSizeX	Sets the initial print size value (width) If nothing is set, 0 is specified. <Setting value> 0 to 255 (unit: mm)
PrintSizeY	Sets the initial print size value (height) If nothing is set, 0 is specified. <Setting value> 0 to 255 (unit: mm)

Print settings(Physical Endorsement)

Section name
PrintSettingsPEndorse

Key name	Description
PrintSizeX	Sets the initial print size value (width) If nothing is set, 0 is specified. <Setting value> Whole number from 0 to the "CutSheetPrintAreaX" on page 399 value/10.
PrintSizeY	Sets the initial print size value (height) If nothing is set, 0 is specified. <Setting value> Whole number from 0 to the "CutSheetPrintAreaY" on page 399 value/10.

OCRA/B recognition

Section name
OCRAB

Key name	Description
Origin	<p>Specifies the starting point for the OCR target area.</p> <p>If nothing is set, 0 is specified.</p> <p><Setting value></p> <p>0: Starts in the upper left</p> <p>1: Starts in the lower left</p> <p>2: Starts in the upper right</p> <p>3: Starts in the lower right</p>

Status

Section name
Status

Key name	Description
OfflineCode	<p>Specifies the OfflineCode device acquired with "BiGetOfflineCodeByIndex" on page 274. If nothing is set, 2 is selected.</p> <p><Setting value></p> <p>1: TM-S1000 (OfflineCode for backward compatibility)</p> <p>2: TM-S9000II, TM-S2000II, TM-S9000 or TM-S2000</p>
ASBStatus	<p>Specifies the Device Status device acquired with "BiGetStatus" on page 85. If nothing is set, 2 is selected.</p> <p><Setting value></p> <p>0: TM-J9000 (Device Status for backward compatibility)</p> <p>1: TM-S1000 (Device Status for backward compatibility)</p> <p>2: TM-S9000II, TM-S2000II, TM-S9000 or TM-S2000</p>

Template printing setting file

This file (*.ini) defines the print area of the template printing.

The print area defined in this setting file is read by "[BiLoadTemplatePrintArea](#)" on page 217.

Template printing setting file is not a prepared file, unlike other setting files.

You can prepare it as you like.

Setting file location
Any folder (Specify the full path for the setting file in " BiLoadTemplatePrintArea " on page 217 and load it.)
Format for the setting file
ini file format
Setting file name
Any file name

Settings for the template print setting file



If all keys are not inside the section, or values outside of the definition are set, when printing the template, it will be read as an invalid value and discarded.

Section name	Description
Any section name	You can enter a section name with a length of up to 127 bytes. The section name specified here becomes the area name specified in " BiSet-TemplatePrintArea " on page 220.

Key name	Description
Measure	Specifies the increment for setting values. <Setting value> 0: mm 1: 0.1 inch
OriginX	Specifies the X coordinate of the starting point for the print area. (Makes the upper left of the possible print area the starting point) <Setting value> Any positive whole number
OriginY	Specifies the Y coordinate of the starting point for the print area. (Makes the upper left of the possible print area the starting point) <Setting value> Any positive whole number
Width	Specifies the width of the print area. <Setting value> Any positive whole number
Height	Specifies the height of the print area. <Setting value> Any positive whole number

Key name	Description
Rotate	<p>Specifies the rotation.</p> <p><Setting value></p> <p>0: Does not rotate.</p> <p>90: Rotates 90 degrees, counterclockwise</p> <p>180: Rotates 180 degrees, counterclockwise</p> <p>270: Rotates 270 degrees, counterclockwise</p>

Setting for the IQA function

The file that settings for the IQA function.

API settings file location
C:\Program Files\EPSON\TMSDriver\bin (If both 32 bit/64 bit are in a 64 bit OS, the 64 bit version is stored in the x64bin folder.)
API settings file name
IQA.ini

Log Function

This chapter provides a description of how to generate a log file.

Overview

The log file is recorded after tracing the interaction of the application and the TM-S9000/S2000 driver. This log file records the functions executed by the application, parameters, and acquired data. It is useful for efficient application development and analyzing errors.



A single file can have a size up to 2 MB. If this limit is exceeded, the file name is changed and a new log file is created. For details, refer to ["Log file output" on page 408](#).

Settings for the log function

The settings for the log function are set in the log setting file.

Log setting file location	
C:\Program Files\EPSON\TMSDriver\bin (If both 32 bit/64 bit are in a 64 bit OS, the 64 bit version is stored in the x64bin folder.)	
Log setting file	
LogSettings.ini	

Log setting file settings

Section name	
ApiTrace	
Key name	Details
LogEnabled	<p>Sets whether the API log output is used or not. If it has not been set, it operates the same as when 0 is specified. The initial value is 0.</p> <p><Setting value></p> <p>0: Disables the API log output 1: Enables the API log output</p>
PersonallInfoEnabled	<p>Sets protection for personal information (MICR magnetic data) output in the API log (by replacing information with an "*"). If it has not been set, it operates the same as when 0 is specified. The initial value is 0.</p> <p><Setting value></p> <p>0: Outputs after replacing personal information (MICR magnetic data) with a character string "*" 1: Outputs personal information (MICR magnetic data)</p>
LogFileNum	<p>Sets the maximum number of API log output files. If not set, the maximum number of files is 3. The initial value is 3.</p> <p><Setting value></p> <p>1 to 99</p>

Log file output



For more about how to analyze the contents of the output log file, please consult your dealer.

Output destination for the log file

%ALLUSERSPROFILE%\EPSON\TMSDriver\Trace

Log file name

- File name
TMSDriverApiTrace_YYYY-MM-DD_XX.log

Format	Details
MM	The month in the log output date is in two digits.
DD	The day in the log output date is in two digits.
YYYY	The year in the log output date is in four digits.
XX	The number indicating which number file was made on the log output data is in two digits.



- Up to 2 MB can be recorded into one log file. If this is exceeded, the number of the file name is changed and a new log file is made.
- If the number of log files generated exceeds the maximum number of files set in the log setting file, files are deleted starting with the oldest.

Output example

```
2011-10-19 10:01:20.277 TMSDriverApiTrace [000011C0] FUI, ,BiOpenMonPrinter,00000002,TM-S9000U,1.00.001
2011-10-19 10:01:22.324 TMSDriverApiTrace [00001384] ASB,USB3;,00000001,4B650014
2011-10-19 10:01:22.324 TMSDriverApiTrace [000011C0] FUI,USB3;,BiOpenMonPrinter,00000002,TM-S9000U,<00000001>
2011-10-19 10:01:22.324 TMSDriverApiTrace [00001384] ISB,USB3;,00000001,00004040
2011-10-19 10:01:22.339 TMSDriverApiTrace [00001384] ASB,USB3;,00000001,4B650014
2011-10-19 10:01:22.339 TMSDriverApiTrace [000011C0] FUI,USB3;,BiSCNMICRSetStatusBackFunction,00000001,004BB1C5
2011-10-19 10:01:22.339 TMSDriverApiTrace [00001384] ISB,USB3;,00000001,00004040
2011-10-19 10:01:22.339 TMSDriverApiTrace [000011C0]
    FUI,USB3;,BiSCNMICRSetStatusBackFunction,00000001,004BB1C5,<00000000>
2011-10-19 10:01:25.995 TMSDriverApiTrace [00001384] ASB,USB3;,00000001,4B250014
```

Appendix

Describes the Software License.

Independent JPEG Group

EPSON TM-S9000/S2000 Driver is based in part on the work of the Independent JPEG Group.

libtiff

Copyright (c) 1988-1997 Sam Leffler

Copyright (c) 1991-1997 Silicon Graphics, Inc.

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that (i) the above copyright notices and this permission notice appear in all copies of the software and related documentation, and (ii) the names of Sam Leffler and Silicon Graphics may not be used in any advertising or publicity relating to the software without the specific, prior written permission of Sam Leffler and Silicon Graphics.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL SAM LEFFLER OR SILICON GRAPHICS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.