# SII Print Class Library for Android™
# Application Programmer's Guide

Rev.06

[Products]

DSP-A01 Series

Seiko Instruments Inc.

| | |
|---|---|
| Rev.01 | July 2019 |
| Rev.02 | March 2022 |
| Rev.03 | April 2023 |
| Rev.04 | March 2024 |
| Rev.05 | May 2025 |
| Rev.06 | June 2026 |

Copyright © 2019-2026 by Seiko Instruments Inc. All rights reserved.

Android™ is a trademark of Google LLC.
The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc.
Oracle and Java® are registered trademarks of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Seiko Instruments Inc. (hereinafter referred to as ”SII”) has prepared this manual for use by SII personnel, licensees, and customers. The information contained herein is the property of SII and shall not be reproduced in whole or in part without the prior written approval of SII.

SII reserves the right to make changes without notice to the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented, including but not limited to typographical, arithmetic, or listing errors.

# Introduction

This manual describes "SII Print Class Library for Android™" (hereinafter referred to as "SII print class library") provided by Seiko Instruments Inc. (hereinafter referred to as "SII").

## Target Product

The product covered by this manual is as follows.

| Product | Interface | Description in this manual |
|---|---|---|
| DSP-A01-1 | USB | Display |

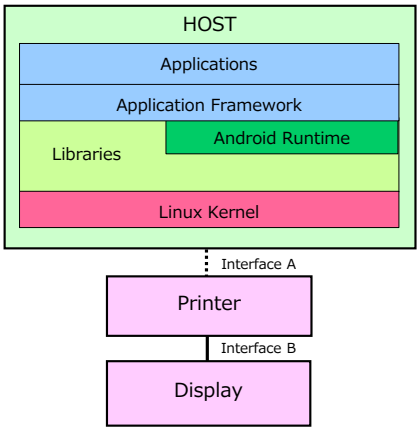
## Target Printer

The printers supported by SII print class library are listed below.

| Printer | Interface A | Display | Interface B |
|---|---|---|---|
| RP-F10 Series | Bluetooth | DSP-A01 Series | USB |
| | USB | | |
| | TCP/IP | | |

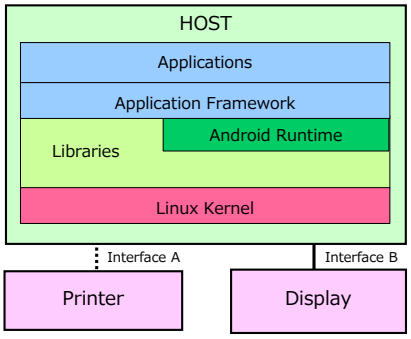
There are 2 types for connecting Display.

1. When Display is used by connecting to the target printer (hereinafter described as "use via a printer")




Configuration image

2. When Display is used alone (hereinafter described as "use alone")




Configuration image

## Terms

The terms used in this manual are defined as below.

### Printer

| Term | Description |
|---|---|
| Technical Reference for Printer | The following Technical Reference.<br>・"RP-F10 SERIES THERMAL PRINTER TECHNICAL REFERENCE" |
| Printer command | Command for controlling the printer described in "Technical Reference for Printer". |

### Display

| Term | Description |
|---|---|
| Technical Reference for Display | The following Technical Reference.<br>・"DSP-A01 SERIES CUSTOMER DISPLAY TECHNICAL REFERENCE" |
| Display command | Command for controlling Display described in "Technical Reference for Display". |
| Slide | The image data of the screen size (Width 480 px × Height 272 px). Displays as a standby screen and as a backscreen superimposed on templates. |
| Template | The stylized form having elements that can set attributes such as drawing areas and mapping positions. The elements include text elements (text data), img elements (image data), barcode elements (barcode data), and qr elements (QR Code data).<br>To register templates, define a map ID each for an element to place. Registered image data or text data is shown on Display by updating the screen after selecting a template and specifying its map ID.<br>The data in the template is required to be specified XML file format. The maximum size of template data is width 480 px × height 272 px. |
| Map ID | An ID defined to an element which is holding positional information or modification information when a template is registered. |
| Macro | A function to register multiple APIs in order of execution, and execute automatically when an event occurs. |
| Event | An event which is defined by "Event Notification" in Display commands. |

# Table of Contents


## Chapter 1 Product Overview 1-1

1.1 Functions Provided by SII Print Class Library ..... 1-1
1.2 SII Print Class Library Overview ..... 1-1
1.2.1 SII Print Class Library Configuration ..... 1-1
1.2.2 Functions Provided by Library ..... 1-2
1.2.3 Registered Data in Display at Shipping ..... 1-2

## Chapter 2 Product Specifications 2-1

2.1 Operating Environment ..... 2-1

## Chapter 3 How to Use Library 3-1

3.1 Android Application Development Environment ..... 3-1
3.2 Provided Files ..... 3-2
3.3 Build Library into Android Studio Project ..... 3-3
3.4 Use Developed Android Application on Android Device ..... 3-5
3.5 Precautions ..... 3-5

## Chapter 4 Functions of Library 4-1

4.1 Log File Output Function ..... 4-1
4.1.1 How to Set Log Output ..... 4-1
4.1.2 Log Output Settings ..... 4-1
4.1.3 Log File ..... 4-2
4.2 API Reference ..... 4-3
4.2.1 PrinterManager Class ..... 4-4
(1) Method List ..... 4-4
(2) Constant List ..... 4-6
① Device model ..... 4-6
② International character set ..... 4-6
③ Codepage ..... 4-7
④ Port type ..... 4-7
⑤ Display response type ..... 4-8
(3) Enumerated Constant List ..... 4-8
① Bold printing (CharacterBold) ..... 4-8
② Underline (CharacterUnderline) ..... 4-8
③ Character font (CharacterFont) ..... 4-8
④ Character scale (CharacterScale) ..... 4-9
⑤ Alignment (PrintAlignment) ..... 4-9
⑥ Module size (ModuleSize) ..... 4-10
⑦ Error correction level (ErrorCorrection) ..... 4-10
⑧ Memory area (MemoryArea) ..... 4-10

⑨ Registered font (RegisteredFont) ..... 4-10
⑩ QR data mode (QrDataMode) ..... 4-11
⑪ QR quiet zone (QrQuietZone) ..... 4-11
⑫ Macro registration processing (MacroRegistrationFunction) ..... 4-11
(4) Method Details ..... 4-12
PrinterManager Constructor ..... 4-12
connect Start communicating with printer (Bluetooth) ..... 4-12
connect Start communicating with device (USB) ..... 4-12
connect Start communicating with printer (TCP/IP) ..... 4-13
disconnect Stop communicating with device ..... 4-13
sendBinary Send binary data ..... 4-14
sendDataFile Send specified file ..... 4-14
startDiscoveryPrinter Start printer search (Bluetooth) ..... 4-15
startDiscoveryPrinter Start printer search (USB) ..... 4-15
startDiscoveryPrinter Start printer search (TCP/IP) ..... 4-16
startDiscoveryDevice Start device search (USB) ..... 4-16
cancelDiscoveryPrinter Cancel printer search ..... 4-17
cancelDiscoveryDevice Cancel device search ..... 4-17
getFoundPrinter Get found printer information ..... 4-17
getFoundDevice Get found device information ..... 4-17
getSendTimeout Get send timeout period ..... 4-17
setSendTimeout Set send timeout period ..... 4-18
getReceiveTimeout Get receive timeout period ..... 4-18
setReceiveTimeout Set receive timeout period ..... 4-18
getInternationalCharacter Get international character set ..... 4-18
setInternationalCharacter Set international character set ..... 4-19
getCodePage Get codepage ..... 4-19
setCodePage Set codepage ..... 4-19
getPrinterModel Get device model ..... 4-20
getPortType Get connecting port type ..... 4-20
isConnect Verify connection state with device ..... 4-20
getSocketKeepingTime Get socket keeping time ..... 4-20
setSocketKeepingTime Set socket keeping time ..... 4-20
getVersion Get SDK version ..... 4-21
defragment Optimize memory area ..... 4-21
initializeMemoryArea Initialize memory area ..... 4-21
showTemplate Display template ..... 4-22
showSlide Display slide ..... 4-23
enterStandbyMode Display standby ..... 4-24
executeMacro Execute macro ..... 4-24
turnOnScreen Turn on/off screen ..... 4-25
selectTemplate Select template ..... 4-25
setTemplateImageData Set image data ..... 4-26
selectTemplateTextObject Select text element ..... 4-26

setTemplateTextAlignment
Alignment of text data....4-27
setTemplateTextLeftMargin
Set left margin of text data....4-27
setTemplateTextLineSpacing
Set line spacing of text data....4-28
setTemplateTextBold Set bold printing of text data....4-29
setTemplateTextUnderline
Set underline of text data....4-29
setTemplateTextSize Set character size of text data....4-30
setTemplateTextFont Set character font of text data....4-30
setTemplateTextRegisteredFont
Set registered font of text data....4-31
setTemplateTextRightSpacing
Set right space of text data....4-31
setTemplateTextColor Set character color of text data....4-32
setTemplateTextData Input text data....4-33
setTemplateBarcodeData
Input barcode data....4-33
setTemplateQrCodeData
Input QR Code data....4-34
registerTemplate Register template....4-35
unregisterTemplate Delete template....4-36
registerImageData Register image data....4-37
unregisterImageData Delete image data....4-38
registerSlideData Register slide data....4-39
unregisterSlideData Delete slide data....4-40
registerUserDefinedCharacter
Register user-defined character....4-40
unregisterUserDefinedCharacter
Delete user-defined character....4-41
registerOptionFont Register optional font....4-42
unregisterOptionFont Delete optional font....4-43
controlMacroRegistration
Start/End of macro registration....4-43
getDisplayResponse Get various response from Display....4-45
4.2.2 PrinterEvent Class....4-47
(1) Method List....4-47
(2) End event constant....4-47
(3) Method Details....4-47
getEventType Get end event....4-47
4.2.3 PrinterListener Interface....4-48
(1) Method List....4-48
(2) Method Details....4-48
finishEvent End event of device search....4-48
4.2.4 PrinterInfo Class....4-49
(1) Method List....4-49
(2) Method Details....4-49

getDevicePath Get device path ....4-49
4.2.5 PrinterException Class ....4-50
(1) Method List ....4-50
(2) Constant List ....4-50
① Error code ....4-50
(3) Method Details ....4-51
PrinterException Constructor ....4-51
getErrorCode Get error codes ....4-51

Chapter 5 Sample Program 5-1

5.1 Screen Layout ....5-1
5.1.1 Main Screen ....5-1
5.1.2 [Setting] Screen ....5-3
5.2 Precaution ....5-3

Appendix A Character Set A-1

A.1 Codepage Table (Character Code Table) .... A-1
A.2 International Character Set .... A-11

# Chapter 1
# Product Overview

This chapter describes the product overview of SII print class library.

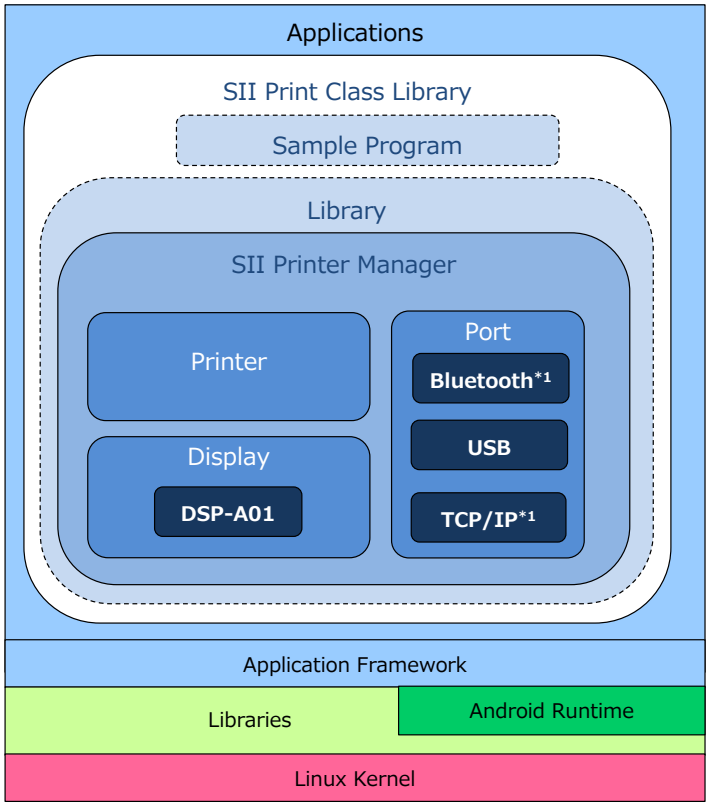
## 1.1 Functions Provided by SII Print Class Library

The SII print class library including the library and the sample program provides the functions to use Display described in "Introduction Target Product" in Android applications.
In addition, the SII print class library provides the library sample program in Android Studio project.

## 1.2 SII Print Class Library Overview

### 1.2.1 SII Print Class Library Configuration

The library and sample program in the SII print class library are indicated with dashed lines in the figure below.




*1: It is valid only when Display is used via a printer.

### 1.2.2 Functions Provided by Library

By using the library, Android applications can easily send display data and Display commands to a device through communication port (Bluetooth*1, USB or TCP/IP*1) on an Android device.
In addition, the applications can get the Display response.

The library provides the following functions:

- Connecting to / disconnecting from a device
- Sending data to a device (display data and/or Display commands*2)
- Sending a data file to a device (display data and/or Display commands*2)
- Getting various responses from Display
- Screen display control
- Outputting a log file

*1: It is valid only when Display is used via a printer.
*2: To read responses from Display, use **getDisplayResponse**.

### 1.2.3 Registered Data in Display at Shipping

In case of connecting Display to the printer to use templates, registered data in Display at the shipping may be added or changed without prior notice for the quality improvement.
A template which is specified appropriate encode is required to use depending on language settings or character codes to specify. See SII's Website for details about the data to be registered at the shipping.
https://www.sii-ps.com/dspa01/

# Chapter 2
# Product Specifications

This chapter describes the product specifications of the library.

## 2.1 Operating Environment

Operating environment for the library is shown in the following table.

<table>
<tr><td rowspan="2">Display</td><td colspan="2">Model</td><td>DSP-A01</td></tr>
<tr><td colspan="2">Communication Interface</td><td>USB</td></tr>
<tr><td rowspan="14">Android Device</td><td colspan="2">Communication Port</td><td>USB*1</td></tr>
<tr><td rowspan="13">OS</td><td>Android 7.0 (API 24)</td><td rowspan="13">Supported</td></tr>
<tr><td>Android 7.1 (API 25)</td></tr>
<tr><td>Android 8.0 (API 26)</td></tr>
<tr><td>Android 8.1 (API 27)</td></tr>
<tr><td>Android 9.0 (API 28)</td></tr>
<tr><td>Android 10.0 (API 29)</td></tr>
<tr><td>Android 11.0 (API 30)</td></tr>
<tr><td>Android 12.0 (API 31)</td></tr>
<tr><td>Android 12.1 (API 32)</td></tr>
<tr><td>Android 13.0 (API 33)</td></tr>
<tr><td>Android 14.0 (API 34)</td></tr>
<tr><td>Android 15.0 (API 35)</td></tr>
<tr><td>Android 16.0 (API 36)</td></tr>
<tr><td colspan="3">Supported Language</td><td>Japanese/English</td></tr>
</table>

*1: The Android device needs to support USB host function.

# Chapter 3
# How to Use Library

This chapter describes the development environment for Android application and how to use the library.

## 3.1 Android Application Development Environment

In order to develop Android applications, the following tools are required.
See each of the following URLs for more details.

- Android Studio

  https://developer.android.com/studio/index.html

- USB driver for Windows (When developing in Windows environment)

  https://developer.android.com/studio/run/oem-usb.html

The description in and after this chapter is on the premise that the environment where each tool is available is prepared.

## 3.2 Provided Files

The file configuration of the SII print class library is as follows.

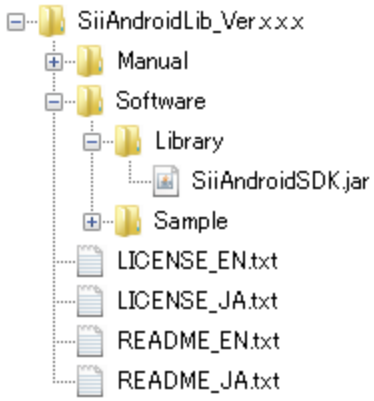



**Figure 3-1**

The file format of the library is JAR. The file name of the library is SiiAndroidSDK.jar.

## 3.3 Build Library into Android Studio Project

This section describes how to build the SII print class library into Android Studio project.

See "Chapter 5 Sample Program" for the sample program included in the SII print class library.

(1) Create a project in Android Studio and copy the library file (SiiAndroidSDK.jar) to the [libs] folder. When the [libs] folder is not automatically created, add the folder manually.
For sample programs included in the SII print class library, the folder is "\Sample\app\libs".

(2) After adding the library, the view looks like Figure 3-2.

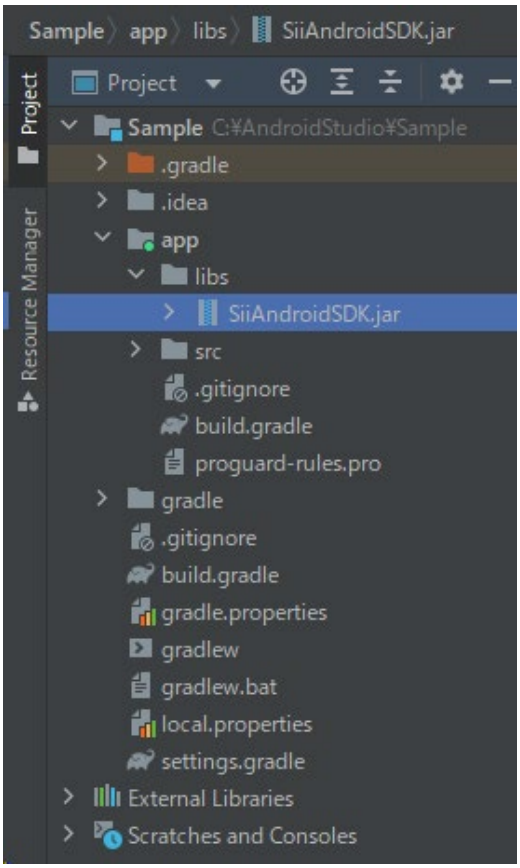



**Figure 3-2**

(3) Add the following to dependencies{} of build.gradle(:app) in the application.

```
implementation 'com.journeyapps:zxing-android-embedded:3.4.0@aar'
implementation 'com.google.zxing:core:3.4.1'
implementation files ('libs/SiiAndroidSDK.jar')
```

(4) Add the following to the beginning of the class file that uses the print class library.
(Import xxxx according to the function to use.)

```
import com.seikoinstruments.sdk.thermalprinter.PrinterManager;
import com.seikoinstruments.sdk.thermalprinter.xxxx;
```

(5) Add the following permission declaration to the application manifest (AndroidManifest.xml).
Also, implement a process (requestPermissions()) that requests the appropriate permissions for the application.

[When using Bluetooth]

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
<uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
```

[When using TCP/IP]

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

By completing these procedures, the library function becomes available.

## 3.4 Use Developed Android Application on Android Device

In order to use the developed Android applications on the Android device, make the following settings on the Android device.

(1) Select [Settings], [Developer options], and turn on [USB debugging]. (Figure 3-3)

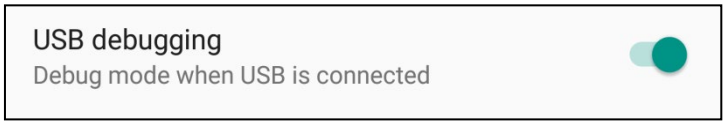



**Figure 3-3**

## 3.5 Precautions

- **About Scoped Storage**

  "Scoped Storage" that is introduced in Android 10 distinguishes between app-specific storage and external storage.
  When targeting Android 10 (API 29) or later, files that do not correspond to media files in the external storage cannot be handled directly. Files that do not correspond to media files can be handled by using the "Storage Access Framework".
  See below for details of Scoped Storage.

  - Data and file storage overview
    https://developer.android.com/training/data-storage

# Chapter 4
# Functions of Library

This chapter describes the APIs of each class implemented in the library.

## 4.1 Log File Output Function

The logs can be retrieved and the log files can be output using the library.

### 4.1.1 How to Set Log Output

Log output settings can be configured by adding the config.ini file with the following content to the specific directory of the Android application that incorporates the library (e.g. internal shared storage \Android\data\<package name>\files).

config.ini

```
LOGLEVEL=x
LOGSIZEMAX=xMB
LOGOUTPUT=x
```

Reference: See "4.1.2 Log Output Settings" for details on the settings for x.

### 4.1.2 Log Output Settings

| Item | Description | Settings |
|---|---|---|
| LOGLEVEL | Log level | 0 : Not record the log.<br>1 : Records an error log when **PrinterException** is thrown.<br>2 : Records API execution history. |
| LOGSIZEMAX | Log file maximum size | 1MB : Log file maximum size is 1 MB<br>5MB : Log file maximum size is 5 MB<br>10MB : Log file maximum size is 10 MB<br>50MB : Log file maximum size is 50 MB |
| LOGOUTPUT | Logcat output enabled/disabled | 0 : Logcat output is disabled<br>1 : Logcat output is enabled |

### 4.1.3 Log File

Log files are saved as local files in the Android application that incorporates the library.

Log file name : PrinterManagerX.log (range of X is 0 to 4)

The 1st log file is created as PrinterManager0.log. If the log file maximum size is exceeded, changes the file name to PrinterManager1.log and creates a new PrinterManager0.log.

Up to 5 log files can be created.

## 4.2 API Reference

The package of the library is **com.seikoinstruments.sdk.thermalprinter**.
**com.seikoinstruments.sdk.thermalprinter** includes the following classes.

| Class Name | Description | Supported *1 |
|---|---|---|
| **PrinterManager** | Provides the API used for communication with the device and for displaying.<br>See "**4.2.1 PrinterManager Class**" for more details. | ✓ |
| **PrinterEvent** | Provides the API that gets the end event when **startDiscoveryPrinter** is completed.<br>See "**4.2.2 PrinterEvent Class**" for more details. | ✓ |
| **PrinterListener** | Interface for getting the end event when **startDiscoveryPrinter** or **startDiscoveryDevice** is completed.<br>See "**4.2.3 PrinterListener Interface**" for more details. | ✓ |
| **PrinterInfo** | Stores the device information found by **startDiscoveryPrinter** or **startDiscoveryDevice**.<br>See "**4.2.4 PrinterInfo Class**". | ✓ |
| **PrinterException** | Exception class that is thrown at API call.<br>See "**4.2.5 PrinterException Class**" for more details. | ✓ |

*1: ✓: Supported, -: Not supported in DSP-A01

### 4.2.1 PrinterManager Class

(1) Method List

Methods provided by the **PrinterManager** class are shown in the following table.

| Name | Description | Supported [*1] |
|---|---|---|
| **PrinterManager** | Constructor | ✓ |
| **connect** | Start communicating with printer (Bluetooth) | ✓ |
| **connect** | Start communicating with device (USB) | ✓ |
| **connect** | Start communicating with printer (TCP/IP) | ✓ |
| **disconnect** | Stop communicating with device | ✓ |
| **sendBinary** | Send binary data | ✓ |
| **sendDataFile** | Send specified file | ✓ |
| **startDiscoveryPrinter** | Start printer search (Bluetooth) | ✓ |
| **startDiscoveryPrinter** | Start printer search (USB) | ✓ |
| **startDiscoveryPrinter** | Start printer search (TCP/IP) | ✓ |
| **startDiscoveryDevice** | Start device search (USB) | ✓ |
| **cancelDiscoveryPrinter** | Cancel printer search | ✓ |
| **cancelDiscoveryDevice** | Cancel device search | ✓ |
| **getFoundPrinter** | Get found printer information list | ✓ |
| **getFoundDevice** | Get found device information list | ✓ |
| **getSendTimeout** | Get send timeout period | ✓ |
| **setSendTimeout** | Set send timeout period | ✓ |
| **getReceiveTimeout** | Get receive timeout period | ✓ |
| **setReceiveTimeout** | Set receive timeout period | ✓ |
| **getInternationalCharacter** | Get international character set | ✓ |
| **setInternationalCharacter** | Set international character set | ✓ |
| **getCodePage** | Get codepage | ✓ |
| **setCodePage** | Set codepage | ✓ |
| **getPrinterModel** | Get device model | ✓ |
| **getPortType** | Get connecting port type | ✓ |
| **isConnect** | Verify connection state with device | ✓ |
| **getSocketKeepingTime** | Get socket keeping time | ✓ |
| **setSocketKeepingTime** | Set socket keeping time | ✓ |
| **getVersion** | Get SDK version | ✓ |
| **defragment** | Optimize memory area | ✓ |
| **initializeMemoryArea** | Initialize memory area | ✓ |
| **showTemplate** | Display template | ✓ |
| **showSlide** | Display slide | ✓ |
| **enterStandbyMode** | Display standby | ✓ |
| **executeMacro** | Execute macro | ✓ |
| **turnOnScreen** | Turn on/off screen | ✓ |
| **selectTemplate** | Select template | ✓ |

| Name | Description | Supported *1 |
|---|---|---|
| **setTemplateImageData** | Set image data | ✓ |
| **selectTemplateTextObject** | Select text element | ✓ |
| **setTemplateTextAlignment** | Alignment of text data | ✓ |
| **setTemplateTextLeftMargin** | Set left margin of text data | ✓ |
| **setTemplateTextLineSpacing** | Set line spacing of text data | ✓ |
| **setTemplateTextBold** | Set bold printing of text data | ✓ |
| **setTemplateTextUnderline** | Set underline of text data | ✓ |
| **setTemplateTextSize** | Set character size of text data | ✓ |
| **setTemplateTextFont** | Set character font of text data | ✓ |
| **setTemplateTextRegisteredFont** | Set registered font of text data | ✓ |
| **setTemplateTextRightSpacing** | Set right space of text data | ✓ |
| **setTemplateTextColor** | Set character color of text data | ✓ |
| **setTemplateTextData** | Input text data | ✓ |
| **setTemplateBarcodeData** | Input barcode data | ✓ |
| **setTemplateQrCodeData** | Input QR Code data | ✓ |
| **registerTemplate** | Register template | ✓ |
| **unregisterTemplate** | Delete template | ✓ |
| **registerImageData** | Register image data | ✓ |
| **unregisterImageData** | Delete image data | ✓ |
| **registerSlideData** | Register slide data | ✓ |
| **unregisterSlideData** | Delete slide data | ✓ |
| **registerUserDefinedCharacter** | Register user-defined character | ✓ |
| **unregisterUserDefinedCharacter** | Delete user-defined character | ✓ |
| **registerOptionFont** | Register optional font | ✓ |
| **unregisterOptionFont** | Delete optional font | ✓ |
| **controlMacroRegistration** | Start/End macro registration | ✓ |
| **getDisplayResponse** | Get various responses from Display | ✓ |

*1: ✓: Supported, -: Not supported in DSP-A01

(2) Constant List

① Device model

Constants used for starting communicating with the device and getting the device model are shown in the following table.

| Constant Name | Description | Value |
|---|---|---|
| **PRINTER_MODEL_RP-FG10**[*1] | RP-F10 | 301 |
| **PRINTER_MODEL_DSP_A01** | DSP-A01 | 303 |
| **PRINTER_MODEL_DEFAULT** | Default value of device model | 284 |

*1: It is valid only when Display is used via a printer.

② International character set

Constants used for setting/getting the international character set are shown in the following table.

| Constant Name | Description | Value |
|---|---|---|
| **COUNTRY_USA** | USA | 0 |
| **COUNTRY_FRANCE** | France | 1 |
| **COUNTRY_GERMANY** | Germany | 2 |
| **COUNTRY_ENGLAND** | United Kingdom | 3 |
| **COUNTRY_DENMARK_1** | Denmark I | 4 |
| **COUNTRY_SWEDEN** | Sweden | 5 |
| **COUNTRY_ITALY** | Italy | 6 |
| **COUNTRY_SPAIN** | Spain I | 7 |
| **COUNTRY_JAPAN** | Japan | 8 |
| **COUNTRY_NORWAY** | Norway | 9 |
| **COUNTRY_DENMARK_2** | Denmark II | 10 |
| **COUNTRY_SPAIN_2** | Spain II | 11 |
| **COUNTRY_LATIN_AMERICA** | Latin America | 12 |
| **COUNTRY_ARABIA** | Arabia | 17 |

③ Codepage

Constants used for setting/getting the codepage are shown in the following table.

| Constant Name | Description | Value |
|---|---|---|
| **CODE_PAGE_437** | USA, Standard Europe (Code Page437) | 0 |
| **CODE_PAGE_KATAKANA** | Katakana | 1 |
| **CODE_PAGE_850** | Multilingual (Code Page850) | 2 |
| **CODE_PAGE_860** | Portuguese (Code Page860) | 3 |
| **CODE_PAGE_863** | Canadian-French (Code Page863) | 4 |
| **CODE_PAGE_865** | Nordic (Code Page865) | 5 |
| **CODE_PAGE_857**[*1] | Turkish (Code Page857) | 13 |
| **CODE_PAGE_737** | Greek (Code Page737) | 14 |
| **CODE_PAGE_1252** | Latin (Code Page1252) | 16 |
| **CODE_PAGE_866** | Russian (Code Page866) | 17 |
| **CODE_PAGE_852** | Eastern Europe (Code Page852) | 18 |
| **CODE_PAGE_858** | Euro (Code Page858) | 19 |
| **CODE_PAGE_855** | Cyrillic (Code Page855) | 34 |
| **CODE_PAGE_864**[*1*2] | Arabic (Code Page864) | 37 |
| **CODE_PAGE_1250** | Central European (Code Page1250) | 45 |
| **CODE_PAGE_1251** | Cyrillic (Code Page1251) | 46 |
| **CODE_PAGE_1253**[*3] | Greek (Code Page1253) | 47 |
| **CODE_PAGE_1254** | Turkish (Code Page1254) | 48 |

*1: 20ACh of the Unicode cannot be displayed.
*2: Font B cannot be displayed.
*3: 00AAh of the Unicode cannot be displayed.

④ Port type

Constants used for starting communicating with the device and getting the connecting port type are shown in the following table.

| Constant Name | Description | Value |
|---|---|---|
| **PRINTER_TYPE_BLUETOOTH**[*1] | Bluetooth | 0 |
| **PRINTER_TYPE_USB** | USB | 1 |
| **PRINTER_TYPE_TCP**[*1] | TCP/IP | 2 |

*1: It is valid only when Display is used via a printer.

⑤ Display response type

Constants used for getting various responses from Display are shown in the following table.

| Constant Name | Description | Value |
|---|---|---|
| **DISPLAY_RESPONSE_REQUEST** | Execution response request | 0 |
| **DISPLAY_RESPONSE_USER_AREA** | Send remaining capacity of user area | 1 |
| **DISPLAY_RESPONSE_TEMPLATE_ID_LIST** | Send template ID | 2 |
| **DISPLAY_RESPONSE_IMAGE_ID_LIST** | Send image ID | 3 |
| **DISPLAY_RESPONSE_SLIDE_ID_LIST** | Send slide ID | 4 |
| **DISPLAY_RESPONSE_TEMPLATE_LABEL** | Send template name | 5 |
| **DISPLAY_RESPONSE_IMAGE_LABEL** | Send image name | 6 |
| **DISPLAY_RESPONSE_SLIDE_LABEL** | Send slide name | 7 |

(3) Enumerated Constant List

① Bold printing (CharacterBold)

Constants of enumerated type used for bold printing are shown in the following table.

| Constant Name | Description |
|---|---|
| **BOLD_CANCEL** | Cancel bold printing |
| **BOLD** | Specify bold printing |

② Underline (CharacterUnderline)

Constants of enumerated type used for underlining are shown in the following table.

| Constant Name | Description |
|---|---|
| **UNDERLINE_CANCEL** | Cancel underline print |
| **UNDERLINE_1** | Specify 1-dot width underline print |

③ Character font (CharacterFont)

Constants of enumerated type used for character font are shown in the following table.

| Constant Name | Description |
|---|---|
| **FONT_A** | Font A (24 × 12) |
| **FONT_B** | Font B (16 × 8) |

④ Character scale (CharacterScale)

Constants of enumerated type used for character scale are shown in the following table.

| Constant Name | Description |
|---|---|
| **VERTICAL_1_HORIZONTAL_1** | Height × 1 and width × 1 |
| **VERTICAL_1_HORIZONTAL_2** | Height × 1 and width × 2 |
| **VERTICAL_1_HORIZONTAL_3** | Height × 1 and width × 3 |
| **VERTICAL_1_HORIZONTAL_4** | Height × 1 and width × 4 |
| **VERTICAL_2_HORIZONTAL_1** | Height × 2 and width × 1 |
| **VERTICAL_2_HORIZONTAL_2** | Height × 2 and width × 2 |
| **VERTICAL_2_HORIZONTAL_3** | Height × 2 and width × 3 |
| **VERTICAL_2_HORIZONTAL_4** | Height × 2 and width × 4 |
| **VERTICAL_3_HORIZONTAL_1** | Height × 3 and width × 1 |
| **VERTICAL_3_HORIZONTAL_2** | Height × 3 and width × 2 |
| **VERTICAL_3_HORIZONTAL_3** | Height × 3 and width × 3 |
| **VERTICAL_3_HORIZONTAL_4** | Height × 3 and width × 4 |
| **VERTICAL_4_HORIZONTAL_1** | Height × 4 and width × 1 |
| **VERTICAL_4_HORIZONTAL_2** | Height × 4 and width × 2 |
| **VERTICAL_4_HORIZONTAL_3** | Height × 4 and width × 3 |
| **VERTICAL_4_HORIZONTAL_4** | Height × 4 and width × 4 |

⑤ Alignment (PrintAlignment)

Constants of enumerated type used for alignment are shown in the following table.
Alignment cannot be added to the text data before inserting a new line feed.

| Constant Name | Description |
|---|---|
| **ALIGNMENT_LEFT** | Left aligned |
| **ALIGNMENT_CENTER** | Align center |
| **ALIGNMENT_RIGHT** | Right aligned |

⑥ Module size (ModuleSize)

Constants of enumerated type used for module size of QR Code are shown in the following table.

| Constant Name | Description | Method to Use |
|---|---|---|
| **QR_MODULE_SIZE_2** | 2 dots | **setTemplateQrCodeData** |
| **QR_MODULE_SIZE_3** | 3 dots | |
| **QR_MODULE_SIZE_4** | 4 dots | |
| **QR_MODULE_SIZE_5** | 5 dots | |
| **QR_MODULE_SIZE_6** | 6 dots | |
| **QR_MODULE_SIZE_7** | 7 dots | |
| **QR_MODULE_SIZE_8** | 8 dots | |
| **QR_MODULE_SIZE_9** | 9 dots | |
| **QR_MODULE_SIZE_10** | 10 dots | |
| **QR_MODULE_SIZE_11** | 11 dots | |

⑦ Error correction level (ErrorCorrection)

Constants of enumerated type used for error correction level are shown in the following table.

| Constant Name | Description | Method to Use |
|---|---|---|
| **QR_ERROR_CORRECTION_L** | Error correction level L | **setTemplateQrCodeData** |
| **QR_ERROR_CORRECTION_M** | Error correction level M | |
| **QR_ERROR_CORRECTION_H** | Error correction level H | |
| **QR_ERROR_CORRECTION_Q** | Error correction level Q | |

⑧ Memory area (MemoryArea)

Constants of enumerated type used for operating memory area are shown in the following table.

| Constant Name | Description |
|---|---|
| **MEMORY_DISPLAY_USERMEMORY** | User area |

⑨ Registered font (RegisteredFont)

Constants of enumerated type used for registered font are shown in the following table.

| Constant Name | Description |
|---|---|
| **FONT_STANDARD** | Standard font |
| **FONT_OPTION** | Optional font |

⑩ QR data mode (QrDataMode)

Constants of enumerated type used for QR data mode are shown in the following table.

| Constant Name | Description |
|---|---|
| **QRDATAMODE_NUMERIC** | Numeric mode |
| **QRDATAMODE_ALPHANUMERIC** | Alphanumeric mode |
| **QRDATAMODE_8BITBYTE** | 8-bit byte mode |
| **QRDATAMODE_KANJI** | Kanji mode |
| **QRDATAMODE_MIXTURE** | Mixed mode |

⑪ QR quiet zone (QrQuietZone)

Constants of enumerated type used for QR quiet zone are shown in the following table.

| Constant Name | Description |
|---|---|
| **QRQUIETZONE_EXIST** | Set QR quiet zone on |
| **QRQUIETZONE_NONE** | Set QR quiet zone off |

⑫ Macro registration processing (MacroRegistrationFunction)

Constants of enumerated type used for macro registration processing are shown in the following table.

| Constant Name | Description |
|---|---|
| **MACRO_REGISTRATION_CLEAR** | Cancel macro registration processing |
| **MACRO_REGISTRATION_START** | Start macro registration processing |
| **MACRO_REGISTRATION_REGIST** | Finish macro registration and macro registration processing |

(4) Method Details

## PrinterManager — Constructor

Constructor for **com.seikoinstruments.sdk.thermalprinter.PrinterManager** class.

Syntax public **PrinterManager**(Context *context*)

Parameter

| | | |
|---|---|---|
| *context* | Specify application context to call this method.<br>Example: **MainActivity.this** |

## connect — Start communicating with printer (Bluetooth)

Starts communication with a printer by Bluetooth connection.
In syntax (a), communication with the printer is always performed in a secure mode.
In syntax (b), communication with the printer is performed by specifying either a secure mode or an insecure mode.

Syntax (a) public void **connect**(int *printerModel*, String *address*) throws **PrinterException**

(b) public void **connect**(int *printerModel*, String *address*, boolean *secure*) throws **PrinterException**

Parameter

| | | |
|---|---|---|
| *printerModel* | Device model constant for Bluetooth connection<br>See "4.2.1(2)① Device model" for available constants. |
| *address* | Bluetooth address<br>Example: "00:11:22:AA:BB:CC" |
| *secure* | true Communicates with a printer in secure mode<br>false Communicates with a printer in insecure mode<br>Normally, communication in secure mode is recommended. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.

Description This method is valid only when Display is used via a printer.

Call this method before using other **PrinterManager** class methods.

The printer specified by *printerModel* is connected to the Bluetooth address specified by *address*.
Also, printer initial setting is performed at the connection based on the specified *printerModel*.

## connect — Start communicating with device (USB)

Starts communication with a device by USB connection.
In syntax (a), communication is performed with a device of the specified model.
In syntax (b), communication is performed with a device of the specified model and path.

Syntax (a) public void **connect**(int *printerModel*) throws **PrinterException**

(b) public void **connect**(int *printerModel,* String *address*) throws **PrinterException**

Parameter

| | |
|---|---|
| *printerModel* | Device model constant for USB connection<br>See "4.2.1(2)① Device model" for available constants. |

| | | |
|---|---|---|
| | *address* | Path of USB device file<br>Example: "/dev/bus/usb/001/002" |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.

Description Call this method before using other **PrinterManager** class methods.

The device specified by *printerModel* and/or *address* is connected.
Also, device initial setting is performed at the connection based on the specified *printerModel*.

## connect — Start communicating with printer (TCP/IP)

Starts communication with a printer by TCP/IP connection.

Syntax public void **connect**(int *printerModel*, String *address*) throws **PrinterException**

| | | |
|---|---|---|
| Parameter | *printerModel* | Device model constant for Ethernet connection<br>See "4.2.1(2)① Device model" for available constants. |
| | *address* | IP address<br>Example: "192.168.0.190" |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.

Description This method is valid only when Display is used via a printer.

Call this method before using other **PrinterManager** class methods.

Starts communication with a printer connected to the same network as the Android device by TCP/IP connection.
Connects to the IP address specified by *address*. TCP ports 9100 and 26100 are used for communication. Also, printer initial setting is performed at the connection based on the specified *printerModel*.

- **Creating/discarding of socket in TCP/IP connection of the library**

  After **connect**, the library retains the created socket until **disconnect**.
  And connecting to the same printer from other applications is not possible until **disconnect**.

  Based on the completion of data transmission to the printer, the socket is once discarded after elapsing the socket keeping time set by **setSocketKeepingTime**. Then the new socket is created immediately and used for the next connection.

## disconnect — Stop communicating with device

Stops communicating with the device.

Syntax public void **disconnect**() throws **PrinterException**

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.

Note It is recommended to get the execution response by **DISPLAY_RESPONSE_REQUEST** of **getDisplayResponse** before executing this method. If not, the communication is disconnected by this method before all data sending from Android device to the device is completed, and a part of the data may be lost.
When **getDisplayResponse** is not executed, evaluate that there is no problem in your program before using.

## sendBinary — Send binary data

Sends binary data to the device.

Syntax public void **sendBinary**(byte [] *binary*) throws **PrinterException**

Parameter *binary* Binary data to send to the device
Data size that can be specified at 1 time is 3 MB (3145728 bytes).

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.

When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description This method sends the specified binary data to the device without conversion.

By sending Display commands as binary data with this method, Display functions which are not supported in the library become available. However, this method does not support commands which get responses from Display.

## sendDataFile — Send specified file

Sends file data.

Syntax public void **sendDataFile**(String *fileName*) throws **PrinterException**

Parameter *fileName* Name of the data file to send to the device
The formats that can be entered are described below.

- Absolute path string handled by Java standard class "java.io.File"
  When targeting Android 10 (API 29) or later, please note that some files cannot be handled directly.
  See "3.5 Precautions - About Scoped Storage" for details.

- URI string of the following scheme name handled by the class "android.net.Uri" prepared for Android
  - file://
  - content://

  It is necessary to specify the URI string obtained from "Storage Access Framework" for this parameter. Please note that URI created without being obtained from "Storage Access Framework" may not be able to open the file.

The maximum file size that can be specified is 16 MB (16777216 bytes).
The file extensions that can be sent and the file transmission are described below.

- .bin, .dat
  Data is sent to the device as the binary data without conversion.

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

## startDiscoveryPrinter — Start printer search (Bluetooth)

Searches for the printer using the Bluetooth connection. The found printer information is stored in **PrinterInfo** class.

Syntax public void **startDiscoveryPrinter**(PrinterListener *listener*) throws **PrinterException**

Parameter

| | |
|---|---|
| *listener* | Instance of **PrinterListener**<br>Completion of this method or cancellation by **cancelDiscoveryPrinter** is notified to the user application as an end event by **finishEvent** through the instance set in *listener*. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.

Description This method may discover other printers besides SII printer.
In addition, the printers in which the Bluetooth connection is already established by the library or other applications are not found.

Do not call this method from the main thread of the application.

## startDiscoveryPrinter — Start printer search (USB)

Searches for the printer using the USB connection. The found printer information is stored in **PrinterInfo** class.

Syntax public void **startDiscoveryPrinter**(PrinterListener *listener,* int *deviceType*) throws **PrinterException**

Parameter

| | |
|---|---|
| *listener* | Instance of **PrinterListener**<br>Completion of this method or cancellation by **cancelDiscoveryPrinter** is notified to the user application as an end event by **finishEvent** through the instance set in *listener*. |
| *deviceType* | Port type<br>Specify **PRINTER_TYPE_USB**. |

Error **PrinterException** is thrown when an error occurs while calling this method.

Description This method searches for SII device. The device information of the found device is stored to **PrinterInfo** class described later.

## startDiscoveryPrinter — Start printer search (TCP/IP)

Searches for SII printer.

Syntax: public void **startDiscoveryPrinter**(PrinterListener *listener*, int *retry*, int *timeout*) throws **PrinterException**

Parameter

| | |
|---|---|
| *listener* | Instance of **PrinterListener**<br>Completion of this method or cancellation by **cancelDiscoveryPrinter** is notified to the user application as an end event by **finishEvent** through the instance set in *listener*. |
| *retry* | Number of retry (times)<br>Sends the local broadcast packet the number of times set by *retry*.<br>The valid range is 1 to 5.<br>When the value is specified less than 1, the number is set to 1.<br>When the value is specified more than 5, the number is set to 5. |
| *timeout* | Search timeout period (millisecond: ms)<br>Sets the timeout period per search. Each time the local broadcast packet is sent, this method waits for a response from the printer until the period specified by *timeout* elapses.<br>The valid range is 3000 to 60000.<br>When the value is specified less than 3000, the period is set to 3000 ms.<br>When the value is specified more than 60000, the period is set to 60000 ms. |

Error: **PrinterException** is thrown when an error occurs while calling this method.

Description: This method searches for SII printer. The printer information of the found printer is stored to **PrinterInfo** class described later.

## startDiscoveryDevice — Start device search (USB)

Searches for SII device.

Syntax: public void **startDiscoveryDevice**(PrinterListener *listener,* int *deviceType*) throws **PrinterException**

Parameter

| | |
|---|---|
| *listener* | Instance of **PrinterListener**<br>Completion of this method or cancellation by **cancelDiscoveryDevice** is notified to the user application as an end event by **finishEvent** through the instance set in *listener*. |
| *deviceType* | Port type<br>Specify **PRINTER_TYPE_USB**. |

Error: **PrinterException** is thrown when an error occurs while calling this method.

Description: This method searches for SII device. The device information of the found device is stored to **PrinterInfo** class described later.

## cancelDiscoveryPrinter — Cancel printer search

Cancels **startDiscoveryPrinter** under execution.

Syntax public void **cancelDiscoveryPrinter**()

Description Cancellation by this method is notified as an end event to the user application by **finishEvent** through the instance set in *listener* of **startDiscoveryPrinter**.

## cancelDiscoveryDevice — Cancel device search

Cancels **startDiscoveryDevice** under execution.

Syntax public void **cancelDiscoveryDevice**()

Description Cancellation by this method is notified as an end event to the user application by **finishEvent** through the instance set in *listener* of **startDiscoveryDevice**.

## getFoundPrinter — Get found printer information

Gets the information of the printer found by **startDiscoveryPrinter** in ArrayList from the **PrinterInfo** class, which is the storage destination.

Syntax public ArrayList<**PrinterInfo**> **getFoundPrinter**()

Return value ArrayList of **PrinterInfo** class

## getFoundDevice — Get found device information

Gets the information of the device found by **startDiscoveryDevice** in ArrayList from the **PrinterInfo** class, which is the storage destination.

Syntax public ArrayList<**PrinterInfo**> **getFoundDevice**()

Return value ArrayList of **PrinterInfo** class

## getSendTimeout — Get send timeout period

Gets the send timeout period.

Syntax public int **getSendTimeout**()

Return value Send timeout period (millisecond: ms)

Description This method can get the send timeout period regardless of whether **isConnect** is true or false.

## setSendTimeout — Set send timeout period

Sets the send timeout period.

Syntax: public void **setSendTimeout**(int *sendTimeout*)

Parameter:

| | | |
|---|---|---|
| *sendTimeout* | Send timeout period (millisecond: ms)<br>The valid range is 100 to 90000.<br>The value is set to 10000 ms when the value out of the valid range is specified. |

Description: When the send timeout period is not set by this method, the value is set to 10000.

This method can set the send timeout period regardless of whether **isConnect** is true or false.

The set timeout period becomes effective at the next data sending.

## getReceiveTimeout — Get receive timeout period

Gets the receive timeout period.

Syntax: public int **getReceiveTimeout**()

Return value: Receive timeout period (millisecond: ms)

Description: This method can get the receive timeout period regardless of whether **isConnect** is true or false.

## setReceiveTimeout — Set receive timeout period

Sets the receive timeout period.

Syntax: public void **setReceiveTimeout**(int *receiveTimeout*)

Parameter:

| | | |
|---|---|---|
| *receiveTimeout* | Receive timeout period (millisecond: ms)<br>The valid range is 100 to 90000.<br>The value is set to 10000 ms when the value out of the valid range is specified. |

Description: When the receive timeout period is not set by this method, the value is set to 10000.

This method can set the receive timeout period regardless of whether **isConnect** is true or false.

The set timeout period becomes effective at the next data receiving.

## getInternationalCharacter — Get international character set

Gets the value of international character set.

Syntax: public int **getInternationalCharacter**()

Return value: See "4.2.1(2)② International character set" for details of the value.

Description When the text data is sent by **sendDataFile** or **setTemplateTextData**, the display result for the following character codes varies. See "Appendix A Character Set" for details about characters to be displayed.

Character codes whose display result varies depending on the international character set configuration
0x23, 0x24, 0x40, 0x5B, 0x5C, 0x5D, 0x5E, 0x60, 0x7B, 0x7C, 0x7D, 0x7E

## setInternationalCharacter — Set international character set

Sets the value of international character set.

Syntax public void **setInternationalCharacter**(int *internationalCharacter*)

Parameter

| | |
|---|---|
| *internationalCharacter* | International character set constant<br>See "4.2.1(2)② International character set" for available constants.<br>When an invalid value is specified, it is ignored. |

Description When the international character set is not set by this method, it is as follows depending on the language setting of an Android device.
When the language setting of the Android device is Japanese:
**COUNTRY_JAPAN**
When the language setting of the Android device is other than Japanese:
**COUNTRY_USA**

## getCodePage — Get codepage

Gets the value of codepage.

Syntax public int **getCodePage**()

Return value See "4.2.1(2)③ Codepage" for details of the value.

## setCodePage — Set codepage

Sets the value of codepage.

Syntax public void **setCodePage**(int *codePage*)

Parameter

| | |
|---|---|
| *codePage* | Codepage constant<br>See "4.2.1(2)③ Codepage" for available constants.<br>When an invalid value is specified, it is ignored. |

Description When the codepage is not set by this method, it is as follows depending on the language setting of an Android device.
When the language setting of the Android device is Japanese:
**CODE_PAGE_KATAKANA**
When the language setting of the Android device is other than Japanese:
**CODE_PAGE_1252**

When the text data is sent by **sendDataFile** or **setTemplateTextData**, an encoding to be used is changed depending on this property setting.

## getPrinterModel — Get device model

Gets the value of the connecting device model.

Syntax public int **getPrinterModel**()

Return value See "4.2.1(2)① Device model" for details of the value.
**PRINTER_MODEL_DEFAULT** is returned when **isConnect** is false.

Description Even when the device is not connected, when **connect** has succeeded once, the device model value successfully connected last time is returned.

## getPortType — Get connecting port type

Gets the port type used for connecting with the device.

Syntax public int **getPortType**()

Return value See "4.2.1(2)④ Port type" for details of the value.
**PRINTER_TYPE_BLUETOOTH** is returned when **isConnect** is false.

Description Even when the device is not connected, when **connect** has been succeeded once, the port type value successfully connected last time is returned.

## isConnect — Verify connection state with device

Verifies connection state with the device.

Syntax public boolean **isConnect**()

Return value true: Connected to a device
false: Not connected to a device

Description When the data transmission is failed, the communication with the device is ended, and this method returns false. When false is returned, reconnect with the printer by **connect**.

## getSocketKeepingTime — Get socket keeping time

Gets the socket keeping time.

Syntax public int **getSocketKeepingTime**()

Return value Socket keeping time (millisecond: ms)

Description This method is valid only when Display is used via a printer.
This method can get the socket keeping time regardless of whether **isConnect** is true or false.

## setSocketKeepingTime — Set socket keeping time

Sets the socket keeping time.

Syntax public void **setSocketKeepingTime**(int *socketKeepingTime*)

Valid range 60000 to 300000 (millisecond: ms)
When the value is specified less than 60000, the time is set to 60000 ms.
When the value is specified more than 300000, the time is set to 300000 ms.

Default 300000

Description This method is valid only when Display is used via a printer.
This method can set the socket keeping time regardless of whether **isConnect** is true or false.

For the socket keeping time, specify a time equal to Receive Timeout of the printer to be connected. See the printer command "Set Wireless LAN Communication" described in "Technical Reference for Printer" for details of Receive Timeout.

The set socket keeping time becomes effective at the next **connect** execution.

## getVersion — Get SDK version

Gets the SDK version as a character string.

Syntax public String **getVersion**()

Return value SDK version character string (Example: When the SDK version is Ver.1.0.0, the return value is "1.0.0")

Description This method is valid only when Display is used via a printer.
This method can get the SDK version regardless of whether **isConnect** is true or false.

## defragment — Optimize memory area

Optimizes the memory area.

Syntax public void **defragment**(MemoryArea *memoryArea*) throws **PrinterException**

Parameter *memoryArea* Memory area
See "4.2.1(3)⑧ Memory area (MemoryArea)" for available constants.

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.
It may take several minutes for optimization.
Do not turn the device power off during optimization.

In use via a printer, this method is ignored when Display is not connected to the printer.

## initializeMemoryArea — Initialize memory area

Initializes the memory area.

Syntax public void **initializeMemoryArea**(MemoryArea *memoryArea*) throws **PrinterException**

Parameter *memoryArea* Memory area
See "4.2.1(3)⑧ Memory area (MemoryArea)" for available constants.

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.
It may take several minutes for initialization.
Do not turn the device power off during initialization.

In use via a printer, this method is ignored when Display is not connected to the printer.

**Note** Registered data in following methods is deleted when the memory area is initialized after specifying **MEMORY_DISPLAY_USERMEMORY**.
- **registerTemplate**
- **registerImageData**
- **controlMacroRegistration**
- **registerUserDefinedCharacter**
- **registerOptionFont**

**In addition, part of data which is registered at the shipping to use for the system is deleted either.**
**Therefore, the guide screen cannot be displayed when an error occurs in the printer.**
The used memory area can be reused after executing **defragment**.

## showTemplate Display template

Displays the template on the screen.

Syntax public void **showTemplate**(int *time_ms*) throws **PrinterException**

Parameter

| | |
|---|---|
| *time_ms* | Display time (ms: millisecond)<br>Specify display time on the screen with *time_ms* (ms).<br>The valid range is 0 to 25500.<br>When the value exceeds 0 and less than 100 is specified, the time is set to 100 ms.<br>When the value exceeding 25500 is specified, the time is set to 25500 ms.<br><br>For macro registration:<br>When 0 is specified with *time_ms*, the template is shown continuously.<br>When other than 0 is specified with *time_ms*, a next template is shown after the display time is elapsed.<br><br>For other than macro registration:<br>When 0 is specified with *time_ms*, the template is shown continuously.<br>When other than 0 is specified with *time_ms*, the template returns to a previous template after the display time is elapsed.<br>In case of the previous screen has been updated with the display time other than 0, the screen is traced back to the template which was updated with the display time 0. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Updates the screen, and displays data being specified with the following methods.
・**selectTemplate**
・**setTemplateImageData**
・**selectTemplateTextObject**
・**setTemplateTextAlignment**
・**setTemplateTextLeftMargin**
・**setTemplateTextLineSpacing**
・**setTemplateTextBold**
・**setTemplateTextUnderline**
・**setTemplateTextSize**
・**setTemplateTextFont**
・**setTemplateTextRightSpacing**
・**setTemplateTextColor**
・**setTemplateTextData**
・**setTemplateBarcodeData**
・**setTemplateQrCodeData**

In use via a printer, this method is ignored when Display is not connected to the printer.

## showSlide Display slide

Displays the slide on the screen.

Syntax public void **showSlide**(int *slideID*, int *time_ms*) throws **PrinterException**

Parameter

| | |
|---|---|
| *slideID* | Slide ID<br>Specify the ID of slide data which was registered at **registerSlideData**.<br>The valid range is 0 to 91.<br>This method is ignored when slide data is not registered in the specified ID. |
| *time_ms* | Display time (ms: millisecond)<br>Specify display time on the screen with *time_ms* (ms).<br>The valid range is 0 to 25500.<br>When the value exceeds 0 and less than 100 is specified, the time is set to 100 ms.<br>When the value exceeding 25500 is specified, the time is set to 25500 ms.<br><br>For macro registration:<br>When 0 is specified with *time_ms*, the slide is shown continuously.<br>When other than 0 is specified with *time_ms*, a next slide is shown after the display time is elapsed.<br><br>For other than macro registration:<br>When 0 is specified with *time_ms*, the slide is shown continuously.<br>When other than 0 is specified with *time_ms*, the slide returns to a previous slide after the display time is elapsed.<br>In case of the previous screen has been updated with the display time other than 0, the screen is traced back to the slide which was updated with the display time 0. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description A selecting template is deselected.
In use via a printer, this method is ignored when Display is not connected to the printer.

## enterStandbyMode — Display standby

Changes Display to Standby mode.

Syntax public void **enterStandbyMode**() throws **PrinterException**

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description This method is ignored during Standby mode or Guide mode is being displayed.
A selecting template is deselected.

In use via a printer, this method is ignored when Display is not connected to the printer.

## executeMacro — Execute macro

Executes the macro.

Syntax public void **executeMacro**(int *macroID*, int *repeatCount*) throws **PrinterException**

Parameter

| | |
|---|---|
| *macroID* | Macro ID<br>Specify the macro ID which was registered at **controlMacroRegistration**.<br>The valid range is 0 to 127.<br>This method is ignored when the macro is not registered in the specified ID. |
| *repeatCount* | The number of execution times<br>Specify times to execute the macro.<br>The valid range is 0 to 255.<br>Continues the repeating when 0 is specified. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description A selecting template is deselected.

In use via a printer, this method is ignored when Display is not connected to the printer.

## turnOnScreen — Turn on/off screen

Sets the screen backlight on/off.

Syntax public void **turnOnScreen**(boolean *isOn*) throws **PrinterException**

Parameter

| | | |
|---|---|---|
| *isOn* | Screen status<br>Specify the screen status from following.<br>true : backlight on<br>false : backlight off | |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.

Description In use via a printer, this method is ignored when Display is not connected to the printer.

## selectTemplate — Select template

Selects the template to show on Display.
In syntax (a), slide data for use in the template or as the template background can be selected.
In syntax (b), a template can be selected.

Syntax (a) public void **selectTemplate**(int *templateID,* int *SlideID*) throws **PrinterException**

(b) public void **selectTemplate**(int *templateID*) throws **PrinterException**

Parameter

| | |
|---|---|
| *templateID* | Template ID<br>Specify the ID of template to select.<br>The valid range is 0 to 127.<br>This method is ignored when the template is not registered in the specified ID. |
| *slideID* | Slide ID<br>Specify the ID of slide data to use for the background of the template.<br>The valid range is 0 to 91.<br>This method is ignored when slide data is not registered in the specified ID. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description The data on the template is cleared when this method is executed.
The selected template is displayed to the screen when **showTemplate** is executed.

The selecting template is deselected when **showSlide**, **enterStandbyMode**, or **executeMacro** is executed.
The selecting template is deselected when the specified time of display is executed at **showTemplate**.

Use following templates depends on the values of **codePage** when characters other than 20h to 7Eh of ASCII character are input with **setTemplateTextData**.
- When **codePage** is **CODE_PAGE_KATAKANA**:
  Use the template which encoding specifying is Shift_JIS.
- When **codePage** is other than **CODE_PAGE_KATAKANA**:
  Use the template which encoding specifying is ISO-2022-JP.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateImageData — Set image data

Sets image data to show on a selecting template.

Syntax public void **setTemplateImageData**(int *mapID*, int *imageID*) throws **PrinterException**

Parameter

| | |
|---|---|
| *mapID* | Map ID<br>The valid range is 0 to 63.<br>This method is ignored when the specified map ID is not defined in the template. |
| *imageID* | Image ID<br>Specify the ID of image data which was registered at **registerImageData**.<br>The valid range is 0 to 63.<br>This method is ignored when image data is not registered in the specified ID. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description After specifying the map ID of the selecting template with this method, specify the image ID to map.
The specified image data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
- When **selectTemplate** is executed.
- When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
- When **showtTemplate** registered in **executeMacro** is executed.

This method is ignored when a template is not selected.
In use via a printer, this method is ignored when Display is not connected to the printer.

## selectTemplateTextObject — Select text element

Selects the text element to start editing.

Syntax public void **selectTemplateTextObject**(int *mapID*) throws **PrinterException**

Parameter

| | |
|---|---|
| *mapID* | Map ID<br>The valid range is 0 to 63.<br>This method is ignored when the specified map ID is not defined in the template. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description After specifying the map ID of the selecting template with this method, start to edit characters.

When a scroll is set with the text element of the specified map ID and this method is executed after **showTemplate**, the scroll is executed.
When a scroll is not set with the text element of the specified map ID and this method is executed after **showTemplate**, input text data is discarded.

This method setting is cleared under the following conditions.
- When **selectTemplate** is executed.
- When **showtTemplate** is executed.

This method is ignored when a template is not selected.
In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextAlignment — Alignment of text data

Sets alignment to text data shown on Display.

Syntax public void **setTemplateTextAlignment**(PrintAlignment *alignment*) throws **PrinterException**

Parameter

| | | |
|---|---|---|
| *alignment* | Alignment<br>See "4.2.1(3)⑤ Alignment (PrintAlignment)" for available constants. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Alignment is valid only as following cases.
- Text data is not entered in the specified map ID at **selectTemplateTextObject**.
- Text data entered in the map ID which is specified at **selectTemplateTextObject** is registered right after a line feed.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
- When **selectTemplate** is executed.
- When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
- When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextLeftMargin — Set left margin of text data

Sets left margin to text data shown on Display.

Syntax public void **setTemplateTextLeftAlignment**(int *margin*) throws **PrinterException**

Parameter

| | |
|---|---|
| *margin* | Left margin (pixel: px)<br>The valid range is 0 to 479. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description The left margin setting is valid only as following cases.
- Text data is not entered in the specified map ID at **selectTemplateTextObject**.
- Text data entered in the map ID which is specified at **selectTemplateTextObject** is registered right after a line feed.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
- When **selectTemplate** is executed.
- When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
- When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextLineSpacing — Set line spacing of text data

Sets line spacing to text data shown on Display.

Syntax public void **setTemplateTextLineSpacing**(int *spacing*) throws **PrinterException**

Parameter

| | | |
|---|---|---|
| *spacing* | Line spacing (pixel: px)<br>The valid range is 0 to 255. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description The line spacing setting is valid only as following cases.
- Text data is not entered in the specified map ID at **selectTemplateTextObject**.
- Text data entered in the map ID which is specified at **selectTemplateTextObject** is registered right after a line feed.

When a scroll is set to a text element of the specified map ID, the line spacing to text data is not reflected.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
- When **selectTemplate** is executed.
- When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
- When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextBold — Set bold printing of text data

Sets bold printings to text data shown on Display.

Syntax public void **setTemplateTextBold**(CharacterBold *bold*) throws **PrinterException**

Parameter

| | |
|---|---|
| *bold* | Bold printing<br>See "4.2.1(3)① Bold printing (CharacterBold)" for available constants. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description From the text data after this method is executed, the bold printings are applied.
The bold printing can be set one by one.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
- When **selectTemplate** is executed.
- When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
- When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextUnderline — Set underline of text data

Sets underline to text data shown on Display.

Syntax public void **setTemplateTextUnderline**(CharacterUnderline *underline*) throws **PrinterException**

Parameter

| | |
|---|---|
| *underline* | Underline<br>See "4.2.1(3)② Underline (CharacterUnderline)" for available constants. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description From the text data after this method is executed, the underlines are applied.
The underline can be set one by one.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
・When **selectTemplate** is executed.
・When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
・When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextSize — Set character size of text data

Sets character size to text data shown on Display.

Syntax public void **setTemplateTextSize**(CharacterScale *scale*) throws **PrinterException**

Parameter

| | | |
|---|---|---|
| *scale* | Character scale<br>See "4.2.1(3)④ Character scale (CharacterScale)" for available constants. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description From the text data after this method is executed, the character sizes are applied.
The character size can be set one by one.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
・When **selectTemplate** is executed.
・When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
・When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextFont — Set character font of text data

Sets a character font to text data shown on Display.

Syntax public void **setTemplateTextFont**(CharacterFont *font*) throws **PrinterException**

Parameter

| | | |
|---|---|---|
| *font* | Character font<br>See "4.2.1(3)③ Character font (CharacterFont)" for available constants. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description From the text data after this method is executed, the character fonts are applied.
The character font can be set one by one.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
・When **selectTemplate** is executed.
・When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
・When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextRegisteredFont — Set registered font of text data

Sets the registered font used for text data to show on Display.

Syntax public void **setTemplateRegisteredFont**(RegisteredFont *font*) throws **PrinterException**

Parameter

| | | |
|---|---|---|
| *font* | Registered font<br>See "4.2.1(3)⑨ Registered font (RegisteredFont)" for available constants.<br>The registered font specifying is ignored when the optional font is not registered. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description From the text data after this method is executed, the registered fonts are applied.
The registered font can be set one by one.

Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
・When **selectTemplate** is executed.
・When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
・When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextRightSpacing — Set right space of text data

Sets the amount of right space to text data shown on Display.

Syntax public void **setTemplateTextRightSpacing**(int *spacing*) throws **PrinterException**

Parameter

| | |
|---|---|
| *space* | The amount of character right space (pixel: px)<br>The valid range is 0 to 255. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description From the text data after this method is executed, the amount of character right space is applied.
The amount of right space can be set one by one.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
・When **selectTemplate** is executed.
・When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
・When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextColor — Set character color of text data

Sets the character color used for text data to show on Display.

Syntax public void **setTemplateTextColor**(int *color*) throws **PrinterException**

Parameter

| | | |
|---|---|---|
| *color* | Character color<br>The valid range is 0 to 0xffffff. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description This method can set character colors to text data.
The character colors can be set in RGB24 bit color. The set color is displayed in the color converted to 16 bit RGB555.

From the text data after this method is executed, the character colors are applied.
The character color can be set one by one.

Specify the map ID at **selectTemplateTextObject** before executing this method.
Input text data at **setTemplateTextData** after executing this method.
The input text data is displayed to the screen when **showTemplate** is executed.

This method setting is cleared under the following conditions.
・When **selectTemplate** is executed.
・When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
・When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateTextData — Input text data

Inputs text data to show on Display.

Syntax public void **setTemplateTextData**(String *text*) throws **PrinterException**

Parameter

| | |
|---|---|
| *text* | Text data to show on Display<br>Data size which is able to be specified it once is 1 to 1020 bytes. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description This method encodes input text data into text data which is enable to display on the basis of settings at **internationalCharacter** and **codePage**, and displays.

Select the template in **selectTemplate** before executing this method.
After specifying the map ID of the selecting template with **selectTemplateTextObject**, input text data with this method.
This method is ignored when the map ID is not specified in **selectTemplateTextObject**.
The input text data is displayed to the screen when **showTemplate** is executed.

The input text data is cleared under the following conditions.
- When **selectTemplate** is executed.
- When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
- When **showtTemplate** registered in **executeMacro** is executed.

In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateBarcodeData — Input barcode data

Specifies a map ID of the barcode element on a selecting template, and inputs barcode data.
In syntax (a), data consisting of character strings for displaying a barcode can be input.
In syntax (b), data consisting of byte arrays for displaying a barcode can be input.

Syntax (a) public void **setTemplateBarcodeData**(int *mapID,* String *text*) throws **PrinterException**

(b) public void **setTemplateBarcodeData**(int *mapID,* byte [] *data*) throws **PrinterException**

Parameter

| | |
|---|---|
| *mapID* | Map ID<br>The valid range is 0 to 7.<br>This method is ignored when the specified map ID is not defined in the selecting template. |
| *text* | Barcode data<br>The characters that can be input are ASCII characters 20h (space) to 7Eh (tilde) such as alphanumeric characters ('0' to '9', 'A' to 'Z', 'a' to 'z').<br>The available number of characters is 1 to 150.<br>Barcode data which is not complying with barcode specification is ignored. |

*data* Barcode data
The value that can be input is 00h to 7Fh.
The available number of data is 1 to 150.
Barcode data which is not complied with barcode specification is ignored.

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description After specifying the map ID of the selecting template with this method, input barcode data.
The input barcode data is displayed to the screen when **showTemplate** is executed.

The input barcode data is cleared under the following conditions.
- When **selectTemplate** is executed.
- When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
- When **showtTemplate** registered in **executeMacro** is executed.

This method is ignored when a template is not selected.
In use via a printer, this method is ignored when Display is not connected to the printer.

## setTemplateQrCodeData — Input QR Code data

Specifies a map ID of the qr element on a selecting template, and inputs QR Code data.
In syntax (a), QR Code data can be input as a character string.
In syntax (b), QR Code data can be input as a character string. The setting of selecting template is reflected to *moduleSize*, *errorCorrection*, *mode*, and *qrQuietZone*.
In syntax (c), QR Code data can be input as a byte array.
In syntax (d), QR Code data can be input as a byte array. The setting of selecting template is reflected to *moduleSize*, *errorCorrection*, *mode*, and *qrQuietZone*.

Syntax (a) public void **setTemplateQrCodeData**(int *mapID,*
ModuleSize *moduleSize,*
ErrorCorrection *errorCorrection,*
QrDataMode *mode,*
QrQuietZone *qrQuietZone,*
String *text*) throws **PrinterException**

(b) public void **setTemplateQrCodeData**(int *mapID,* String *text*) throws **PrinterException**

(c) public void **setTemplateQrCodeData**(int *mapID,*
ModuleSize *moduleSize,*
ErrorCorrection *errorCorrection,*
QrDataMode *mode,*
QrQuietZone *qrQuietZone,*
byte [] *data*) throws **PrinterException**

(d) public void **setTemplateQrCodeData**(int *mapID,* byte [] *data*) throws **PrinterException**

Parameter *mapID* Map ID
The valid range is 0 to 7.
This method is ignored when the specified map ID is not defined in the selecting template.

*moduleSize* Module size
See "4.2.1(3)⑥ Module size (ModuleSize)" for available constants.

| | | |
|---|---|---|
| | *errorCorrection* | Error correction level<br>See "4.2.1(3)⑦ Error correction level (ErrorCorrection)" for available constants. |
| | *mode* | Data mode<br>See "4.2.1(3)⑩ QR data mode (QrDataMode)" for available constants. |
| | *qrQuietZone* | Quiet zone<br>See "4.2.1(3)⑪ QR quiet zone (QrQuietZone)" for available constants. |
| | *text* | QR Code data<br>The characters that can be input are as follows:<br>・ASCII code characters 20h (space) to 7Eh (tilde) such as alphanumeric characters ('0' to '9', 'A' to 'Z', 'a' to 'z').<br>・8 bits Latin / Katakana based on JIS X 0201<br>・Shift-JIS code based on JIS X 0208<br>The available data size is 1 to 3909 bytes.<br>QR Code data which is not complied with QR Code specification is ignored. |
| | *data* | QR Code data<br>QR Code data shown on Display.<br>The value that can be input is 00h to FFh.<br>The available number of data is 1 to 3909.<br>QR Code data which is not complied with QR Code specification is ignored. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description After specifying the map ID of the selecting template with this method, input QR Code data.
The input QR Code data is displayed to the screen when **showTemplate** is executed.

The input QR Code data is cleared under the following conditions.
- ・When **selectTemplate** is executed.
- ・When other than 0 is specified at *time_ms* of **showTemplate**, and the specified display time has elapsed.
- ・When **showtTemplate** registered in **executeMacro** is executed.

This method is ignored when a template is not selected.
In use via a printer, this method is ignored when Display is not connected to the printer.

## registerTemplate — Register template

Registers a template.

Syntax public void **registerTemplate**(int *templateID,*
String *label,*
String *filepath*) throws **PrinterException**

| | | |
|---|---|---|
| Parameter | *templateID* | Template ID<br>Specify an ID of template to register.<br>The valid range is 0 to 127.<br>Do not specify the template ID of 127 because it is being used for the system. |

| | | |
|---|---|---|
| | *label* | Template name<br>A name for identification can be specified to the template to be registered.<br>The characters that can be specified are ASCII code characters 20h (space) to 7Eh (tilde) such as alphanumeric characters ('0' to '9', 'A' to 'Z', 'a' to 'z').<br>Do not use Unicode 00A5h ('¥').<br>The available number of characters is 0 to 32.<br>This *label* is optional. Specify null when the template name is not registered.<br>The specified template name can be retrieved with **getDisplayResponse**. |
| | *filePath* | File path of the template data to register in Display<br>The formats that can be entered are described below.<br>• Absolute path string handled by Java standard class "java.io.File"<br>When targeting Android 10 (API 29) or later, please note that some files cannot be handled directly.<br>See "3.5 Precautions - About Scoped Storage" for details.<br>• URI string of the following scheme name handled by the class "android.net.Uri" prepared for Android<br>・file://<br>・content://<br>It is necessary to specify the URI string obtained from "Storage Access Framework" for this parameter. Please note that URI created without being obtained from "Storage Access Framework" may not be able to open the file.<br>Supported file extension is .xml.<br>The maximum data size that can be registered is 8192 bytes.<br>See "Technical Reference for Display" for details on registration of the template data. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.

In use via a printer, this method is ignored when Display is not connected to the printer.

**Note** **Registered data at the shipping may be added or changed without prior notice for quality improvement.**

## unregisterTemplate — Delete template

Deletes the registered template.

Syntax public void **unregisterTemplate**(int *templateID*) throws **PrinterException**

| | | |
|---|---|---|
| Parameter | *templateID* | Template ID<br>Specify an ID of template to delete.<br>The valid range is 0 to 127.<br>This method is ignored when a template is not registered in the specified ID. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.

Used memory is not released even the template is deleted. The used memory can be reused after executing **defragment**.

In use via a printer, this method is ignored when Display is not connected to the printer.

## registerImageData — Register image data

Registers image data.

Syntax public void **registerImageData**(int *imageID,*
String *label,*
String *filepath*) throws **PrinterException**

Parameter

*imageID* — Image ID
The valid range is 0 to 63.
Do not specify the image IDs of 49 to 63 because they are being used for the system.

*label* — Image name
A name for identification can be specified to image data to be registered.
The characters that can be specified are ASCII characters 20h (space) to 7Eh (tilde) such as alphanumeric characters ('0' to '9', 'A' to 'Z', 'a' to 'z').
Do not use Unicode 00A5h ('¥').
The available number of characters is 0 to 32.
This *label* is optional. Specify null when the template name is not registered.
The specified image name can be retrieved with **getDisplayResponse**.

*filePath* — File path
The formats that can be entered are described below.

- Absolute path string handled by Java standard class "java.io.File"
  When targeting Android 10 (API 29) or later, please note that some files cannot be handled directly.
  See "3.5 Precautions - About Scoped Storage" for details.

- URI string of the following scheme name handled by the class "android.net.Uri" prepared for Android
  - file://
  - content://

  It is necessary to specify the URI string obtained from "Storage Access Framework" for this parameter. Please note that URI created without being obtained from "Storage Access Framework" may not be able to open the file.

Specify the file name of image data to register.
Supported file extensions are .jpg, .jpeg, and .png.
However, even the supported extensions may not be registered depending on the format.
The maximum file size that can be registered is 786432 bytes.
The maximum data size that can be registered is 480 horizontal × 272 vertical pixels (px).

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.

In use via a printer, this method is ignored when Display is not connected to the printer.

**Note** **Registered data at the shipping may be added or changed without prior notice for quality improvement.**

## unregisterImageData — Delete image data

Deletes registered image data.

Syntax public void **unregisterImageData**(int *imageID*) throws **PrinterException**

Parameter *imageID* Image ID
The valid range is 0 to 63.
This method is ignored when image data is not registered in the specified ID.

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.

Used memory is not released even image data is deleted. The used memory can be reused after executing **defragment**.

In use via a printer, this method is ignored when Display is not connected to the printer.

## registerSlideData — Register slide data

Registers slide data.

Syntax

public void **registerSlideData**(int *slideID,*
String *label,*
String *filePath*) throws **PrinterException**

Parameter

*slideID*

Slide ID
The valid range is 0 to 91.
Do not specify the slide IDs of 80 to 90 because they are being used for the system.

*label*

Slide name
A name for identification can be specified to slide data to be registered.
The characters that can be specified are ASCII characters 20h (space) to 7Eh (tilde) such as alphanumeric characters ('0' to '9', 'A' to 'Z', 'a' to 'z').
Do not use Unicode 00A5h ('¥').
The available number of characters is 0 to 32.
This *label* is optional. Specify null when the template name is not registered.
The specified slide name can be retrieved with **getDisplayResponse**.

*filePath*

File path
The formats that can be entered are described below.

- Absolute path string handled by Java standard class "java.io.File"
  When targeting Android 10 (API 29) or later, please note that some files cannot be handled directly.
  See "3.5 Precautions - About Scoped Storage" for details.

- URI string of the following scheme name handled by the class "android.net.Uri" prepared for Android
  - file://
  - content://

  It is necessary to specify the URI string obtained from "Storage Access Framework" for this parameter. Please note that URI created without being obtained from "Storage Access Framework" may not be able to open the file.

Specify the file name of slide data to register.
Supported file extensions are .jpg, .jpeg, and .png.
However, even the supported extensions may not be registered depending on the format.
The maximum file size that can be registered is 786432 bytes.
The data size that can be registered is fixed to 480 horizontal × 272 vertical pixels (px).

Exception

**PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description

Display is changed to Standby mode when this method is executed. A selecting template is deselected.

Execute **showSlide** to show the registered slide data.
Specify the slide ID at **showTemplate** to use the registered slide data as a backscreen of the template.

In use via a printer, this method is ignored when Display is not connected to the printer.

**Note** **Registered data at the shipping may be added or changed without prior notice for quality improvement.**

## unregisterSlideData — Delete slide data

Deletes registered slide data.

Syntax public void **unregisterSlideData**(int *imageID*) throws **PrinterException**

Parameter

| | |
|---|---|
| *slideID* | Slide ID<br>The valid range is 0 to 91.<br>This method is ignored when slide data is not registered in the specified ID. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.

In use via a printer, this method is ignored when Display is not connected to the printer.

## registerUserDefinedCharacter — Register user-defined character

Registers user-defined characters in Display.

Syntax public void **registerUserDefinedCharacter**(String *filePath*) throws **PrinterException**

Parameter

| | |
|---|---|
| *filePath* | File path<br>The formats that can be entered are described below.<br>• Absolute path string handled by Java standard class "java.io.File"<br>When targeting Android 10 (API 29) or later, please note that some files cannot be handled directly.<br>See "3.5 Precautions - About Scoped Storage" for details.<br>• URI string of the following scheme name handled by the class "android.net.Uri" prepared for Android<br>・file://<br>・content://<br>It is necessary to specify the URI string obtained from "Storage Access Framework" for this parameter. Please note that URI created without being obtained from "Storage Access Framework" may not be able to open the file. |

Specify the file name of the user-defined characters to register.
Supported file extension is .bin.
See "Register User-Defined Character" of the Display command in "Technical Reference for Display" for details on the user-defined character data.

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.
User-defined characters are overwritten when this method is executed with user-defined character registered status.

Use the template which encoding specifying is Shift_JIS for displaying user-defined characters.
Specify **CODE_PAGE_KATAKANA** for *codePage* of **setCodePage** before user-defined characters are displayed.
Specify the character codes that can be specified for *text* of **setTemplateTextData** when user-defined characters are displayed.
The character codes that can be specified are E000h to E05Dh.

In use via a printer, this method is ignored when Display is not connected to the printer.

## unregisterUserDefinedCharacter — Delete user-defined character

Deletes registered user-defined characters.

Syntax public void **unregisterUserDefinedCharacter**() throws **PrinterException**

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.

All registered user-defined characters are deleted.
Used memory is not released even the user-defined characters are deleted. The used memory can be reused after executing **defragment**.

In use via a printer, this method is ignored when Display is not connected to the printer.

## registerOptionFont — Register optional font

Registers optional fonts in Display.

Syntax

public void **registerOptionFont**(int *startCode,*
int *endCode,*
int *width,*
int *height,*
String *filePath*) throws **PrinterException**

Parameter

*startCode* — Character code for registration starting
The valid range is 20h to FFh of ASCII character code.

*endCode* — Character code for registration finishing
The valid range is 20h to FFh of ASCII character code.

*width* — Character width (pixel: px)
The valid range is 1 to 255.

*height* — Character height (pixel: px)
The valid range is 1 to 255.

*filePath* — File path
The formats that can be entered are described below.

- Absolute path string handled by Java standard class "java.io.File"
  When targeting Android 10 (API 29) or later, please note that some files cannot be handled directly.
  See "3.5 Precautions - About Scoped Storage" for details.

- URI string of the following scheme name handled by the class "android.net.Uri" prepared for Android
  - file://
  - content://

  It is necessary to specify the URI string obtained from "Storage Access Framework" for this parameter. Please note that URI created without being obtained from "Storage Access Framework" may not be able to open the file.

Specify the file name of the optional font to register.
Supported file extension is .bin.
See "Register Optional Font" of the Display command in "Technical Reference for Display" for details on optional font data.

Exception

**PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description

Display is changed to Standby mode when this method is executed. A selecting template is deselected.

When this method is executed with optional font registered status, the registered optional fonts are deleted, and a new memory area is allocated to register optional fonts.
Used memory is not released even the registered optional fonts are deleted. The used memory can be reused after executing **defragment**.

In use via a printer, this method is ignored when Display is not connected to the printer.

## unregisterOptionFont — Delete optional font

Deletes registered optional fonts.

Syntax public void **unregisterOptionFont**() throws **PrinterException**

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description Display is changed to Standby mode when this method is executed. A selecting template is deselected.

All registered optional fonts are deleted.
Used memory is not released even the optional fonts are deleted. The used memory can be reused after executing **defragment**.

In use via a printer, this method is ignored when Display is not connected to the printer.

## controlMacroRegistration — Start/End of macro registration

Specifies start or end of macro registration.

Syntax public void **controlMacroRegistration**(int *macroID,*
MacroRegistrationFunction *control*) throws **PrinterException**

Parameter

| | |
|---|---|
| *macroID* | Macro ID<br>The valid range is -1 to 127.<br>Do not specify the macro IDs of 120 to 126 because they are being used for the system. |
| *control* | Macro registration processing<br>See "4.2.1(3)⑫ Macro registration processing (MacroRegistrationFunction)" for available constants. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

Description The procedures of the macro registration processing are as follows:
(1) Starts macro registration processing.
Specify -1 for *macroID.*
Specify **MACRO_REGISTRATION_START** for *control.*

(2) Executes methods.
Starts buffering of transmit data when methods are targeted in macro registration processing.
The transmit data of a macro registration processing target method which is executed during the buffering is not sent to the device, buffered in macro data buffer.
The maximum transmit data size to be able to buffer is 1024 bytes.
When the buffered transmit data exceeds the maximum size, a macro registration processing target method at the point of exceeding is to be error.
When the error occurs, data under the registration is discarded and canceled the macro mode.
Regarding transmit data which is held, finish the macro registration processing by procedure (3).
When a method is out of the macro registration processing target, it is executed immediately without buffering the transmit data.

Methods for the macro registration processing target are shown below.

- **showTemplate**
- **showSlide**
- **selectTemplate**
- **setTemplateImageData**
- **selectTemplateTextObject**
- **setTemplateTextAlignment**
- **setTemplateTextLeftMargin**
- **setTemplateTextLineSpacing**
- **setTemplateTextBold**
- **setTemplateTextUnderline**
- **setTemplateTextSize**
- **setTemplateTextFont**
- **setTemplateTextRegisteredFont**
- **setTemplateTextRightSpacing**
- **setTemplateTextColor**
- **setTemplateTextData**
- **setTemplateBarcodeData**
- **setTemplateQrCodeData**

(3) Finishes macro registration processing.
Specify a macro ID (0 to 127) to register at *macroID.*
When **MACRO_REGISTRATION_REGIST** is specified at *control*, buffered transmit data is sent to the device. The buffered transmit data is held even after transmitting to the device.
Display is changed to Standby mode when this method is executed.
A selecting template is deselected.

The holding transmit data is discarded by following processes.

- Specify **MACRO_REGISTRATION_CLEAR.**
- Specify **MACRO_REGISTRATION_START.**
- Execute **disconnect.**

The registered macro can be executed at **executeMacro.**

The process to delete the registered macro is as follow.
Specify **MACRO_REGISTRATION_START** at *control* and specify -1 for *macroID* to call this method.
Specify **MACRO_REGISTRATION_REGIST** at *control* and specify the macro ID to delete for *macroID*, and then call this method.
Display is changed to Standby mode when this method is executed.
A selecting template is deselected.

In use via a printer, this method is ignored when Display is not connected to the printer.

**Note** **Registered data at the shipping may be added or changed without prior notice for quality improvement.**

## getDisplayResponse — Get various response from Display

Gets response data from Display.

Syntax public void **getDisplayResponse**(int *id,* int *param,* Object *buf*) throws **PrinterException**

Parameter

| | |
|---|---|
| *id* | Display response type constant<br>See "4.2.1(2)⑤ Display response type" for available constants. |
| *param* | Command parameter<br>The buffer type varies depending on the Display response type constant.<br>See the following table for description of the value to be specified. |
| *buf* | Buffer for storing the retrieved response data<br>Stores the response data specified with id in the object specified with *buf*.<br>The buffer type varies depending on the Display response type constant.<br>See the following table for the buffer type. |

| Response Type Constant | | |
|---|---|---|
| | **Parameter** | **Description** |
| **DISPLAY_RESPONSE_REQUEST** (Execution response request) | | |
| | *param* | Specify 0 to 15 (00h to 0Fh) in int type. |
| | *buf* | Specify an int type array of length 1.<br>Specify 0 to 15 (00h to 0Fh) for *buf[0]*.<br>When the response is retrieved successfully, the response code of the execution response request is stored with 64 to 79 (40h to 4Fh). |
| **DISPLAY_RESPONSE_USER_AREA** (Send remaining capacity of user area) | | |
| | *param* | Specify 0 in int type. |
| | *buf* | Specify an int type array of length 1.<br>When the response is retrieved successfully, the remaining capacity of the user area is stored as a numerical value in bytes. |
| **DISPLAY_RESPONSE_TEMPLATE_ID_LIST** (Send template ID) | | |
| | *param* | Specify 0 in int type. |
| | *buf* | Specify an ArrayList<Integer> type array.<br>When the response is retrieved successfully, the registered template ID is stored as an int array. |
| **DISPLAY_RESPONSE_IMAGE_ID_LIST** (Send image ID) | | |
| | *param* | Specify 0 in int type. |
| | *buf* | Specify an ArrayList<Integer> type array.<br>When the response is retrieved successfully, the registered image ID is stored as an int array. |
| **DISPLAY_RESPONSE_SLIDE_ID_LIST** (Send slide ID) | | |
| | *param* | Specify 0 in int type. |
| | *buf* | Specify an ArrayList<Integer> type array.<br>When the response is retrieved successfully, the registered slide ID is stored as an int array. |

| Response Type Constant | |
|---|---|
| **Parameter** | **Description** |
| **DISPLAY_RESPONSE_TEMPLATE_LABEL** (Send template name) | |
| *param* | Specify 0 to 127 (00h to 7Fh) in int type. |
| *buf* | Specify an ArrayList<String> type array.<br>When the response is retrieved successfully, the template name specified at template registration is stored as a character string. |
| **DISPLAY_RESPONSE_IMAGE_LABEL** (Send image name) | |
| *param* | Specify 0 to 63 (00h to 3Fh) in int type. |
| *buf* | Specify an ArrayList<String> type array.<br>When the response is retrieved successfully, the image name specified at image data registration is stored as a character string. |
| **DISPLAY_RESPONSE_SLIDE_LABEL** (Send slide name) | |
| *param* | Specify 0 to 91 (00h to 5bh) in int type. |
| *buf* | Specify an ArrayList<String> type array.<br>When the response is retrieved successfully, the slide name specified at slide data registration is stored as a character string. |

Exception **PrinterException**
**PrinterException** is thrown when an error occurs while calling this method.
See "**4.2.5 PrinterException Class**" for details of the error.
When the data transmission is failed, the communication with the device is ended, and **PrinterException** may be thrown. See **isConnect** for verifying the connection state with the device.

### 4.2.2 PrinterEvent Class

**PrinterEvent** class gets the end event that occurs when **startDiscoveryPrinter** or **startDiscoveryDevice** is completed.

(1) Method List

Methods provided by the **PrinterEvent** class are shown in the following table.

| Name | Description |
|---|---|
| **getEventType** | Get end event |

(2) End event constant

Constants used for getting the end event are shown in the following table.

| Constant Name | Description | Value |
|---|---|---|
| **EVENT_FINISHED_DISCOVERY** | Completion of **startDiscoveryPrinter** or **startDiscoveryDevice**. | 1 |

(3) Method Details

**getEventType** Get end event

Gets the end event when **startDiscoveryPrinter** or **startDiscoveryDevice** is completed.

Syntax public int **getEventType**()

Return value See "4.2.2(2) End event constant" for details of the value.

Description The judgement conditions for the end event are as follows.
・**startDiscoveryPrinter** has been completed
・**startDiscoveryDevice** has been completed
Even when the device was not discovered, **EVENT_FINISHED_DISCOVERY** is returned.

### 4.2.3 PrinterListener Interface

**PrinterListener** interface is for getting the end event when **startDiscoveryPrinter** or **startDiscoveryDevice** is completed.

(1) Method List

Methods of the **PrinterListener** interface are shown in the following table.

| Name | Description |
|---|---|
| **finishEvent** | End event of device search |

(2) Method Details

**finishEvent** End event of device search

End event that is called when **startDiscoveryPrinter** or **startDiscoveryDevice** is completed.

Syntax public void **finishEvent**(**PrinterEvent** *event*)

Parameter *event* End event
It is specified by **PrinterEvent** class.

Description This method is an interface, so it is not implemented.
Implement this method in the user application that receives the notification of the end event by completion of **startDiscoveryPrinter** or **startDiscoveryDevice**. Determine the type of the end event by **getEventType** in **PrinterEvent** class.

### 4.2.4 PrinterInfo Class

**PrinterInfo** class stores the information of the device found by **startDiscoveryPrinter** or **startDiscoveryDevice**.

(1) Method List

Port name can be retrieved. Methods of **PrinterInfo** class are shown in the following table.

| Name | Description |
|---|---|
| **getDevicePath** | Get device path |

(2) Method Details

**getDevicePath** Get device path

Gets the character string of the USB device file path from the device information found by **startDiscoveryPrinter** or **startDiscoveryDevice**.

Syntax public String **getDevicePath**()

Return value Device path

### 4.2.5 PrinterException Class

(1) Method List

Methods provided by the **PrinterException** class are shown in the following table.

| Name | Description |
|---|---|
| **PrinterException** | Constructor |
| **getErrorCode** | Get error code |

(2) Constant List

① Error code

Constants used for getting error codes are shown in following table.

| Constant Name | Description | Value |
|---|---|---|
| **ERROR_ACCESS_DENIED** | Failed to get the handle.*1 | -1 |
| | An unavailable port was specified. | |
| | An unsupported method was specified. | |
| **ERROR_SHARING_ VIOLATION** | An already opened port was specified. | -11 |
| **ERROR_PORT_NOT_ OPENED** | The port is not open. | -12 |
| **ERROR_DEVICE_NOT_CONNECTED** | There is a problem with connection between the Android device and the printer. | -21 |
| **ERROR_OFFLINE** | Disconnected state or the printer is offline. | -22 |
| **ERROR_EXTERNAL_DEVICE_NOT_CONNECTED** | Display is not connected. | -23 |
| **ERROR_DEVICE_INITIALIZE_FAILED** | Failed to change the printer settings. Data sending to the printer is not completed within the send timeout period, or data receiving from the printer is not completed within the receive timeout period. | -31 |
| **ERROR_DATA_SIZE_ZERO** | 0-byte data was specified. | -101 |
| **ERROR_OVER_MAX_DATA_SIZE** | Maximum data size is exceeded. | -102 |
| **ERROR_DATA_SIZE_INVALID** | Data size is invalid. | -103 |
| **ERROR_ENCODE_FAILED** | An error occurred in encoding text data.*1 | -111 |
| **ERROR_TIMEOUT** | Send timeout occurred. | -201 |
| | Receive timeout occurred. | |
| **ERROR_FILE_NOT_FOUND** | The specified file is not found. | -301 |
| **ERROR_FILE_USED** | The specified file is in use by another process. | -302 |
| **ERROR_FILE_INVALID** | The specified file is invalid. | -303 |
| **ERROR_LOW_MEMORY** | Memory shortage occurred when loading image data file. | -311 |
| **ERROR_OVER_ MAX_IMAGE** | Either or both of width and height of image data exceeds the number of printable maximum dots. | -312 |

| Constant Name | Description | Value |
|---|---|---|
| **ERROR_LOGO_NOT_ DEFINED** | The logo is not registered. | -313 |
| **ERROR_LOW_USER_AREA** | Remaining user area is insufficient. | -401 |
| **ERROR_LOW_EXTERNAL_ RAM** | Remaining RAM capacity is insufficient. | -402 |
| **ERROR_NOT_REGISTERED** | The template is not registered.<br>Image data is not registered.<br>Slide data is not registered.<br>The optional font is not registered.<br>The user-defined character is not registered. | -403 |
| **ERROR_NOT_UNREGISTERED** | The template is not deleted.<br>Image data is not deleted.<br>Slide data is not deleted.<br>The optional font is not deleted.<br>The user-defined character is not deleted. | -404 |
| **ERROR_INVALID_NO** | The specified value for the logo ID is invalid. | -501 |
| **ERROR_INVALID_DATA** | The specified data is invalid. | -503 |
| **ERROR_INVALID_PARAM** | The specified parameter is invalid. | -9999 |

*1: Abnormal processing might have occurred.

(3) Method Details

## PrinterException — Constructor

Constructor for the **com.seikoinstruments.sdk.thermalprinter.PrinterException** class.

Syntax public **PrinterException**(int *code*, String *message*)

## getErrorCode — Get error codes

Gets the error code for thrown exception.

Syntax public int **getErrorCode**()

Return value See "4.2.5(2) Constant List" for details of the error.

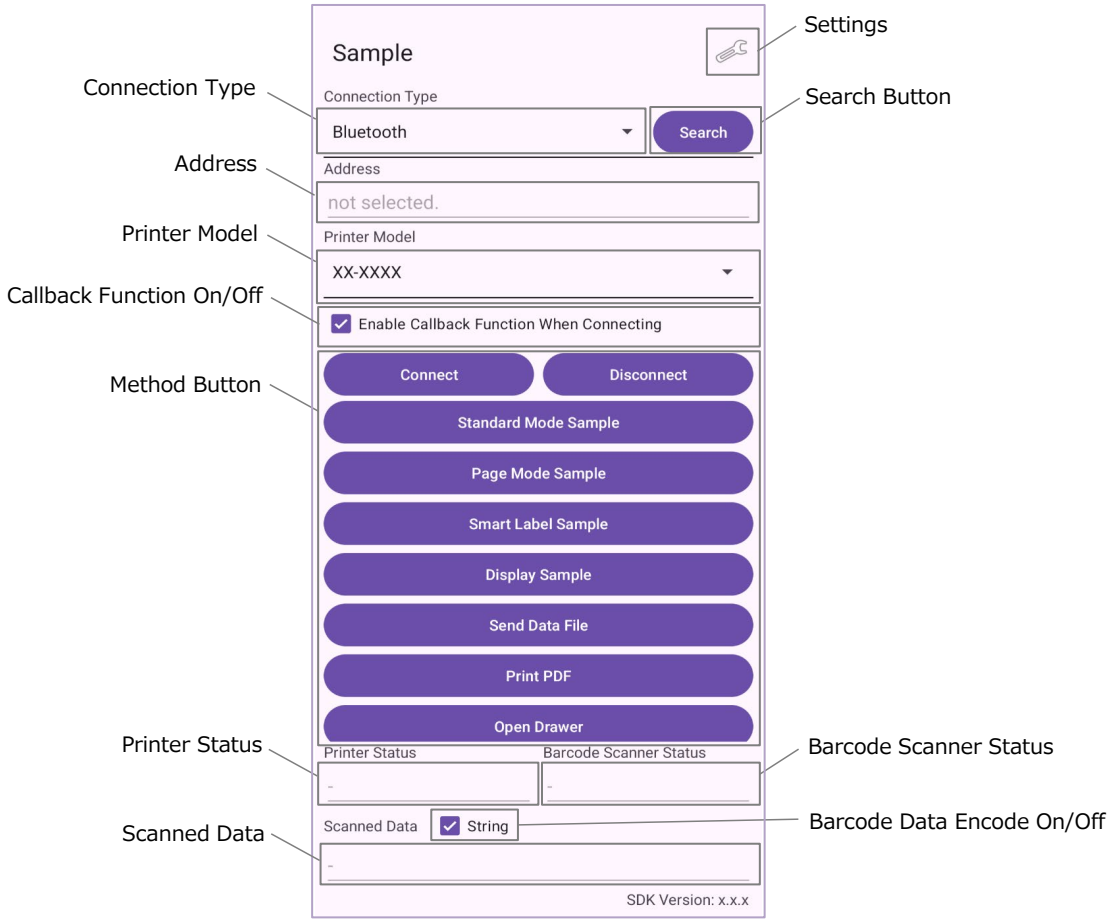
# Chapter 5
# Sample Program

This chapter describes the sample program provided by SII print class library.

## 5.1 Screen Layout

SII print class library includes the sample program in Android Studio project format.
This section describes the screen of the sample program.

### 5.1.1 Main Screen

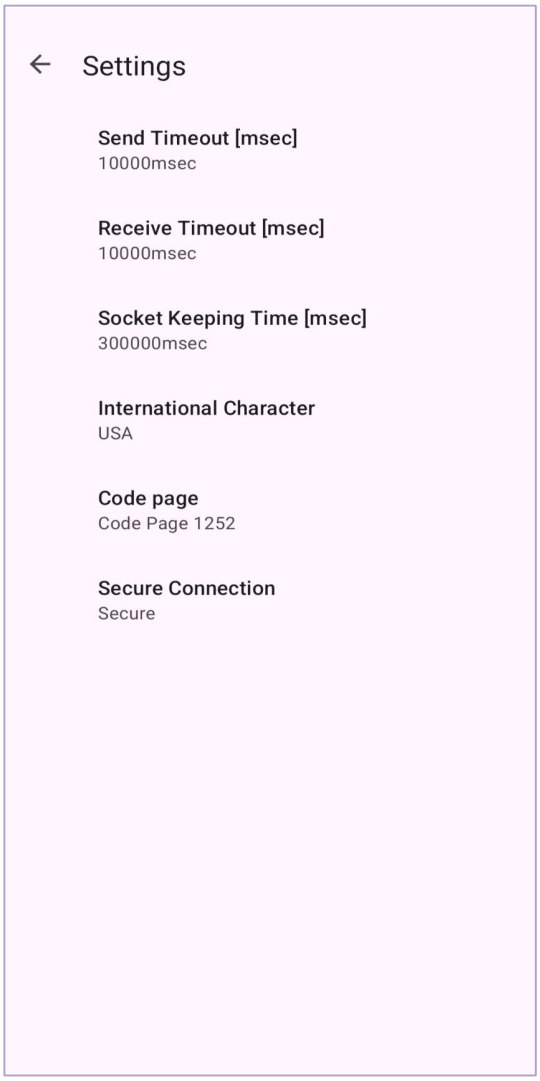


| Item | Description |
|---|---|
| Settings | Tapping the [Settings] button opens the function setting screen.<br>In order to go back to the main screen, tap ← button on the top left of the screen. |
| Connection type | Selects the connection type to the printer. |
| Search Button | Starts searching for the type of printer specified in [Connection Type]. Transits to the printer search view. A list of the searched printers is displayed. The printer is selected by tapping the searched printer and returns to the main view. |
| Address | Displays the information about the selected printer. |
| Printer Model | Specifies the printer model. |
| Callback Function On/Off | Select whether to enable the callback function when connecting to the printer.<br>On : Starts the callback function when connecting.<br>Off : The callback function does not respond. |
| Method Button*1*2 | In addition to the method buttons for executing **connect** and **disconnect**, the sample by the combination of some methods can be printed and checked for the operation of peripheral devices. |
| Printer Status | Displays the printer status.<br>When [Callback Function On/Off] is On, the latest printer status is displayed. |
| Barcode Scanner Status | Displays the connection status of the barcode scanner.<br>DSP-A01 does not support the barcode scanner. |
| Barcode Data Encode On/Off | Selects the conversion of barcode data read by the barcode scanner.<br>DSP-A01 does not support the barcode scanner. |
| Scanned Data | Displays the barcode data read by the barcode scanner.<br>DSP-A01 does not support the barcode scanner. |

*1: Supported functions vary by model. Only supported functions can be operated.
*2: It is valid only when Display is used via a printer.

### 5.1.2 [Settings] Screen

Various setting functions are displayed in [Settings].




## 5.2 Precaution

The sample program is subject to change without notice.

No guarantee of proper operation and support are provided for the sample program.

# Appendix A
# Character Set

## A.1 Codepage Table (Character Code Table)

The codepages when **COUNTRY_USA** is set for the international character set are shown below. Display results of the specific character codes vary depending on the setting of the international character set. See "A.2 International Character Set" for the specific character codes.

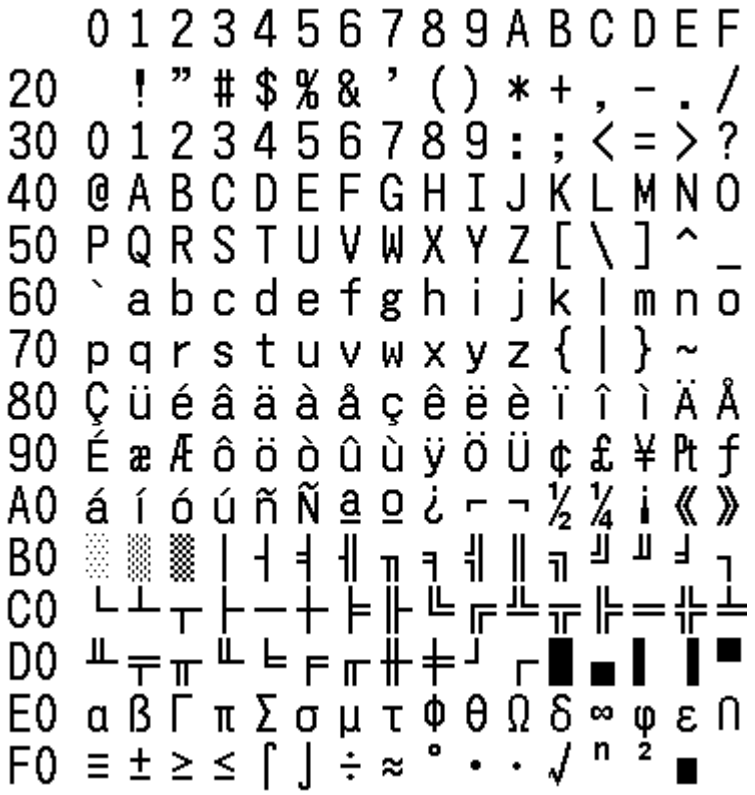



**Figure A-1 CODE_PAGE_437 (USA, Standard Europe)**

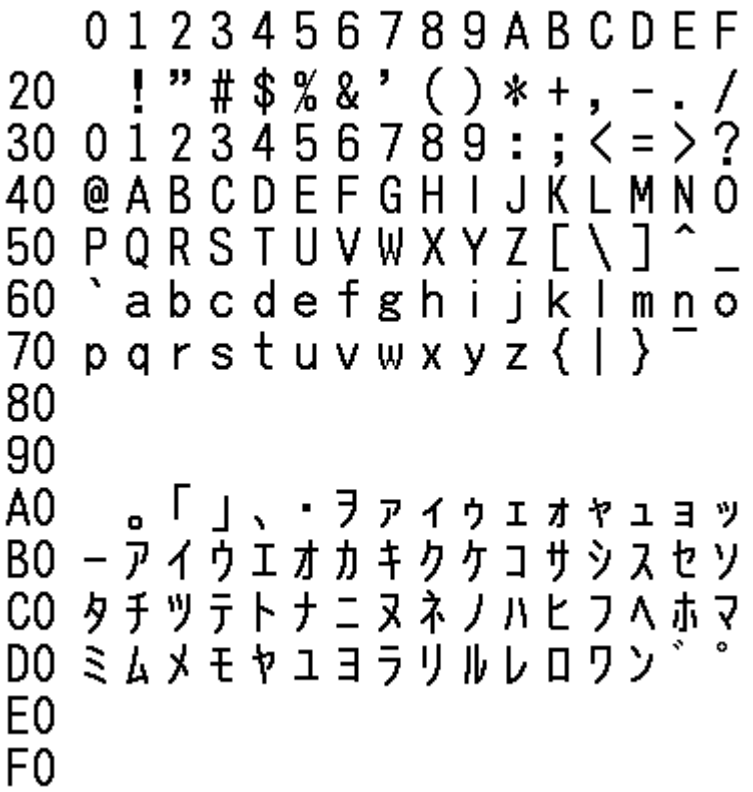



**Figure A-2 CODE_PAGE_KATAKANA**

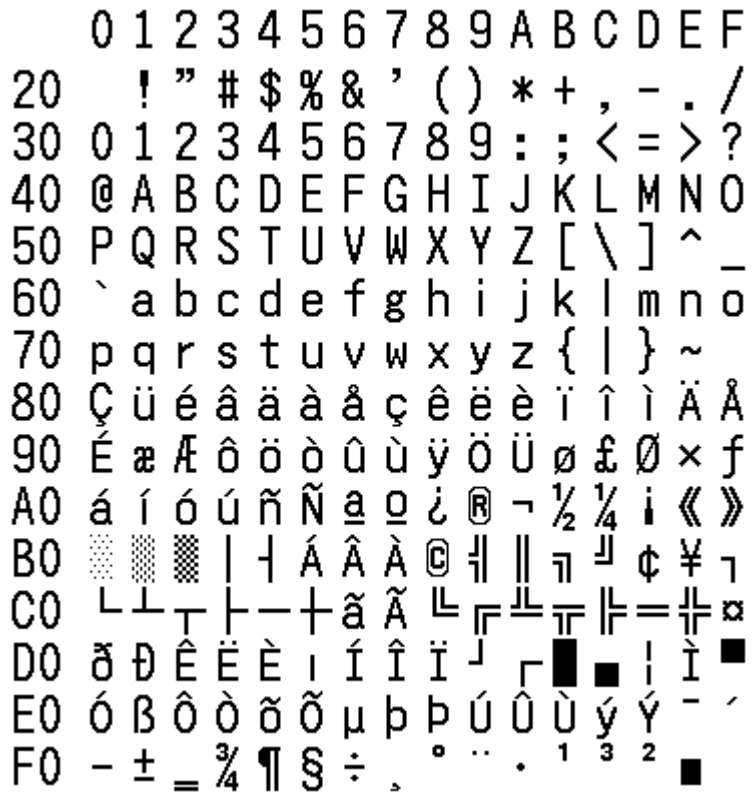



**Figure A-3 CODE_PAGE_850 (Multilingual)**

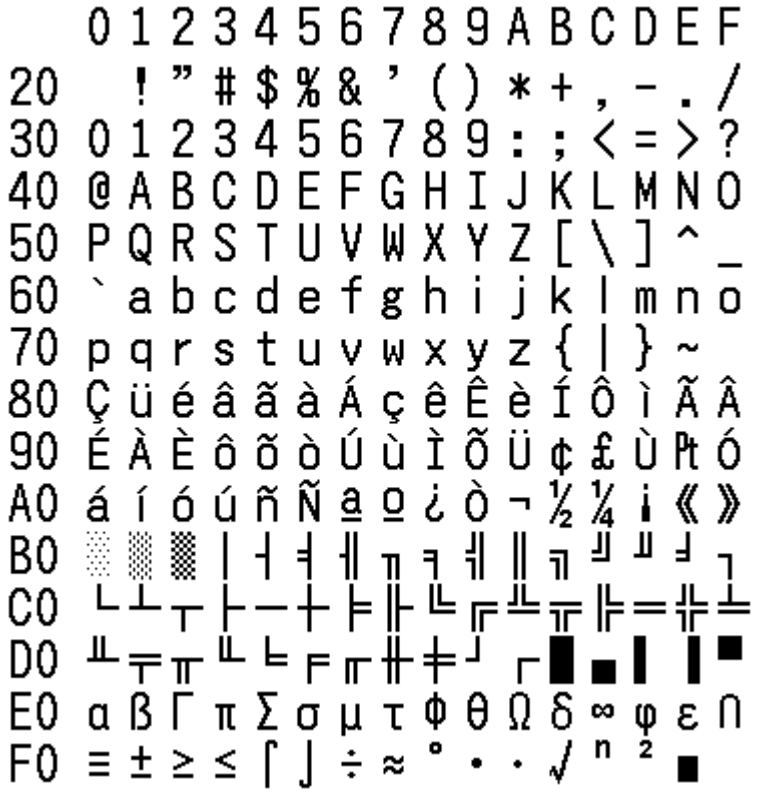



**Figure A-4 CODE_PAGE_860 (Portuguese)**

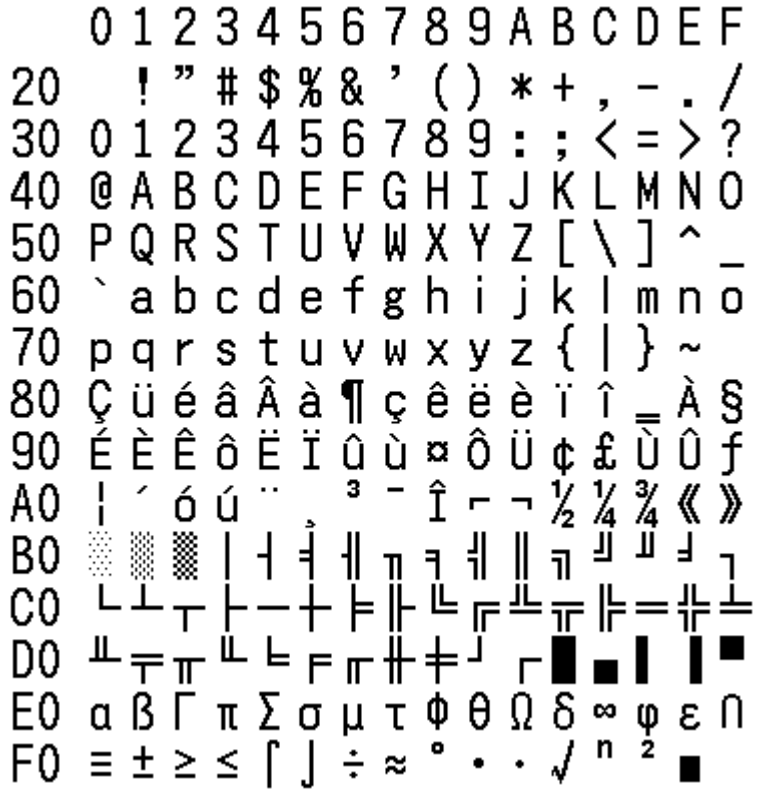



**Figure A-5 CODE_PAGE_863 (Canadian-French)**

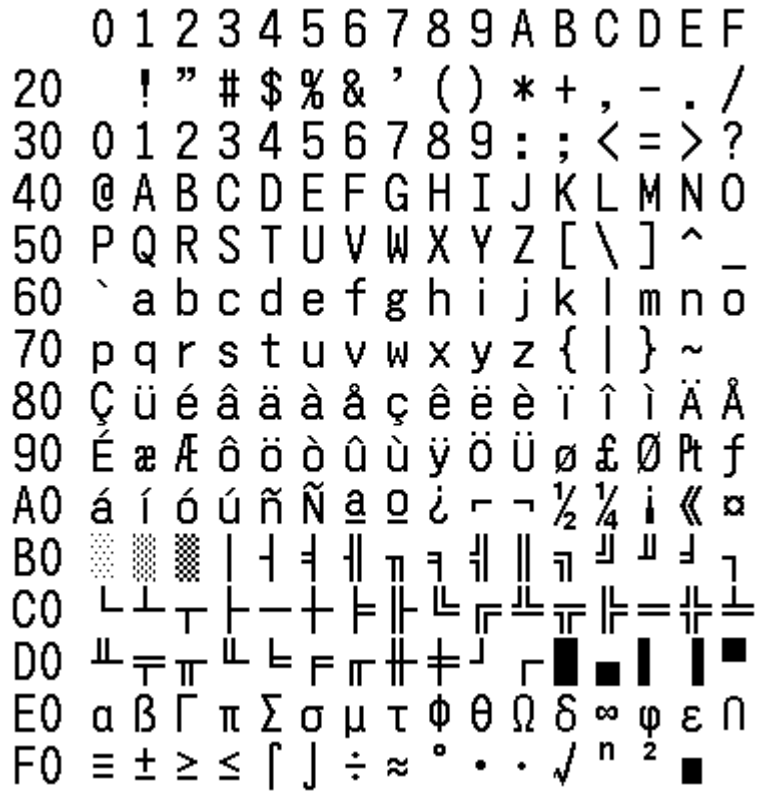



**Figure A-6 CODE_PAGE_865 (Nordic)**

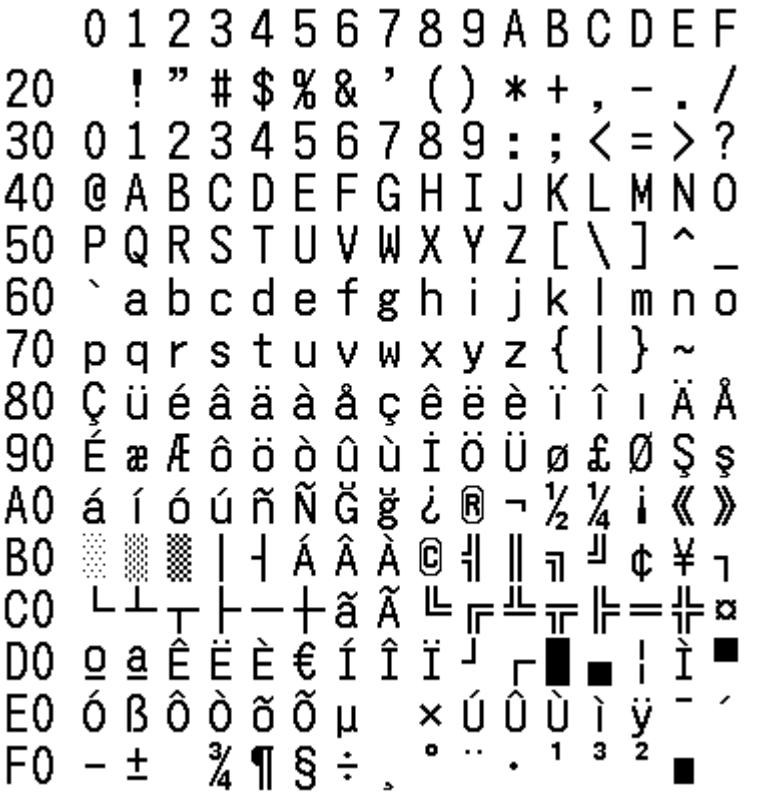



**Figure A-7 CODE_PAGE_857 (Turkish)**

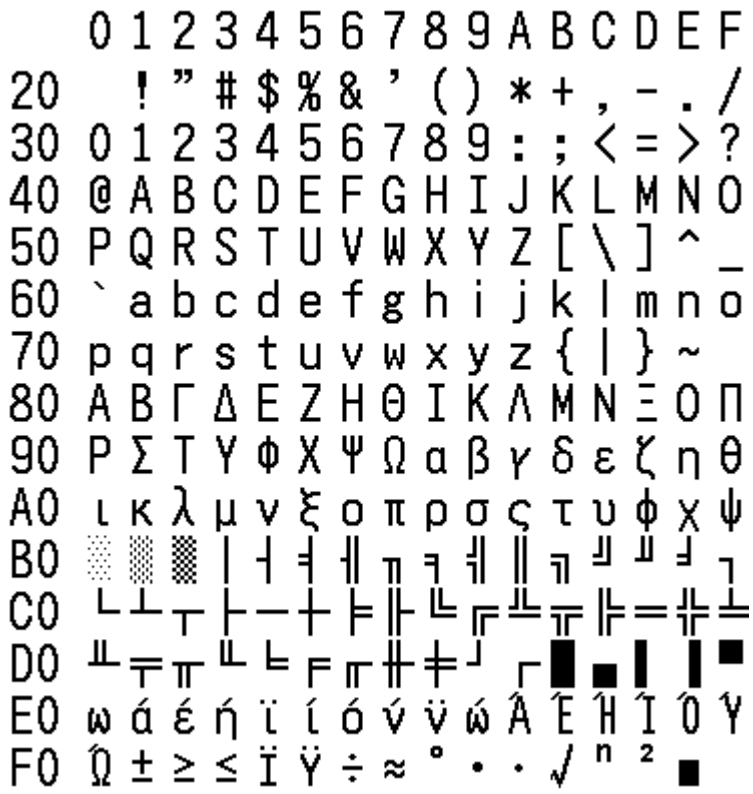



**Figure A-8 CODE_PAGE_737 (Greek)**

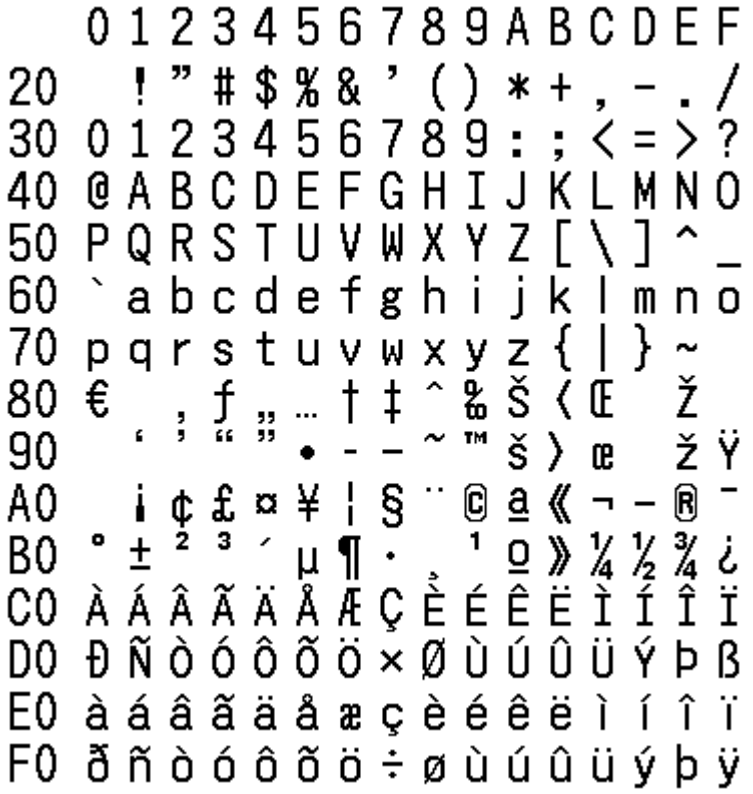



**Figure A-9 CODE_PAGE_1252 (Latin)**

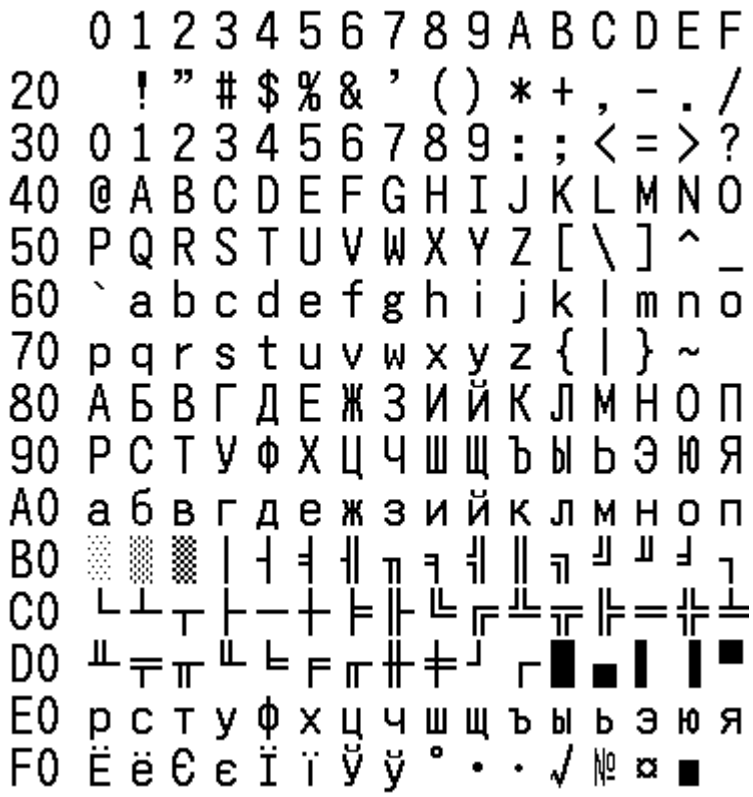



**Figure A-10 CODE_PAGE_866 (Russian)**

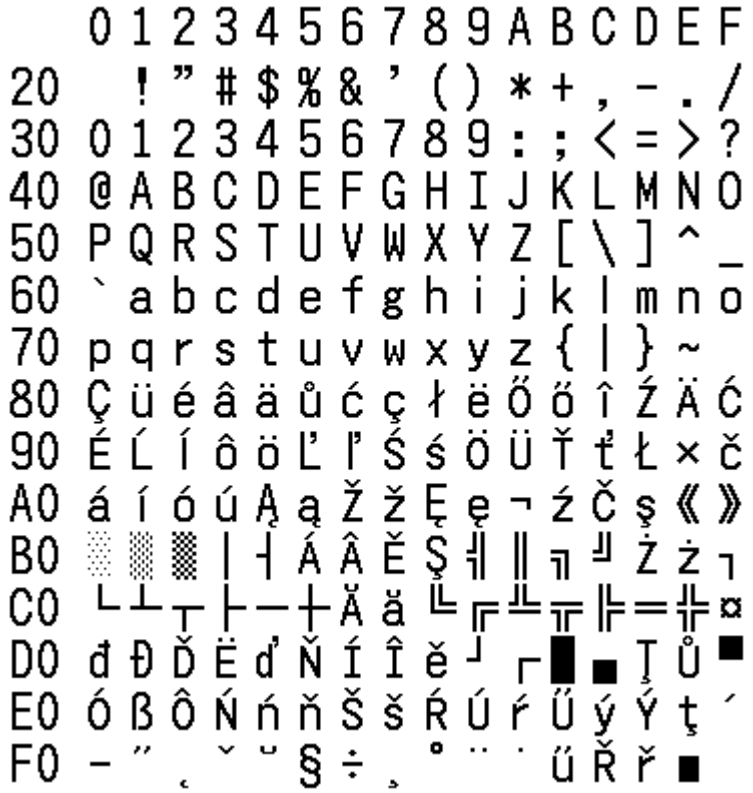



**FIgure A-11 CODE_PAGE_852 (Eastern Europe)**

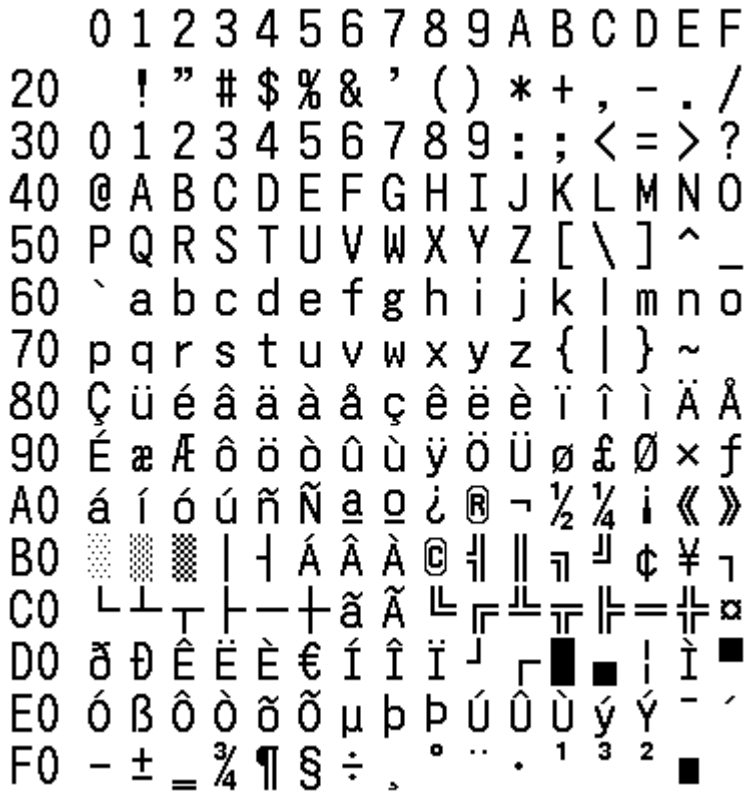



**Figure A-12 CODE_PAGE_858 (Euro)**

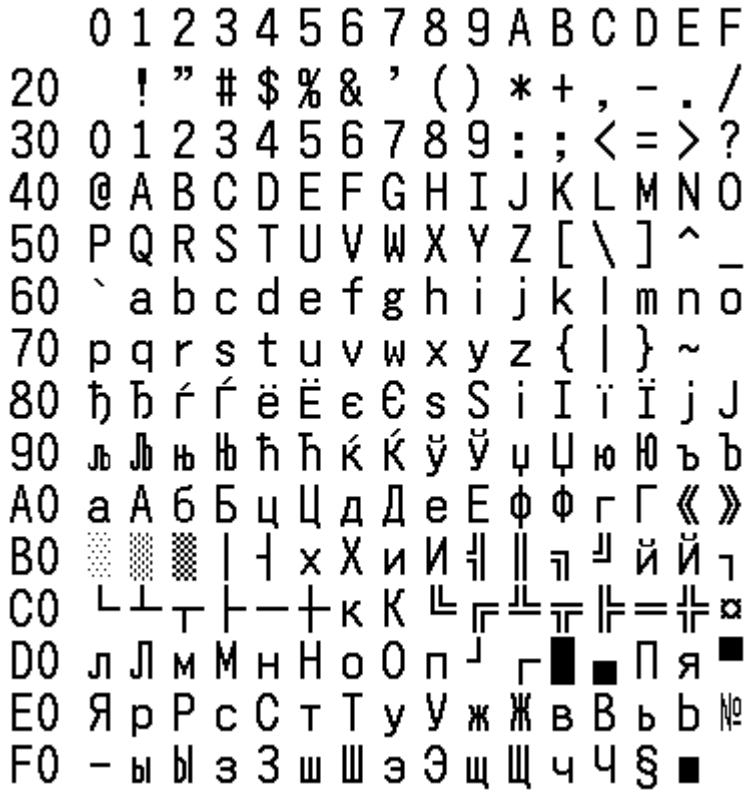



**Figure A-13 CODE_PAGE_855 (Cyrillic)**




**Figure A-14 CODE_PAGE_864 (Arabic)**




**Figure A-15 CODE_PAGE_1250 (Central European)**




**Figure A-16 CODE_PAGE_1251 (Cyrillic)**

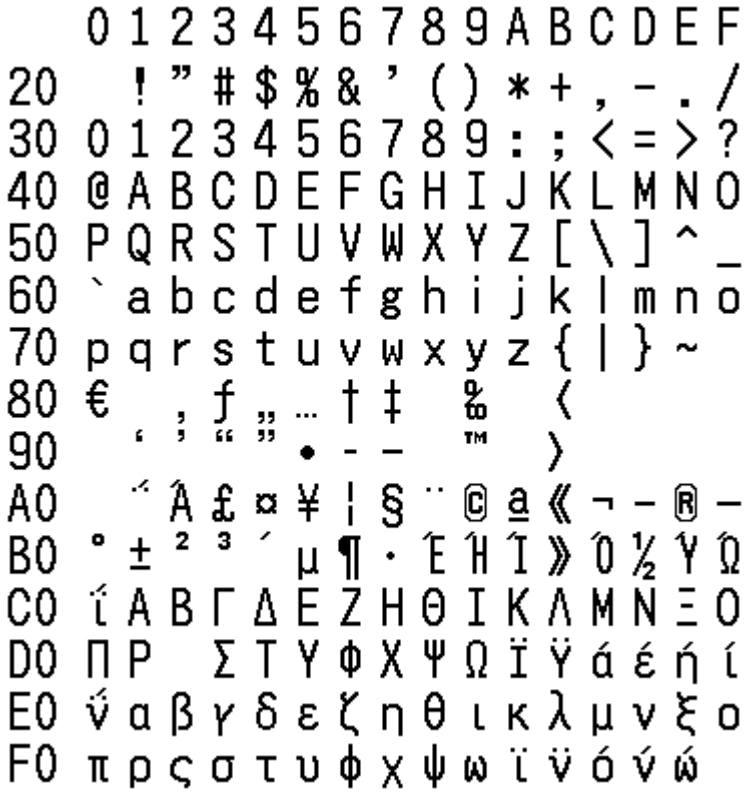



**Figure A-17 CODE_PAGE_1253 (Greek)**




**Figure A-18 CODE_PAGE_1254 (Turkish)**

## A.2 International Character Set

Display results of the specific character codes vary depending on the setting of the international character set.
The following table shows the specific character codes and their display results.

| | 23 | 24 | 40 | 5B | 5C | 5D | 5E | 60 | 7B | 7C | 7D | 7E |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COUNTRY_USA | # | $ | @ | [ | \ | ] | ^ | ` | { | \| | } | ~ |
| COUNTRY_FRANCE | # | $ | à | ° | ç | § | ^ | ` | é | ù | è | ¨ |
| COUNTRY_GERMANY | # | $ | § | Ä | Ö | Ü | ^ | ` | ä | ö | ü | ß |
| COUNTRY_ENGLAND | £ | $ | @ | [ | \ | ] | ^ | ` | { | \| | } | ~ |
| COUNTRY_DENMARK_1 | # | $ | @ | Æ | Ø | Å | ^ | ` | æ | ø | å | ~ |
| COUNTRY_SWEDEN | # | ¤ | É | Ä | Ö | Å | Ü | é | ä | ö | å | ü |
| COUNTRY_ITALY | # | $ | @ | ° | \ | é | ^ | ù | à | ò | è | ì |
| COUNTRY_SPAIN | ₧ | $ | @ | ¡ | Ñ | ¿ | ^ | ` | ¨ | ñ | } | ~ |
| COUNTRY_JAPAN | # | $ | @ | [ | ¥ | ] | ^ | ` | { | \| | } | ~ |
| COUNTRY_NORWAY | # | ¤ | É | Æ | Ø | Å | Ü | é | æ | ø | å | ü |
| COUNTRY_DENMARK_2 | # | $ | É | Æ | Ø | Å | Ü | é | æ | ø | å | ü |
| COUNTRY_SPAIN_2 | # | $ | á | ¡ | Ñ | ¿ | é | ` | í | ñ | ó | ú |
| COUNTRY_LATIN_AMERICA | # | $ | á | ¡ | Ñ | ¿ | é | ü | í | ñ | ó | ú |
| COUNTRY_ARABIA | # | $ | @ | [ | \ | ] | ^ | ` | { | \| | } | ~ |

**Figure A-19 International Character Set**