Woosim Android SDK
Programmer Reference

Version 2.6.1

March 2020

Contents

1. Overview

This release of the Woosim printer Android Software Development Kit(SDK) documentation provides information about Android application development.

Copyright © 2020 Woosim Systems Inc.

1.1. Introduction

This SDK package includes a sample project based Android Studio IDE, and the project is composed of 4 modules using Woosim Android library. The library file name is WoosimLibxxx.jar. These libraries included in each sample module are exactly same.

You can easily create new projects by using provided samples.

1.2. Definitions and Abbreviations

AP	Access Point
DBCS	Double-Byte Character Set
HRI	Human Readable Interpretation
IDE	Integrated Development Environment
JDK	Java Development Kit
MCU	Main Control Unit
MSR	Magnetic Stripe Reader
SCR	Smart Card Reader
SDK	Software Development Kit
USB OTG	Universal Serial Bus On-The-Go

2.   Summary

Package name of library: com.woosim.printer.

This SDK can be compatible with M16C, ARM, and RX MCU, but the old MCU like M37702 may not work properly. Most of descriptions are based on RX MCU printers.

2.1.   Message Handler Parameters

name	value	meaning
MESSAGE_PRINTER	100	data income from printer
N.A	200	reserved for internal use
N.A	201	reserved for internal use
N.A	202	reserved for internal use

2.2.   Public Methods

2.2.1. WoosimService Class

void   clearRcvBuffer( )
            clear receiving buffer explicitly

void   processRcvData(byte[] data, int length)
            parsing incoming data from printer

2.2.2. WoosimBarcode Class

byte[]   create2DBarcodeAztec (int layer, int level, int size, byte[] data)
              Create printable Aztec 2D barcode byte stream

byte[]   create2DBarcodeDataMatrix (int height, int width, int size, byte[] data)
              Create printable DATAMATRIX(ECC200) 2D barcode byte stream

byte[]   create2DBarcodeMaxicode (int mode, byte[] data)
              Create printable Maxicode 2D barcode byte stream

byte[]   create2DBarcodeMicroPDF417 (int width, int column, int row, int ratio, byte[] data)
              Create printable micro PDF417 2D barcode byte stream

byte[]   create2DBarcodePDF417 (int width, int column, int level, int ratio, boolean HRI,
              byte[] data)
              Create printable PDF417 2D barcode byte stream

byte[]   create2DBarcodeQRCode (int version, byte level, int size, byte[] data)
              Create printable QR code 2D barcode byte stream

byte[]   create2DBarcodeTruncPDF417 (int width, int column, int level, int ratio, boolean HRI, byte[] data)
Create printable Truncated PDF417 2D barcode byte stream

byte[]   createBarcode (int type, int width, int height, boolean HRI, byte[] data)
Create printable 1D barcode byte stream

byte[]   createGS1Databar (int type, int seg, byte[] data)
Create printable GS1 databar byte stream

2.2.3. WoosimImage Class

Bitmap   adjustBitmap (Bitmap bmp, float brightness, float contrast)
Create a new Bitmap that adjusts brightness and contrast from a given Bitmap

byte[]   drawBitmap (int x, int y, Bitmap bmp)
Convert Bitmap data to user defined bit-image format excluding print command

byte[]   drawBox (int x, int y, int width, int height, int thickness)
Draw horizontal/vertical line or box on assigned position.

byte[]   drawColorBitmap (int x, int y, Bitmap bmp)
Convert color Bitmap data to user defined bit-image format excluding print command

byte[]   drawEllipse (int x, int y, int radiusW, int radiusH, int thickness)
Draw ellipse on assigned position.

byte[]   drawLine (int x1, int y1, int x2, int y2, int thickness)
Draw diagonal line on assigned position.

byte[]   fastPrintBitmap (int x, int y, int width, int height, Bitmap bmp)
It divides printing into several times if the image is large.

byte[]   printBitmap (int x, int y, int width, int height, Bitmap bmp)
Convert Bitmap data to user defined bit-image format including print command

byte[]   printBitmapLandscape (int x, int y, int width, int height, Bitmap bmp)
Convert Bitmap data to user defined bit-image format including print command in landscape mode

byte[]   printColorBitmap (int x, int y, int width, int height, Bitmap bmp)
Convert color Bitmap data to user defined bit-image format including print command

byte[]   printColorBitmapLandscape (int x, int y, int width, int height, Bitmap bmp)

Convert color Bitmap data to user defined bit-image format including print command in landscape mode

byte[]   printCompressedBitmap (int x, int y, int width, int height, Bitmap bmp)

Convert Bitmap data to compressed user defined bit-image format including print command

byte[]   printCompressedBitmapLandscape (int x, int y, int width, int height, Bitmap bmp)

Convert Bitmap data to compressed user defined bit-image format including print command in landscape mode

byte[]   printStdModeBitmap (Bitmap bmp)

Convert Bitmap data to user defined bit-image format including print command in standard mode

byte[]   printStdModeColorBitmap (Bitmap bmp)

Convert color Bitmap data to user defined bit-image format including print command in standard mode

byte[]   printStoredImage (int id)

Print image downloaded in printer.

2.2.4. WoosimCmd Class

byte[]   cutPaper (int mode)

Paper cutting.

byte[]   feedToMark ( )

Feed paper to the black mark position.

byte[]   getTTFcode (int width, int height String string)

Set TrueType font size and get Unicode value from the given string.

Byte[]   initPrinter ( )

Initialize printer.

byte[]   MSR_1stTrackMode ( )

Enter MSR 1st track reading mode.

byte[]   MSR_2ndTrackMode ( )

Enter MSR 2nd track reading mode.

byte[]   MSR_3rdTrackMode ( )

Enter MSR 3rd track reading mode.

byte[]   MSR_doubleTrackMode ( )

Enter MSR double track reading mode.

byte[]   MSR_exit ( )

Exit MSR reading mode.

byte[]   MSR_tripleTrackMode ( )

Enter MSR triple track reading mode.

byte[]   moveAbsPosition (int distance)

Move printing position from the beginning of the line.

byte[]   moveRelPosition (int distance)

Move printing position from the current position.

byte[]   openCashDrawer ( )

Open cash drawer.

byte[]   PM_deleteData ( )

In page mode, delete all data in current printable area.

byte[]   PM_moveAbsVertical (int distance)

Move vertical printing position from the start position in page mode.

byte[]   PM_moveRelVertical (int distance)

Move vertical printing position from the current position in page mode.

byte[]   PM_printData ( )

Print data in page mode.

byte[]   PM_printStdMode ( )

Print data in page mode, and return to standard mode.

byte[]   PM_setArea (int x, int y, int width, int height)

Set printing area in page mode.

byte[]   PM_setDirection (int direction)

Set the printing direction and start position in page mode.

byte[]   PM_setPosition (int x, int y)

Set printing start position in page mode.

byte[]   PM_setStdMode ( )

Set standard mode.

byte[]   printData ( )

Print data and line feed.

byte[]   printDotFeed (int n)

Print the data in the print buffer and feed n dots.

byte[]   printLineFeed (int n)

Print the data in the print buffer and feed n lines.

byte[]   queryDeviceVersion ( )

Inquire device version and build date of the connected printer.

byte[]   queryModelName ( )

Inquire device model name and MCU type of the connected printer.

byte[]   queryStatus ( )

Inquire status of the connected printer.

byte[]   resetLineSpace ( )

Set line spacing to default value.

byte[]   SCR_enter ( )

Enter SCR mode.

byte[]   SCR_enterNonSecureMode ( )

Enter non-secure SCR reading mode.

byte[]   SCR_exit ( )

Exit SCR mode.

byte[]   SCR_ exitNonSecureMode ( )

Exit non-secure SCR reading mode.

byte[]   selectTTF (String filename)

Select TrueType font file name stored in the printer.

byte[]   setAlignment (int align)

Align data in one line.

byte[]   setCharacterSpace (int n)

Set right-side character spacing.

byte[]   setCodeTable (int mcu, int codeTable, int size)

Set character code table and font size.

byte[]   setLeftMargin (int margin)

Set left margin.

byte[]   setLineSpace (int n)

Set line spacing.

byte[]   setPageMode ( )

Set page mode.

byte[]   setPositionFromMark (int distance)

Set the movement position from the black mark.

byte[]   setPrintingWidth (int width)

Set printing area width after left margin.

byte[]   setTabPosition (int[] pos)

Set horizontal tab positions.

byte[]   setTextAlign (int align)

Align text data in one line.

byte[]   setTextStyle (boolean bold, boolean underline, boolean reverse, int extWidth, int extHeight)

Apply text style including bold, underline, reverse, and extension horizontal and vertical directions.

byte[]   setUpsideDown (boolean mode)

Turn on or off upside down printing.

byte[]   SMSR_enter ( )

Enter secured MSR reading mode.

byte[]   SMSR_exit ( )

Exit secured MSR reading mode.

byte[]   SMSR_writeData (byte[] data, int length)

Write command data for secured MSR.

3.   WoosimService Class

The WoosimService class defines general constants and methods for parsing incoming data from printer.

3.1.   Message Handler Parameters

public static final int **MESSAGE_PRINTER**

Some data is incoming from printer. The type of data and contents are dependent on below parameters.

arg1	arg2	obj	description
0x00	*obj* length	ByteBuffer	In the case that the meaning of the data cannot be read with only the data, the data received as ByteBuffer type is saved in *obj* and the byte unit length of the data is saved in *arg2*.
0x02	0		MSR or SCR reading failure
0x02	0x43	byte[3][] *	MSR 1st track read result. If MSR is 23 track version, reading data is track 2 and saved in *obj*[1], otherwise(12 track or 123 track version) it is track 1 and saved in *obj*[0].
0x02	0x44	byte[3][]	MSR 2nd track read result. If MSR is 23 track version, read data is track 3 and saved in *obj*[2], otherwise it is track 2 and saved in *obj*[1].
0x02	0x45	byte[3][]	MSR double track read result. If MSR is 23 track version, read data is track 2 and 3, otherwise it is track 1 and 2.
0x02	0x46	byte[3][]	MSR triple track read result. It can work only MSR 123 track version. MSR data are read from track 1, 2, and 3 and then saved in *obj*[0], *obj*[1], and *obj*[2] respectively.
0x02	0x47	byte[3][]	MSR 3rd track read result. It can work only MSR 123 track version. Read data is track 3 and saved in *obj*[2]
0x02	0x6E	byte[3][]	SCR read result. *obj*[1] has data same as MSR track 2.

* 2 dimension array *obj* has MSR data of each track if the track read successfully. Track 1 data is saved in *obj*[0], track 2 is in *obj*[1], and track 3 is in *obj*[2] respectively. The size of each array is same as that of track, that is *obj*[0][76], *obj*[1][37], and *obj*[2][104].

3.2. Constants

public static final int **MESSAGE_PRINTER**

   Constant value: 100 (0x64)

public static final int **MSR**

   Constant value: 2 (0x02)

public static final int **UNPRESCRIBED**

   Constant value: 0 (0x00)

3.3. Public Constructors

public **WoosimService**(Handler handler)

3.4. Public Methods

public void **clearRcvBuffer**( )

   Clear receiving buffer explicitly.

   It can be used to reset receiving buffer to make sure there is no remaining garbage data.

public void **processRcvData**(byte[] data, int length)

   Parse data stream from printer.

   Incoming data from printer may be MSR data or response signal in protocol mode etc.

   Parameters

   data   The data stream from printer

   length   The length of data

4. WoosimBarcode Class

The WoosimBarcode class provides methods to create command and data stream which can print 1D or 2D barcode for Woosim printer.

4.1. Constants

public static final int **UPC_A**
   Constant value: 65 (0x41)

public static final int **UPC_E**
   Constant value: 66 (0x42)

public static final int **EAN13**
   Constant value: 67 (0x43)

public static final int **EAN8**
   Constant value: 68 (0x44)

public static final int **CODE39**
   Constant value: 69 (0x45)

public static final int **ITF**
   Constant value: 70 (0x46)

public static final int **CODEBAR**
   Constant value: 71 (0x47)

public static final int **CODE93**
   Constant value: 72 (0x48)

public static final int **CODE128**
   Constant value: 73 (0x49)

4.2. Public Methods

public static byte[] **createBarcode**(int type, int width, int height, boolean HRI, byte[] data)
   Create printable 1D barcode byte stream.
   Parameters

type   The barcode type

       Refer to the above constants

width   The barcode width (1 ~ 8)

       If the width is out of printable area, barcode printing is ignored

height   The barcode height by dot unit (0 ~ 255)

HRI   Human Readable Interpretation(HRI) characters print mode indicator

       If it is true, numeric values are printed at the bottom of barcode

data   The barcode source data

       Data length and value is dependent on barcode type. Refer to the command manual

Returns

 Returns printable byte streams with control command or null if any argument has invalid value.

---

public static byte[] create2DBarcodePDF417(int width, int column, int level, int ratio, boolean HRI, byte[] data)

Create printable PDF417 2D barcode byte stream.

Parameters

width   The barcode width (1 ~ 8)

       If the width is out of printable area, barcode printing is ignored

column   The column number of 2D barcode (1 ~ 30)

level   The security level to restore when barcode image is damaged (0 ~ 8)

ratio   The horizontal and vertical ratio (2 ~ 5)

HRI   Human Readable Interpretation(HRI) characters print mode indicator

       If it is true, numeric values are printed at the bottom of barcode

data   The barcode source data

Returns

 Returns printable byte streams with control command or null if any argument has invalid value.

---

public static byte[] create2DBarcodeDataMatrix(int height, int width, int size, byte[] data)

Create printable DATAMATRIX(ECC200) 2D barcode byte stream.

Refer to the command manual for details.

Parameters

height   The height of the symbol (0 : auto size)

width　　　　　The width of the symbol (0 : auto size)

size　　　　　The module size (1 ~ 8)

data　　　　　The barcode source data

Returns

Returns printable byte streams with control command or null if any argument has invalid value.

public static byte[] **create2DBarcodeQRCode**(int version, byte level, int size, byte[] data)

Create printable QR-CODE 2D barcode byte stream.

Refer to the command manual for relation between version and capacity by EC level.

Parameters

version　　　　The version of the symbol (0 ~ 40, 0 : auto size)

level　　　　　The EC level (0x4C: 7%, 0x4D: 15%, 0x51: 25%, 0x48: 30%)

size　　　　　The module size (1 ~ 8)

data　　　　　The barcode source data

Returns

Returns printable byte streams with control command or null if any argument has invalid value.

public static byte[] **create2DBarcodeMicroPDF417**(int width, int column, int row, int ratio, byte[] data)

Create printable Micro PDF417 2D barcode byte stream.

Refer to the command manual for combination between column, row and maximum data size.

Parameters

width　　　　　The barcode width (1 ~ 8)

　　　　　　　If the width is out of printable area, barcode printing is ignored

column　　　　The column number of 2D barcode (1 ~ 4)

row　　　　　The row number of 2D barcode (4 ~ 44, 0 : auto size)

ratio　　　　　The horizontal and vertical ratio (2 ~ 5)

data　　　　　The barcode source data

Returns

Returns printable byte streams with control command or null if any argument has invalid value.

public static byte[] **create2DBarcodeTruncPDF417**(int width, int column, int level, int ratio, boolean HRI, byte[] data)

Create printable Truncated PDF417 2D barcode byte stream.

Parameters

> *width*　　　　The barcode width (1 ~ 8)
>
> 　　　　　　　If the width is out of printable area, barcode printing is ignored
>
> *column*　　　The column number of 2D barcode (1 ~ 4)
>
> *level*　　　　The security level to restore when bar code image is damaged (0 ~ 8)
>
> *ratio*　　　　The horizontal and vertical ratio (2 ~ 5)
>
> *HRI*　　　　　Human Readable Interpretation(HRI) characters print mode indicator
>
> 　　　　　　　If it is true, numeric values are printed at the bottom of barcode
>
> *data*　　　　The barcode source data

Returns

> Returns printable byte streams with control command or null if any argument has invalid value.

public static byte[] **create2DBarcodeMaxicode**(int mode, byte[] data)

Create printable Maxicode 2D barcode byte stream.

Maxicode can be printed with RX MCU printer only.

Refer to the command manual for relation between mode and data.

Parameters

> *mode*　　　　The mode of Maxicode (2 ~ 6)
>
> *data*　　　　The barcode source data

Returns

> Returns printable byte streams with control command or null if any argument has invalid value.

public static byte[] **create2DBarcodeAztec**(int layer, int level, int size, byte[] data)

Create printable Aztec 2D barcode byte stream.

Aztec code can be printed with RX MCU printer firmware released after Feb. 26, 2019.

Parameters

> *layer*　　　　Aztec layers
>
> 　　　　　　　0 : auto sized with ECC level
>
> 　　　　　　　1~32 : full with auto ECC level
>
> 　　　　　　　101~104 : compact
>
> *level*　　　　Error correction level (0 ~ 3)

    size   Module size (1 ~ 8)

    data   The barcode source data

Returns

    Returns printable byte streams with control command or null if any argument has invalid value.

public static byte[] **createGS1Databar**(int type, int seg, byte[] data)

Create printable GS1 databar byte stream.

GS1 databar can be printed with RX MCU printer firmware released after Oct. 11, 2012.

Refer to the command manual for details.

Parameters

    type   The GS1 databar type (0 ~ 6)

    seg   The segment per row that should be even number (2 ~ 20)
                      This parameter is valid only for type 6

    data   The GS1 source data array

Returns

    Returns printable byte streams with control command or null if any argument has invalid value.

5. WoosimImage Class

The WoosimImage class provides methods to convert bitmap image data to printable image data by Woosim printer.

5.1. Public Methods

public static Bitmap **adjustBitmap** (Bitmap bmp, float brightness, float contrast)

    Create a new Bitmap that adjusts brightness and contrast from a given Bitmap.

    Parameters

        bmp   The original bitmap

        brightness   The brightness value to be adjusted (-255 ~ 255)

        contrast   The contrast value to be adjusted (0 ~ 10)

    Returns

       A new bitmap with brightness and contrast adjustments.

    Example code

```
BitmapFactory.Options options = new BitmapFactory.Options();
options.inScaled = false;
Bitmap bmp = BitmapFactory.decodeResource(getResources(), R.drawable.logo, options);
Bitmap newBmp = WoosimImage.adjustBitmap(bmp, -100, (float) 1.3);
byte[] data = WoosimImage.printColorBitmap(0, 0, 0, 0, newBmp);
bmp.recycle();
newBmp.recycle();

sendData(WoosimCmd.setPageMode());
sendData(data);
sendData(WoosimCmd.PM_setStdMode());
```

public static byte[] **drawBitmap** (int x, int y, Bitmap bmp)

    Convert Bitmap data to user defined bit-image formats.

    It should be called in page mode.

    Because returned data does not include print command, other images can be added.

    The maximum length of original bitmap image is 255(dots).

    Parameters

        x   The horizontal starting position to print in dot unit

        y   The vertical starting position to print in dot unit

        bmp   The raw bitmap image data

    Returns

       Byte array of user defined bit-image.

public static byte[] **drawBox** (int x, int y, int width, int height, int thickness)

Draw horizontal/vertical line or box on assigned position.

It should be called in page mode.

Because returned data does not include print command, other items can be added.

If the value of width or height is 0, it represents vertical line or horizontal line respectively.

Parameters

x   The horizontal starting position to print in dot unit

y   The vertical starting position to print in dot unit

width   The box width

height   The box height

thickness The thickness of line

Returns

Returns command byte streams to draw box on assigned position or null in case invalid argument.

public static byte[] **drawColorBitmap** (int x, int y, Bitmap bmp)

Convert color Bitmap data to user defined bit-image formats.

It should be called in page mode.

Because returned data does not include print command, other images can be added.

The maximum length of original bitmap image is 255(dots).

Parameters

x   The horizontal starting position to print in dot unit

y   The vertical starting position to print in dot unit
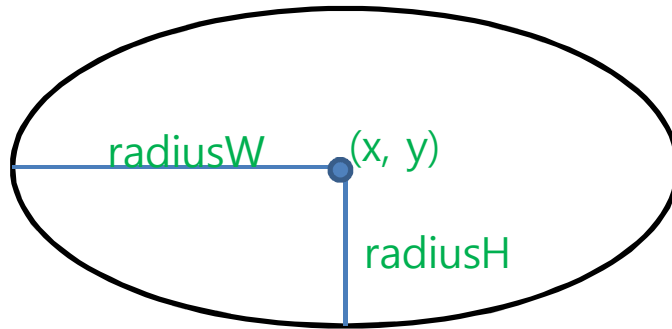
bmp   The raw bitmap image data

Returns

Byte array of user defined bit-image.

public static byte[] **drawEllipse** (int x, int y, int radiusW, int radiusH, int thickness)

Draw ellipse on assigned position.

It should be called in page mode.

Because returned data does not include print command, other items can be added.

Parameters

x	x-coordinate of central point
y	y-coordinate of central point
radiusW	radius of horizontal axis
radiusH	radius of vertical axis
thickness	The thickness of line

Returns

Returns command byte streams to draw ellipse on assigned position or null in case invalid argument.

public static byte[] **drawLine** (int x1, int y1, int x2, int y2, int thickness)

Draw line on assigned position.

It should be called in page mode.

Because returned data does not include print command, other items can be added.

x1	x-coordinate of start position
y1	y-coordinate of start position
x2	x-coordinate of end position
y2	y-coordinate of end position
thickness	The thickness of line

Returns

Returns command byte streams to draw line on assigned position or null in case invalid argument.

public static byte[] **fastPrintBitmap** (int x, int y, int width, int height, Bitmap bmp)

It works similar to printBitmap().

While printBitmap() starts printing after receiving all image data, it divides printing into several times if the image is large (more than 255 dots length).

Parameters

x	The horizontal starting position to print in dot unit
y	The vertical starting position to print in dot unit
width	The printing area width in dot unit. If width is less or equal to 0, it is used image width. Maximum value is according to each product width: 1 inch: 192, 2 inch: 384, 3 inch: 576, 4 inch: 832
height	The printing area height in dot unit. If height is less or equal to 0, it is used image height. It can be more than 2400.
bmp	The raw bitmap image data

Returns

    Returns printable byte streams with control command.

public static byte[] **printBitmap** (int x, int y, int width, int height, Bitmap bmp)

Convert Bitmap data to user defined bit-image formats.

It should be called in page mode. It can cover large image within the maximum page length limit, and returned byte array includes print command.

Parameters

x	The horizontal starting position to print in dot unit
y	The vertical starting position to print in dot unit
width	The printing area width in dot unit. If width is less or equal to 0, it is used image width. Maximum value is according to each product width: 1 inch: 192, 2 inch: 384, 3 inch: 576, 4 inch: 832
height	The printing area height in dot unit. If height is less or equal to 0, it is used image height. Maximum value is 2400
bmp	The raw bitmap image data

Returns

    Returns printable byte streams with control command.

public static byte[] **printBitmapLandscape** (int x, int y, int width, int height, Bitmap bmp)

Convert Bitmap data to user defined bit-image formats in landscape mode.

It should be called in page mode. It can cover large image within the maximum page length limit, and returned byte array includes print command.

Parameters

x	The horizontal starting position to print in dot unit
y	The vertical starting position to print in dot unit
width	The printing area width in dot unit. It is not width of landscape image.

> Maximum value is according to each product width:
>
> 1 inch: 192, 2 inch: 384, 3 inch: 576, 4 inch: 832

height	The printing area height in dot unit. It is not height of landscape image Maximum value is 2400.
bmp	The raw bitmap image data

Returns

Returns printable byte streams with control command.

---

public static byte[] **printColorBitmap** (int x, int y, int width, int height, Bitmap bmp)

Convert color Bitmap data to user defined bit-image formats.

It should be called in page mode. It can cover large image within the maximum page length limit, and returned byte array includes print command.

Parameters

x	The horizontal starting position to print in dot unit
y	The vertical starting position to print in dot unit
width	The printing area width in dot unit. If width is less or equal to 0, it is used image width. Maximum value is according to each product width: 1 inch: 192, 2 inch: 384, 3 inch: 576, 4 inch: 832
height	The printing area height in dot unit. If height is less or equal to 0, it is used image height. Maximum value is 2400
bmp	The raw bitmap image data

Returns

Returns printable byte streams with control command.

---

public static byte[] **printColorBitmapLandscape** (int x, int y, int width, int height, Bitmap bmp)

Convert color Bitmap data to user defined bit-image formats in landscape mode.

It should be called in page mode. It can cover large image within the maximum page length limit, and returned byte array includes print command.

Parameters

x	The horizontal starting position to print in dot unit
y	The vertical starting position to print in dot unit
width	The printing area width in dot unit. It is not width of landscape image. Maximum value is according to each product width: 1 inch: 192, 2 inch: 384, 3 inch: 576, 4 inch: 832
height	The printing area height in dot unit. It is not height of landscape image

Maximum value is 2400.

bmp The raw bitmap image data

Returns

Returns printable byte streams with control command.

public static byte[] **printCompressedBitmap** (int x, int y, int width, int height, Bitmap bmp)

Convert Bitmap data to compressed user defined bit-image formats.

It should be called in page mode. It can cover large image within the maximum page length limit, and returned byte array includes print command.

Parameters

x The horizontal starting position to print in dot unit

y The vertical starting position to print in dot unit

width The printing area width in dot unit. If width is less or equal to 0, it is used image width. Maximum value is according to each product width:
1 inch: 192, 2 inch: 384, 3 inch: 576, 4 inch: 832

height The printing area height in dot unit. If height is less or equal to 0, it is used image height. Maximum value is 2400

bmp The raw bitmap image data

Returns

Returns printable byte streams with control command.

public static byte[] **printCompressedBitmapLandscape** (int x, int y, int width, int height, Bitmap bmp)

Convert Bitmap data to compressed user defined bit-image formats in landscape mode.

It should be called in page mode. It can cover large image within the maximum page length limit, and returned byte array includes print command.

Parameters

x The horizontal starting position to print in dot unit

y The vertical starting position to print in dot unit

width The printing area width in dot unit. It is not width of landscape image.
Maximum value is according to each product width:
1 inch: 192, 2 inch: 384, 3 inch: 576, 4 inch: 832

height The printing area height in dot unit. It is not height of landscape image
Maximum value is 2400.

bmp The raw bitmap image data

Returns

Returns printable byte streams with control command.

public static byte[] **printStdModeBitmap** (Bitmap bmp)

Convert Bitmap data to user defined bit-image formats.

It can be called in standard mode. It can cover large image more than 255 dots length, and returned byte array includes print command.

Parameters

bmp   The raw bitmap image data

Returns

Returns printable byte streams with control command.

public static byte[] **printStdModeColorBitmap** (Bitmap bmp)

Convert color Bitmap data to user defined bit-image formats.

It can be called in standard mode. It can cover large image more than 255 dots length, and returned byte array includes print command.

Parameters

bmp   The raw bitmap image data

Returns

Returns printable byte streams with control command.

public static byte[] **printStoredImage** (int id)

Print image downloaded in printer.

User may need to refer to Woosim command manual to find out the images can be downloaded.

Parameters

id   Stored image number starting from 1. The maximum count of id is dependent on MCU.

Returns

Command byte streams or null in case invalid argument.

5.2. Deprecated Methods

public static byte[] **fastPrintARGBbitmap** (int x, int y, int width, int height, Bitmap bmp)

It is deprecated from version 2.2, and replaced by fastPrintBitmap().

public static byte[] **fastPrintRGBbitmap** (int x, int y, int width, int height, Bitmap bmp)

It is deprecated from version 2.2, and replaced by fastPrintBitmap().

public static byte[] **printARGBbitmap** (int x, int y, int width, int height, Bitmap bmp)

It is deprecated from version 2.2, and replaced by printBitmap().

public static byte[] **printRGBbitmap** (int x, int y, int width, int height, Bitmap bmp)

It is deprecated from version 2.2, and replaced by printBitmap().

public static byte[] **putARGBbitmap** (int x, int y, Bitmap bmp)

It is deprecated from version 2.2, and replaced by drawBitmap().

public static byte[] **putRGBbitmap** (int x, int y, Bitmap bmp)

It is deprecated from version 2.2, and replaced by drawBitmap().

6.   WoosimCmd Class

The WoosimCmd class provides methods to create command bytes stream for Woosim printers. These are helpful to improve program readability and to avoid from misusing of commands.

6.1.   Constants

public static final int **ALIGN_CENTER**
  Indicate center align. Constant value: 1 (0x01).

public static final int **ALIGN_LEFT**
  Indicate left align. Constant value: 0 (0x00).

public static final int **ALIGN_RIGHT**
  Indicate right align. Constant value: 2 (0x02).

public static final int **CT_ARABIC_FARSI**
  Code table for vendor specific Arabic. Constant value: 42 (0x2A).

public static final int **CT_ARABIC_FORMS_B**
  Code table Arabic Presentation Forms-B for Arabic. Constant value: 43 (0x2B).

public static final int **CT_AZERBAIJANI**
  Code table for vendor specific Azerbaijani. Constant value: 24 (0x18).

public static final int **CT_CP437**
  Code table CP437 for USA. Constant value: 0 (0x00).

public static final int **CT_CP720**
  Code table CP720 for Arabic. Constant value: 40 (0x28).

public static final int **CT_CP737**
  Code table CP737 for Greek. Constant value: 8 (0x08).

public static final int **CT_CP775**
  Code table CP775 for Baltic. Constant value: 11 (0x0B).

public static final int **CT_CP850**
  Code table CP850 for Western European (Latin I). Constant value: 2 (0x02).

public static final int **CT_CP852**

    Code table CP852 for Central European (Latin II). Constant value: 6 (0x06).

public static final int **CT_CP855**

    Code table CP855 for Cyrillic. Constant value: 16 (0x10).

public static final int **CT_CP857**

    Code table CP857 for Turkish. Constant value: 7 (0x07).

public static final int **CT_CP858**

    Code table CP858 for Western European with Euro sign. Constant value: 15 (0x0F).

public static final int **CT_CP860**

    Code table CP860 for Portuguese. Constant value: 3 (0x03).

public static final int **CT_CP862**

    Code table CP862 for Hebrew. Constant value: 10 (0x0A).

public static final int **CT_CP863**

    Code table CP863 for Canadian French. Constant value: 4 (0x04).

public static final int **CT_CP865**

    Code table CP865 for Nordic European. Constant value: 5 (0x05).

public static final int **CT_CP866**

    Code table CP866 for Cyrillic. Constant value: 9 (0x09).

public static final int **CT_DBCS**

    Code table for double-byte character set. Constant value: 255 (0xFF).

    It can be indicated to Korean, Chinese, Japanese shift-JIS, etc. Real code table depends on the printers.

public static final int **CT_HINDI_DEVANAGARI**

    Code table for Hindi Devanagari. Constant value: 50 (0x32).

public static final int **CT_IRAN_SYSTEM**

    Code table Iran System encoding standard for Persian language. Constant value: 41 (0x29).

    This code table is applied to firmware until May 2014.

public static final int **CT_ISO8859_15**

Code table ISO/IEC 8859-15 for Western European (Latin 9). Constant value: 13 (0x0D).

public static final int **CT_KATAKANA**

Code table for Japanese Katakana. Constant value: 1 (0x01).

public static final int **CT_POLISH**

Code table for vendor specific Polish. Constant value: 12 (0x0C).

public static final int **CT_WIN874**

Code table Windows 874 for Thai. Constant value: 30 (0x1E).

It is almost same as ISO/IEC 8859-11 and TIS-620.

public static final int **CT_WIN1250**

Code table Windows 1250 for Central European and Eastern European. Constant value: 18 (0x12).

public static final int **CT_WIN1251**

Code table Windows 1251 for Cyrillic. Constant value: 17 (0x11).

public static final int **CT_WIN1252**

Code table Windows 1252 for Latin. Constant value: 14 (0x0E).

public static final int **CT_WIN1253**

Code table Windows 1253 for Greek. Constant value: 19 (0x13).

public static final int **CT_WIN1254**

Code table Windows 1254 for Turkish. Constant value: 20 (0x14).

public static final int **CT_WIN1255**

Code table Windows 1255 for Hebrew. Constant value: 21 (0x15).

public static final int **CT_WIN1256**

Code table Windows 1256 for Arabic. Constant value: 41 (0x29).

This code table is applied to firmware after June 2014.

public static final int **CT_WIN1257**

Code table Windows 1257 for Baltic. Constant value: 23 (0x17).

public static final int **CT_WIN1258**

    Code table Windows 1258 for Vietnamese. Constant value: 22 (0x16).

public static final int **CUT_FULL**

    Indicate full cutting. Constant value: 0 (0x00).

public static final int **CUT_PARTIAL**

    Indicate partial cutting. Constant value: 1 (0x01).

public static final int **FONT_LARGE**

    Font size is 12x24. Some fonts use 16x24 instead.

    Constant value: 0 (0x00).

public static final int **FONT_MEDIUM**

    Font size is 9x24. Some fonts do not support this size.

    Constant value: 1 (0x01).

public static final int **FONT_SMALL**

    Font size is 8x16. Some fonts do not support this size.

    It is available RX MCU printers only. Constant value: 2 (0x02).

public static final int **MCU_ARM**

    Indicate ARM MCU.

public static final int **MCU_M16C**

    Indicate M16C MCU.

public static final int **MCU_RX**

    Indicate RX MCU.

6.2. Public Methods

6.2.1. Printer Commands

public static byte[] **cutPaper** (int mode)

    Set paper cutting mode and cut paper.

    It works for auto-cutter installed printers.

    Parameters

        mode   Paper cutting mode. It should be one of CUT_XXX constants.

    Returns

Command byte array or null if there is invalid argument.

public static byte[] **feedToMark** ( )

Feed the paper to the black mark setting position. Refer to the **setPositionFromMark** method description.

Returns

Command byte streams.

public static byte[] **initPrinter** ( )

Initialize printer.

Clear the data in the printer buffer and resets the printer configuration. The data in the receive buffer is not cleared.

Returns

Command byte streams.

public static byte[] **openCashDrawer** ( )

Generate pulse to open cash drawer.

Returns

Command byte streams.

public static byte[] **printData** ( )

Print the data in the print buffer and feed one line based on the current line spacing. It sets the print position to the beginning of the line.

Returns

Command byte streams.

public static byte[] **printDotFeed** (int n)

Print the data in the print buffer and feed n dots.

Parameters

n　　　　　　　Feed length in dot unit (0~255).

Returns

Command byte streams.

public static byte[] **printLineFeed** (int n)

Print the data in the print buffer and feed n lines. It sets the print position to the beginning of the line.

Parameters

n　　　　　　　Feed length in line unit (0~255).

Returns

    Command byte streams.

public static byte[] **queryDeviceVersion** ( )

Inquire device version and build date of the connected printer.

The response data from device is firmware dependant. For example, [Ver 2.0 2013/10/31].

Returns

    Command byte streams.

Example code

At first, send requirement to printer.

```
public void queryDeviceVersion (View v) {
        sendData(WoosimCmd.queryDeviceVersion());
}
```

Device version information coming from printer reaches the message handler.

The below code is referred if the data can be received through the routine supported in the library.

The device version information requested in the below code is saved on the bufferData variable

```
public void handleMessage(Message msg) {
    switch (msg.what) {
    ...
    case MESSAGE_READ:
        mWoosim.processRcvData((byte[])msg.obj, msg.arg1);
        break;

    case WoosimService.MESSAGE_PRINTER:
        switch (msg.arg1) {
        ...
         case WoosimService.UNPRESCRIBED:
           int length = msg.arg2;
           ByteBuffer bufferData = (ByteBuffer)msg.obj;
           ...
           break;
        }
        break;
    }
}
```

public static byte[] **queryModelName** ( )

Inquire device model name and MCU type of the connected printer.

The response data from device is firmware dependant. For example, R240(RX)_.

Returns

    Command byte streams.

Example code is same as **queryDeviceVersion** case.

public static byte[] **queryStatus** ( )

Inquire status of the connected printer.

The returned status includes condition information by paper sensor, cover sensor, mark sensor and so on. Refer to the "Woosim Command Manual" for details because the value varies with each printer model.

Returns

　　Command byte streams.

Example code

Declare a variable to count how many queries the printer status has been checked and increase the variable value for each query.

```java
private static int statusWaiting = 0;

public void getStatus(View v) {
        statusWaiting++;
        sendData(WoosimCmd.queryStatus());
}
```

Status byte coming from printer reaches the message handler. The handler examines whether the received data is status byte and there has ever been queries for printer status. And after examining, the handler analyzes the byte to check status of printer.

The code below is to check paper status and cover status for WSP-R240 model group.

```java
public void handleMessage(Message msg) {
    switch (msg.what) {
    ...
    case MESSAGE_READ:
        byte[] data = (byte[])msg.obj;
        if (statusWaiting > 0 && msg.arg1 == 1 && (data[0] & 0x30) == 0x30) {
            statusWaiting--;
            switch (data[0]) {
            case 0x30: /* Device status is OK */ break;
            case 0x31: /* Paper is not present */ break;
            case 0x32: /* Cover is opened */ break;
            case 0x33: /* Paper is not present & cover is opened */ break;
            }
        } else {
            ...
        }
        break;
    ...
```

public static byte[] **setPageMode** ( )

　　Change mode from standard mode to page mode.

　　Returns

　　　　Command byte streams.

public static byte[] **setPositionFromMark** (int distance)

　　Save the movement position from the black mark to the printer flash memory area.

　　It is strongly recommended that this method will be used in separated setting utility program because it does flash memory writing action.

After the movement position is set to the printer, paper will feed the amount of assigned distance from the black mark whenever call **feedToMark**.
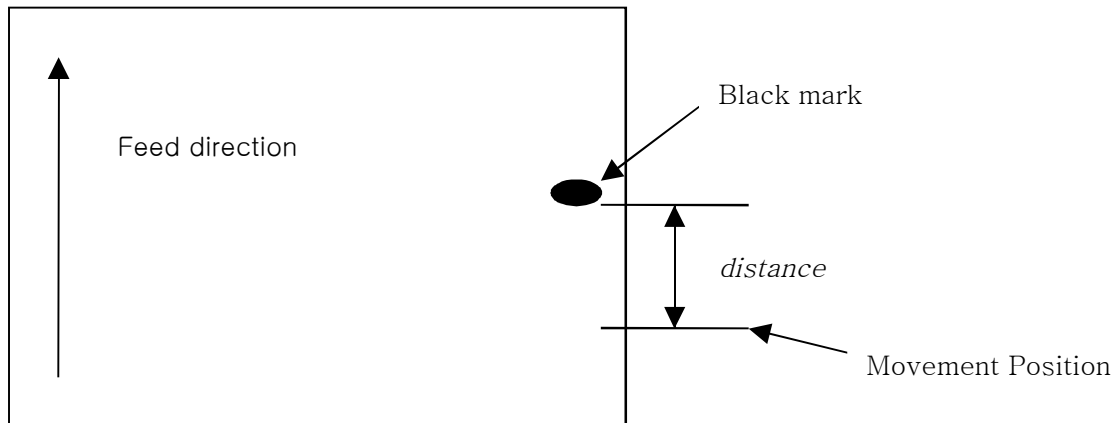
Parameters

> *distance* Distance from the mark in dot unit.

Returns

> Command byte streams.

6.2.2. Text Commands

public static byte[] **getTTFcode** (int width, int height, String string)

> Set TrueType font size and get Unicode value from the given string.
>
> Parameters
>
>> *width* Font width size that should be more than or equal to 4.
>>
>> *height* Font height size that should be more than or equal to 4.
>>
>> *string* String to be printed.
>
> Returns
>
>> Command and Unicode byte streams.

public static byte[] **moveAbsPosition** (int distance)

> Move printing position from the beginning of the line.
> If the position is out of printable range, command is ignored.
> Parameters
>
>> *distance* Distance from the beginning of the line in dot unit.
>
> Returns
>
>> Command byte streams.

public static byte[] **moveRelPosition** (int distance)

> Move printing position from the current position.
> If moved position exceeds printable area, command is ignored.

Parameters

distance   Distance from the current position in dot unit.

Returns

    Command byte streams.

public static byte[] **resetLineSpace** ( )

Set line spacing to default value. It can be set independently in standard mode and page mode.

Returns

    Command byte streams.

public static byte[] **selectTTF** (String filename)

Select TrueType font file name stored in the printer.

The designated TrueType font file should be saved in the printer in advance. Refer to the "Downloader Manual" to download TrueType font file to printer. The file name for TrueType font should be less than 30 letters long in English.

Parameters

String   TrueType font file name.

Returns

    Command byte streams.

Example code

```
byte[] cmdSetTTF = WoosimCmd.selectTTF("WSTTFL.ttf");
byte[] data = WoosimCmd.getTTFcode(40,40,"True Type font print");

ByteArrayBuffer buffer = new ByteArrayBuffer(256);
buffer.append(cmdSetTTF, 0, cmdSetTTF.length);
buffer.append(data, 0, data.length);

sendData(buffer.toByteArray());
sendData(WoosimCmd.printData());
```

public static byte[] **setAlignment** (int align)

Align data in one line including True Type font and bar codes as well as text.

It is affect in the area between the current position and the end of printing area.

It works for firmware released after Oct. 2015.

Parameters

align   Align type. It should be one of ALIGN_XXX constants.

Returns

    Command byte streams.

public static byte[] **setCharacterSpace** (int n)

Set right-side character spacing. It can be set independently in standard mode and page mode.

Parameters

 n   Character spacing in dot unit (0~255).

Returns

 Command byte streams.

---

public static byte[] **setCodeTable** (int mcu, int codeTable, int size)

Set character code table and font size.

Supported code tables are dependent on MCU version of target printer. In case of M16C or ARM version, they are CP437, Katakana, CP850, CP860, ISO8859-15, Polish, and DBCS.

CT_DBCS means one of double byte character set like Korean, Chinese, Japanese shift-JIS. It supports fixed font size according to language.

When code table is changed in application program, all of text styles are initialized.

It will be helpful to see "Command Manual" for more details.

Parameters

 mcu   Printer MCU type. It should be one of MCU_XXX constants.

 codeTable Character code table ID. It should be one of CT_XXX constants.

 size   Font size. It should be one of FONT_XXX constants.

Returns

 Command byte streams or null if there is any invalid argument.

---

public static byte[] **setLeftMargin** (int margin)

Set left margin.

If the margin value exceeds the printable area, command is ignored.

Parameters

 margin left margin value from the beginning of the line in dot unit

Returns

 Command byte streams.

---

public static byte[] **setLineSpace** (int n)

Set line spacing. It can be set independently in standard mode and page mode.

Parameters

 n   Line spacing in dot unit (0~255).

Returns

 Command byte streams.

---

public static byte[] **setPrintingWidth** (int width)

Set printing area width after left margin.

If the sum of left margin and printing area width exceeds the printable area, command is ignored.

Parameters

    width   the length of printing area

Returns

    Command byte streams.

public static byte[] **setTabPosition** (int[] pos)

    Set horizontal tab positions.

    Position values should be placed in ascending order.

    Total number of positions are less than or equal to 32.

    Parameters

        pos   integer array of column number from the beginning of the line

    Returns

        Command byte streams.

public static byte[] **setTextAlign** (int align)

    Align text data in one line.

    It is affect in the area between the current position and the end of printing area.

    Parameters

        align   Align type. It should be one of ALIGN_XXX constants.

    Returns

        Command byte streams.

public static byte[] **setTextStyle** (boolean bold, boolean underline, boolean reverse, int extWidth, int extHeight)

    Apply text style including bold, underline, reverse, and extension horizontal and vertical directions.

    Parameters

        bold   true for bold mode otherwise false

        underline   true for underline mode otherwise false

        reverse   true for black/white reverse mode otherwise false

   *extWidth*   The number of times extension on horizontal direction (1~8)

        extHeight   The number of times extension on vertical direction (1~8)

    Returns

        Command byte streams.

public static byte[] **setUpsideDown** (boolean mode)

Turn on or off upside down printing. It is enabled only when processed at the beginning of a line in standard mode.

Default printing	Upside-down printing
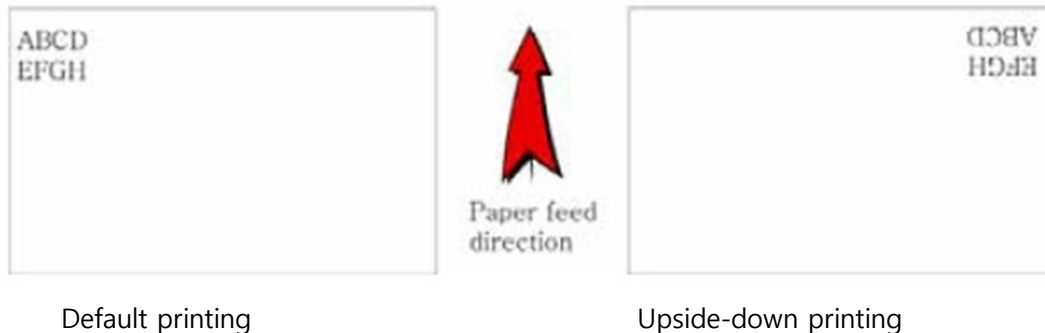
Parameters

mode   True for upside down printing, false otherwise.

Returns

Command byte streams.

6.2.3. Page Mode Commands

public static byte[] **PM_deleteData** ( )

In page mode, delete all data from current printable area.

Returns

Command byte streams.

public static byte[] **PM_moveAbsVertical** (int distance)

Move vertical printing position from the start position in page mode.

If moved position exceeds specified printing area, the command is ignored.

This method operates depending on the printing direction set by **PM_setDirection**.

Parameters

distance   vertical distance from the start position in dot unit

Returns

Command byte streams.

public static byte[] **PM_moveRelVertical** (int distance)

Move vertical printing position from the current position in page mode.

If moved position exceeds specified printing area, the command is ignored.

This method operates depending on the printing direction set by **PM_setDirection**.

Parameters

distance   vertical distance from the current position in dot unit

Returns

Command byte streams.

public static byte[] **PM_printData** ( )

Print data in page mode.

Returns

Command byte streams.

public static byte[] **PM_printStdMode** ( )

Print data in page mode, and change mode to standard mode.

Returns

Command byte streams.

public static byte[] **PM_setArea** (int x, int y, int width, int height)

Set printing area in page mode.

If the horizontal or vertical starting position is set outside the printable area, this command is ignored.

If the sum of *x* and *width* is bigger than printable area, the horizontal printable area is set to printable area minus *x*. If sum of *y* and *height* is bigger than printable area, the vertical printable area is set to printable area minus *y*.
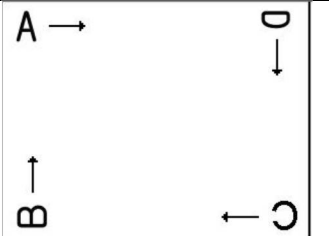
Parameters

x	The horizontal starting position to print in dot unit
y	The vertical starting position to print in dot unit
width	The printing area width in dot unit. It should be bigger than 0. The maximum horizontal printable area is according to each product width: 1 inch: 192, 2 inch: 384, 3 inch: 576, 4 inch: 832
height	The printing area height in dot unit. It should be bigger than 0. The maximum vertical printable area is 2400.

Returns

Command byte streams or null if there is any invalid argument.

public static byte[] **PM_setDirection** (int direction)

In page mode, set the printing direction and start position in printable area set by **PM_setArea**.

direction	starting position	printing direction	
0	upper left (A)	left → right	A → ▱ ↓
1	lower left (B)	bottom → up	
2	lower right (C)	right → left	↑
3	upper right (D)	top → bottom	฿ ← ↻

Parameters

    direction   specify printing direction and start position (0~3)

  Returns

    Command byte streams or null if there is any invalid argument.

---

public static byte[] **PM_setPosition** (int x, int y)

  Set printing start position in page mode.

  Parameters

    x   The horizontal starting position to print in dot unit

    y   The vertical starting position to print in dot unit

  Returns

    Command byte streams.

---

public static byte[] **PM_setStdMode** ( )

  Change mode from page mode to standard mode.

  Returns

    Command byte streams.

6.2.4. Card Reading Commands

public static byte[] **MSR_1stTrackMode** ( )

  Enter MSR 1st track reading mode.

  The 1st track means track 1 for 12 track MSR and 123 track MSR, and track 2 for 23 track MSR.

  The printer waits for MSR reading. After successful reading, printer sends the card data to host and exits MSR reading mode.

  Returns

    Command byte streams.

---

public static byte[] **MSR_2ndTrackMode** ( )

  Enter MSR 2nd track reading mode.

  The 2nd track means track 2 for 12 track MSR and 123 track MSR, and track 3 for 23 track MSR.

  The printer waits for MSR reading. After successful reading, printer sends the card data to host and exits MSR reading mode.

  Returns

    Command byte streams.

---

public static byte[] **MSR_3rdTrackMode** ( )

  Enter MSR 3rd track reading mode.

  The 3rd track means track 3 for 123 track MSR. It is working for 123 track MSR only.

The printer waits for MSR reading. After successful reading, printer sends the card data to host and exits MSR reading mode.

Returns

    Command byte streams.

public static byte[] **MSR_doubleTrackMode** ( )

Enter MSR double track reading mode.

The double track means track 1 and 2 for 12 track MSR and 123 track MSR, and track 2 and 3 for 23 track MSR.

The printer waits for MSR reading. After successful reading, printer sends the card data to host and exits MSR reading mode.

Returns

    Command byte streams.

public static byte[] **MSR_exit** ( )

Exit MSR reading mode.

Returns

    Command byte streams.

public static byte[] **MSR_tripleTrackMode** ( )

Enter MSR triple track reading mode.

The triple track means all tracks for 123 track MSR. It is working for 123 track MSR only.

The printer waits for MSR reading. After successful reading, printer sends the card data to host and exits MSR reading mode.

Returns

    Command byte streams.

public static byte[] **SCR_enter** ( )

Enter SCR mode.

After enter SCR mode, application should control directly the SCR module.

Returns

    Command byte streams.

public static byte[] **SCR_enterNonSecureMode** ( )

Enter non-secure SCR reading mode to get card data same as MSR track 2.

The printer waits for IC card reading. After successful reading, printer sends the card data to host and exits SCR reading mode.

Returns

    Command byte streams.

public static byte[] **SCR_exit** ( )

Exit SCR mode.

Returns

Command byte streams.

public static byte[] **SCR_exitNonSecureMode** ( )

Exit non-secure SCR reading mode.

Returns

Command byte streams.

public static byte[] **SMSR_enter** ( )

Enter secured MSR reading mode.

Returns

Command byte streams.

public static byte[] **SMSR_exit** ( )

Exit secured MSR reading mode.

Returns

Command byte streams.

public static byte[] **SMSR_writeData** (byte[] data, int length)

Write command data for secured MSR.

Parameters

data　　　　command data for secured MSR

length　　　length of command data

Returns

Command byte streams.

7. WspSamples

Woosim Android SDK includes several sample applications to provide useful help.

7.1. BTPrint

It is a traditional sample application. It includes following functions:
- Text file printing
- Bitmap printing
- An arbitrary ASCII text printing with attributes like underline, bold, alignment
- Various 1D and 2D bar code printing
- MSR mode setting and show the magnetic card data by swiping

To setup Bluetooth connection, you just touch action bar menu item after launch this application.
User can select one either secure connection or insecure connection.
You can download this APK file from Google Play Store.

7.2. WifiPrint

It has the same functions as BTPrint, but Wi-Fi protocol is used instead of Bluetooth.
To setup Wi-Fi connection, Android device and printer should be connect to the same AP. In case of printer, you can use "WSwLAN_Manager.exe" utility to connect the designated AP.
Woosim printers use 1470 or 9100 port as default.

7.3. UsbPrint

It has similar functions as BTPrint, but USB protocol is used instead of Bluetooth.
To setup USB connection, the printer should support USB connection and should be connected to the Android device through USB OTG.

7.4. Sample

It looks like API Demos of Android SDK.
Before selecting one, you should establish Bluetooth connection.
After Bluetooth connection, you may select one among samples. Samples will be added continually.

- **SignPad**
  Simple example to draw signature on touch device and print it.
- **Multi-Language (RX only)**
  It is an example to use several character sets in one application.
  It works properly only with Woosim printer installed RX MCU.

Please refer to <ESC t> command in Woosim command manual if you want to know code tables supported by printer. Some code tables cannot be used easily in Android because JDK does not provide them as available Charsets.

● **Page mode**
Simple examples to use page mode commands.

8. Tips for Application Program

■ PDF File Printing

From Android v5.0 (API level 21), you can print PDF files through PdfRenderer class.

This is the sample source code that can be used in conjunction with the sample project included in the SDK.

```java
/**
* print PDF file.
* @param file  PDF file to be printed.
*/
public void printPdfFile(final File file) {
  AsyncTask<Void, Void, Void> task = new AsyncTask<Void, Void, Void>() {
    @Override
    protected Void doInBackground(Void... params) {
      sendData(WoosimCmd.setPageMode());
      try {
        ParcelFileDescriptor pfd = ParcelFileDescriptor.open(file,
                                   ParcelFileDescriptor.MODE_READ_ONLY);
        PdfRenderer renderer = new PdfRenderer(pfd);
        // Roll paper width in dot unit
        // 2inch = 384, 3inch = 576, 4inch = 832
        int paperWidth = 384;
        for (int i=0 ; i < renderer.getPageCount() ; i++) {
          Page page = renderer.openPage(i);
          // The destination bitmap format must be ARGB.
          // Original page is resized to fit roll paper width.
          Bitmap bmp = Bitmap.createBitmap(
                        paperWidth,
                        page.getHeight()*paperWidth/page.getWidth(),
                        Config.ARGB_8888);
          page.render(bmp, null, null, Page.RENDER_MODE_FOR_PRINT);
          sendData(WoosimImage.printCompressedBitmap(
                        0, 0, bmp.getWidth(), bmp.getHeight(), bmp));
          bmp.recycle();
          page.close();
        }
        sendData(WoosimCmd.PM_setStdMode());
        renderer.close();
      } catch (IOException e) {
        e.printStackTrace();
      }
      return null;
    }
  };
  task.executeOnExecutor(AsyncTask.SERIAL_EXECUTOR);
}
```