SMART Printer SDK 2
User's Guide

2018, 04
IDP Corp., Ltd.

Table of Contents

1 Introduction

SMART Printer can be used through device driver so it can be printed in the same way as other printers'. However, when a user operates more complex functions such as encoding data on the magnetic card, duplex communication should be supported and each function should be followed step by step. "SMART Printer SDK" enables to develop the customized software for software developers and it supports the methods to operate various functions of printers.

[Compatibility]

SMART Printer SDK 2 is compatible with SDK 1 and it is added functions to support network printers.

Previous SmartCommon.dll in SDK1 is changed to SmartComm2.dll in SDK 2.

If your software is developed with SDK 1 and wants to use SDK 2, please change the "DLL" name from SmartComm2.dll to SmartCommon.dll in your program.

[USB and Network Communication]

Basically, SMART Printer offers only USB port for communication with PC. However, by using "Network Module", Both USB and network can be used.

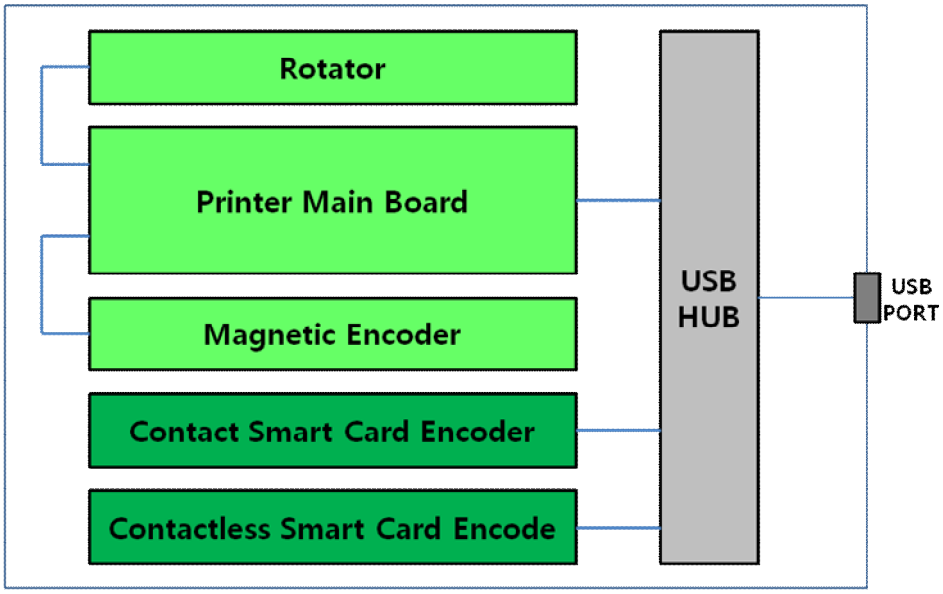

Figure 1. The Configuration of Basic Options.

Figure 1 shows the configuration of basic options and other options can be added as well. The options of Flipper for double-sided printing and Magnetic Encoder are connected to PC directly and are controlled by Printer Main Board. However, Contact Smart Card Encoder and Contactless Smart Card Encode options

are controlled by PC directly. For Smart Card encoding, a card is sent to the place for encoding by controlling main board and is encoded by each encoder. Smart Card Encoder for SMART Printer uses PC/SC Protocol. “SMART Printer SDK” offers various functions for smart card encoding but also it can be encoded directly using PC/SC protocol.

Figure2 shows the configuration how the options are installed with Network module. First, Network module should be installed to use SMART Printer on the network. After Network module is installed, Network module controls USB and network communications as Figure 2. Network Module communicates with PC using PC/SC protocol, so the encoding process should be done using “SMART Printer SDK” and it means that direct control in PC is not available through PC/SC protocol. In case of using Network module, Both Contact Smart Card Encoding and Contactless Smart Card Encoding are supported.

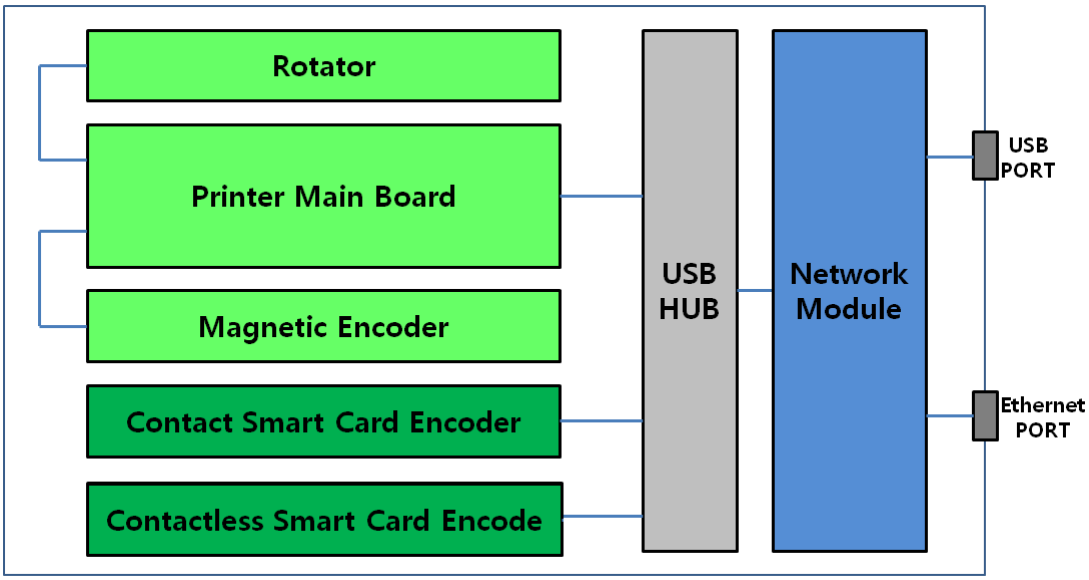

Figure 2. The Configuration of Using Network Module

“SMART Printer SDK” supports USB and network as the same interface for developers. Figure 3 shows the configuration of “SMART Printer SDK” and “SMART Printer Interface” is the functions for developers and can be used regardless of communication types. “SMART Printer Common Library” is the common functions those are used in SDK and is used “USB Library” or “Network Library” according to the communication types.

In the case of the configuration of basic options which is connected with USB, it communicates with Smart Card Encoder using PC/SC protocol. However, if Network module is installed, the Smart Card encoding process is through Network module. In the case of using encoding smart card encoding functions from SDK, it communicates automatically according to the connection, so it is recommended not to control PC/SC directly.

In case using Network module, basically, SSL (Secure Socket Layer) is supported and data is encrypted in safety. For more information about Network module configuration, please refer to NetAdmin part of the user manual.

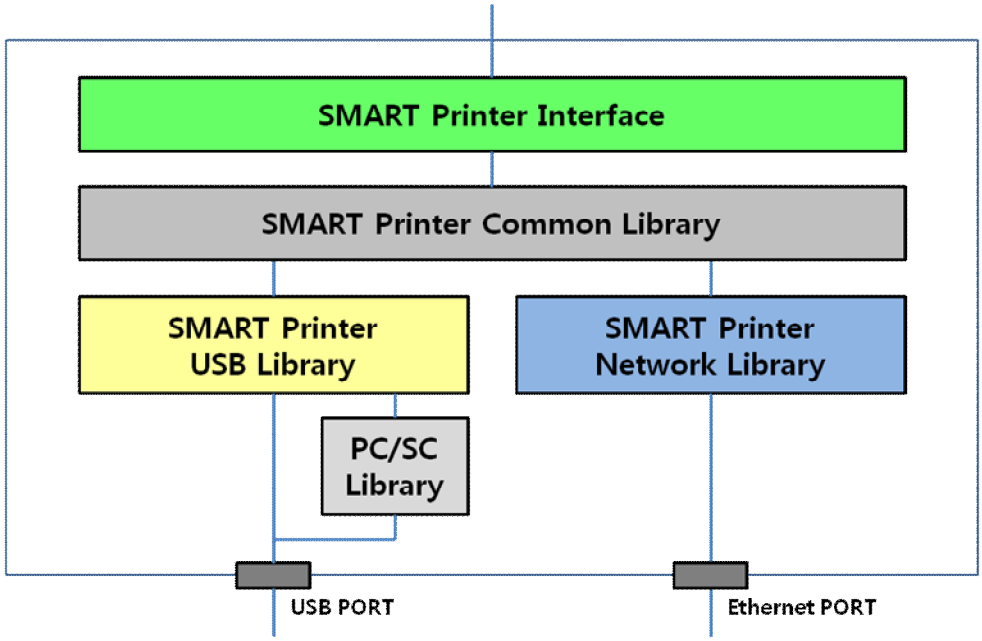

Figure 3. The Configuration of SMART Printer SDK

[Using Printer Driver]

"SMART Printer SDK" has two operation modes, Driver and DCL (Direct Command Language) modes. The Standard mode uses printer driver as Figure 4 (a) and DCL mode communicates with the printer without printer driver as Figure 4 (b). Almost functions provided in this SDK work on both operation mode, but some functions support one operation mode only. SmartComm_OpenDevice, SmartComm_OpenDevice2, SmartComm_Print and SmartComm_CloseDevice are used for Driver mode and SmartDCL_OpenDevice, SmartDCL_OpenDevice2, SmartDCL_Print and SmartDCL_CloseDevice are used for DCL mode. In case of using network module, only DCL mode is supported. Please refer to Function Reference.

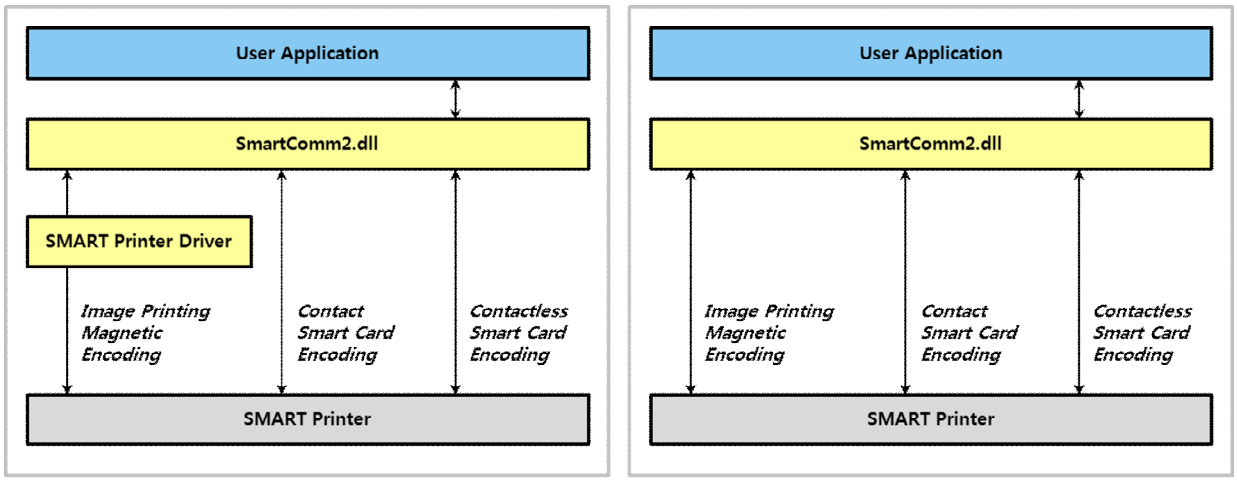

Figure 4. SMART Printer Operation Mode

[Printer Operating Modes]

SMART Printer is operated by two modes, Standard and SBS (Step By Step). Standard mode is the same as a general printer. In this mode, if printing data is sent, it prints an image immediately and ejects a card automatically.

SBS mode can be set by SDK function. If a printer is set as SBS mode, all operations can be controlled by step by step such as Card In, Card Out, Magnetic Stripe Read/Write, Contact IC Read/Write, Contactless IC Read/Write so it is so helpful to add other functions after the printing process.

The main difference between Standard mode and SBS mode is, after printing, Card Out function that it is done automatically or is done by another command. The default set of SMART Card printer is Standard mode and it is set as a Standard mode in the factory when it comes out.

2 Getting Started

2.1 Construction of SMART Printer SDK

2.1.1 SMART Printer Documents

/Documents/English/*

English user's manual for SMART Printer SDK

/Documents/Korean/*

Korean user's manual for SMART Printer SDK

2.1.2 SMART Printer Libraries

/Lib/SmartComm2.dll

Basic DLL to control local and network SMART Printer

/Lib/libcv.dll

/Lib/*.xml

Files for "Auto portrait" and "Auto Effect" functions when SmartComm2.dll refers CSD document.

2.1.3 SMART Printer Header Files

/Include/SmartComm2.dll.h

This file defines the structure and function of SmartComm2.

2.1.4 SMART Printer Reference Class

/Class/SmartComm2.h

C++ class header file to use SmartComm2.dll conveniently

/Class/SmartComm2.cpp

C++ class source file to use SmartComm2.dll conveniently

/Class/SmartComm2.dll.h

This file defines the structure and function of SmartComm2.dll.

2.1.5 SMART Printer Sample Code

/Sample/VC/*

VC++ Samples for controlling SMART Printer using SmartComm2.dll

/Sample/VB/*

VB6 Samples for controlling SMART Printer using SmartComm2.dll

/Sample/VB/*

VB.NET Samples for controlling SMART Printer using SmartComm2.dll

/Sample/JNI/*

Java (JNI) Samples for controlling SMART Printer using SmartComm2.dll

2.1.6 SMART Printer Drivers

/Drivers/Card Printer Device Driver/SMART/*

Drivers for SMART-50 and SMART-30 Printer

To install the driver, you should execute DDInstall.exe

/Drivers/Card Printer Device Driver/SMART51/*

Drivers for SMART-51 and SMART-31 Printer

To install the driver, you should execute DDInstall.exe

/Drivers/Contact Smart Card Reader/Gemalto PC Twin/*

Drivers for Gemalto contact smart card (IC).

/Drivers/Contactless Smart Card Reader/Omnikey 5121/*

Drivers for Omnikey contactless smart card (RF).

/Drivers/Contactless Smart Card Reader/DUALi DE-ABCM/*

Drivers for DUALi contactless smart card (RF).

2.1.7 SMART Firmwares

/Firmwares/Ethernet/smartnet2-fs-X.X.X.X.bin

Firmware for SMART Printer Ethernet Module's file system. You can update it using NetAdmin.exe.

/Firmwares/Ethernet/smartnet2-os-X.X.X.X.bin

Firmware for SMART Printer Ethernet Module's operating system. You can update it using NetAdmin.exe.

/Firmwares/Ethernet/smart-fs-X.X.X.X.bin

Firmware for SMART Printer old Ethernet Module's file system. You can update it using NetAdmin.exe. (This module is discontinued in 2015)

/Firmwares/Ethernet/smart-os-X.X.X.X.bin

Firmware for SMART Printer old Ethernet Module's operating system. You can update it using NetAdmin.exe. (This module is discontinued in 2015)

/Firmwares/Printer/50,30/SmartX_X_X.bin

Firmware for SMART-50 and SMART-30 Printer. You can update it using CardPrinterFirmware.exe.

/Firmwares/Printer/50,30/SL_X_X_X.bin

Firmware for SMART-50 Laminator Module. You can update it using CardPrinterFirmware.exe.

/Firmwares/Printer/51,31/Smart51_app_X_X_X_SPI.bin

Firmware for SMART-51 and SMART-31 Printer. You can update it using CardPrinterFirmware.exe.

/Firmwares/Printer/51,31/Smart51Laminator_X_X_X.bin

Firmware for SMART-51 Laminator Module. You can update it using CardPrinterFirmware.exe.

/Firmwares/Printer/51,31/Smart51Flipper_X_X_X.bin

Firmware for SMART-51 and SMART-31 Flipper Module. You can update it using CardPrinterFirmware.exe.

2.1.8 SMART Utilities

/Utilities/CardPrinterDiagnostic.exe

A program to check sensors and motors in SMART Printer.

/Utilities/CardPrinterFirmware.exe

A program for updating the firmware of SMART Printer

Firmware which is in “/Firmwares/Printer” folder will be updated to SMART Printer

/Utilities/CardPrinterInfo.exe

A Program for checking printer information

/Utilities/CardPrinterConfig.exe

A program to manage SMART Printer’s setting.

/Utilities/CardPrinterConfig.ini

An information file which is used in CardPrinterConfig.exe.

/Utilities/CardPrinterTest.exe

A program to test SMART Printer

/Utilities/CardPrinterTest.ini

An information file which is used in CardPrinterTest.exe.

/Utilities/NetAdmin.exe

A program to manage settings of ethernet module in the SMART printer.

/Utilities/SCardMonitor.exe

A program to view events of PC/SC service.

/Utilities/SmartComm2.dll

Basic DLL to control local and network SMART Printer.

/Utilities/viewprn.exe

A program to show .prn files.

2.2 Usage of Driver Mode

The basic way to program using SmartComm2.dll is as follows;

```
SMART_PRINTER_LIST list;
HSMART             smart;
SmartComm_GetDeviceList2(&list);                // Get list of Printer
SmartComm_OpenDevice2(&smart, list.item[0].desc,
                        SMART_OPENDEVICE_BYDESC); // Open first printer
SmartComm_SBSStart(smart);                      // Turn on SBS mode
//
// SmartComm_XXX();                             // Call  functions
//
SmartComm_SBSEnd(smart);                        // Turn off SBS mode
SmartComm_CloseDevice(smart);                   // Close Printer
```

To operate SMART Printer, bring the information of printers those are connected to local or network using SmartComm_GetDeviceList2. Description, printer ID, the name of the printer driver, and connection type(USB or Network) are included in the information. Printer ID is the string which can distinguish printers and the IDs of all printers are set as “SMART” when those are produced in a factory. In case of installing one Smart printer to PC, you don’t need to change the ID. However, if you want to use more than two SMART Printers, the string of ID should be changed. You can refer to the SMART Printer user manual to change the string of ID.

In case, more than two printers have the same printer ID on the network, you can use printer’s description instead of printer ID to open the device.

After you check the printer’s ID or description, open the printer using SmartComm_OpenDevice2. If SmartComm_OpenDevice2 succeeds, smart will not return NULL;

After opening the printer, use SmartComm_SBSStart function and the printer can be operated as SBS(Step-By-Step) mode.

After SmartComm_SBSStart starts, you can operate printer step by step. For example, to print a card, you can use commands such as CardIn, DrawImage, Print, DoPrint, CardOut.

Before finishing operation of SMART Printer, end SBS mode using SmartComm_SBSEnd and close printer using CloseDevice.

Please be sure, the printer driver is installed on your PC, before you use SmartComm_OpenDevice2 function to open a printer. Because the opening function refers the set values of the driver, so if the driver is not installed, there could be some errors when you try to operate printer using other functions. (If you want to operate the printer, not using printer driver, use DCL mode)

For the compatibility with SDK1, SmartComm_GetDeviceList, SmartComm_GetDeviceInfo, SmartComm_OpenDevice, SmartDCL_OpenDevice functions can be used as the same with SDK 1. Also SmartComm_GetDeviceList2, SmartComm_GetDeviceInfo2, SmartComm_OpenDevice2, SmartDCL_ OpenDevice2 are added in SDK 2 and this functions are almost the same as the previous functions but changed little bit.

For more detail information, refer to the Function Reference and Sample Code, please.

2.3 Usage of DCL Mode

The basic method to control SMART Printer using DCL mode is as follows.

```
SMART_PRINTER_LIST list;
HSMART             smart;
SmartComm_GetDeviceList2(&list);          // Get list of Printer
SmartDCL_OpenDevice2(&smart, list.item[0].desc,
                     SMART_OPENDEVICE_BYDESC, DMORIENT_LANDSCAPE );
                                          // Open first printer
SmartComm_SBSStart(smart);                // Turn on SBS mode
//
// SmartComm_XXX();                       // Call desired functions
// SmartDCL_XXX();                        // Call DCL functions
//
SmartComm_SBSEnd(smart);                  // Turn off SBS mode
SmartDCL_CloseDevice(smart);              // Close Printer
```

DCL mode usage is similar to the usage of driver mode.

First, get printer information (those are connected to USB or network), using SmartComm_GetDeviceList2. After you get the printer's ID and description, open the printer using SmarDCL_OpenDevice2. If SmartDCL_OpenDevice2 succeeds, the function will not return a NULL value.

After you succeed to open the printer, operate Step-By-Step mode using SmartComm_SBSStart.

After SmartComm_SBSStart start, you can operate various functions to the printer.

To finish operating the printer, exit Step-By-Step mode using SmartComm_SBSEnd and then use SmartDCL_CloseDevice to close the printer.

In DCL mode, Users can make printing data directly instead of using CSD file or printing with SmartComm_DrawXXX. Get the structure of SMART_SURFACE or SMART51_SURFACE of the front side and back side using SmartDCL_GetSurface or SmartDCL_GetSurface2. The SMART_SURFACE structure includes printing features and DC, therefore, users can draw printing data directly to DC.

Caution! When you make printing data using SmartComm_OpenDocument or SmartComm_DrawXXX, data in SMART_SURFACE or SMART51_SURFACE DC will be ignored.

For further information, refer to Function Reference and Sample Code.

Followings are the structure definition of SMART_SURFACE that used in SMART-50 and SMART-30.

```
typedef struct {
   DWORD        side;
   DWORD        orientation;
   DWORD        ribbon;
   DWORD        ribbon_type;
```

```
    DWORD        width;
    DWORD        height;
} SMART_SURFACE_PROPERTIES;

typedef struct {
    HDC          hdcColor;
    HDC          hdcResin;
    HDC          hdcFluorescent;
    HDC          hdcOverlay;
    HBITMAP      hBmpColor;
    HBITMAP      hBmpResin;
    HBITMAP      hBmpFluorescent;
    HBITMAP      hBmpOverlay;
    BYTE *       lpPRN;
} SMART_SURFACE_BITMAPS;

typedef struct {
    DWORD        dwCCMain;
    DWORD        dwCCYellow;
    DWORD        dwCCMagenta;
    DWORD        dwCCCyan;
    DWORD        dwCCBlack;
    DWORD        dwCCOverlay;
    DWORD        dwDocRibbonSplit;
    DWORD        dwDocFlip;

    DWORD        dwBPText;
    DWORD        dwBPDot;
    DWORD        dwBPThreshold;
    DWORD        dwBPDitherDegree;

    DWORD        dwEdgeSize;
    DWORD        dwEjectCard;
    DWORD        dwErase;
    DWORD        dwDocHeatControl;

    DWORD        dwMSTrack1;
    DWORD        dwMSTrack2;
    DWORD        dwMSTrack3;
    DWORD        dwJISTrack;

    DWORD        dwManualEncoding;
    WCHAR        strMSTrack1[1024];
    WCHAR        strMSTrack2[1024];
    WCHAR        strMSTrack3[1024];
    WCHAR        strJISTrack[1024];

    DWORD        dwDocSupply;
```

```
    DWORD       dwDocTray;
    DWORD       dwDocRibbon;
    DWORD       dwDocResin;
    DWORD       dwDocQuality;
    DWORD       dwDocColor;
    DWORD       dwDocDither;

    DWORD       dwDocPrintSide;
    DWORD       dwDocMediaFront;
    DWORD       dwDocMediaBack;
    DWORD       dwDocLaminateSide;
    DWORD       reserved;
    WCHAR       strDocMediaUserFront[1024];
    WCHAR       strDocMediaUserBack[1024];
} SMART_SURFACE_OEMDEV;

typedef struct  {
    SMART_SURFACE_PROPERTIES    prop;
    SMART_SURFACE_BITMAPS       bmp;
    SMART_SURFACE_OEMDEV        oemdev;
}  SMART_SURFACE;
```

SMART_SURFACE_PROPERTIES	

Side	Indicates printing sides information
	
	`#define PAGE_FRONT 0 // front`
	`#define PAGE_BACK 1 // back`

orientation	Indicates printing direction.
	Defined at SmartDCL_OpenDevice2()
	
	`#define DMORIENT_PORTRAIT 1 // Vertical`
	`#define DMORIENT_LANDSCAPE 2 // Horizontal`

ribbon	Defines which kind of ribbon is used.
ribbon_type	Indicates Standard / Premium ribbon.
width	Indicates horizontal card width The width is determined by orientation value when SmartDCL_OpenDevice2 function is called.
height	Indicates vertical card height. The height is determined by orientation value when SmartDCL_OpenDevice2 function is called.

SMART_SURFACE_BITMAPS	
hdcColor	DC for color panel printing
hdcResin	DC for resin panel printing
hdcFluorescent	DC for fluorescent printing
hdcOverlay	DC for overlay panel printing
hBmpColor	Bitmap for color panel printing
hBmpResin	Bitmap for resin panel printing
hBmpFluorescent	Bitmap for fluorescent printing
hBmpOverlay	Bitmap for overlay panel printing
lpPRN	Printing data buffer-treated printing format

SMART_SURFACE_OEMDEV	
dwCCMain	Controls the all color density to adjust printing quality. Available value is from -100 to 100. The bigger number is the higher its density.
dwCCYellow	Controls the yellow density. Available value is from -100 to 100. The bigger number is the higher its density.
dwCCMagenta	Controls the Magenta density. Available value is from -100 to 100. The bigger number is the higher its density.
dwCCCyan	Controls the Cyan density. Available value is from -100 to 100. The bigger number is the higher its density.
dwCCBlack	Controls the Resin Black density. Available value is from -100 to 100. The bigger number is the higher its density.
dwCCOverlay	Controls the Overlay density. Available value is from -100 to 100. The bigger number is the higher its density.
dwDocRibbonSplit	Sets the part of ribbon to print opposite side. This function is applied when ribbon type is YMC.K.O and dual sides are printed. YMC prints front side -> card rotates -> Black(K) prints on back side -> card rotates -> Overlay(0) coats on front side. If the value is“0”, it does not separate and if the value is “1”, it separates.
dwBPText	Defines which level of density will be extracted to resin when extract text with resin panel. The default is to extract genuine black text. The index is from 0 to 100. The bigger number is the higher density.
dwBPDot	Defines which level of density will be extracted to resin when extract dot with

	resin panel. The default is to extract genuine black text. The index is from 0 to 100. The bigger number is the higher density.
dwBPThreshold	Defines which level of color will be changed to black. The default value is “50”. The index is from 0 to 100. The bigger number is the higher density.
dwBPDitherDegree	Defines the level of dithering when black dithering is used. When the number is big, the degree of dispersion is big and the printing looks soft. When the number is small, the degree of dispersion is small and the printing looks rough. The index is from 0 to 100.
dwEdgeSize	Defines how wide the edge does not print when the “dwDocEdgeFront, dwDocEdgeBack”value is “Use”. The index is from 0 to 100 and the default value is 20.
dwEjectCard	Defines whether printed card ejects or not. If the value is “0”, the card will be ejected and the value is “1”, the card will not be ejected after printing.
dwErase	At rewritable printer, sets “Erase” dignity. The default is 50 and 0 to 100 value can be used.
dwDocHeatControl	Defines whether thermal printer head temperature is controlled or not. If the value is “0”, it is not controlled and if the value is “1”, it is controlled.

dwMSTrack1	Defines ISO type track 1 encoding mode.
	Defines which encoding mode will be used when printing and magnetic encoding do at the same time.
	There are 6 type encoding modes, that is Normal, Reverse, Bit, Normal HiCo, Reverse Bit, Bit HiCo.
	Normal : Encode input date with the normal way
	Reverse : Encode input date with the reverse way
	Bit : Treat input data as binary data(input 0 or 1)
	Hico means encode at HiCo card.
	
	`#define SMART_ENCODE_NORMAL 0x00`
	`#define SMART_ENCODE_REVERSE 0x01`
	`#define SMART_ENCODE_BIT 0x02`
	`#define SMART_ENCODE_HICO_NORM 0x03`
	`#define SMART_ENCODE_HICO_REV 0x04`
	`#define SMART_ENCODE_HICO_BIT 0x05`

dwMSTrack2	Defines ISO type track2 encoding mode.
dwMSTrack3	Defines ISO type track3 encoding mode.
dwJISTrack	Defines JIS II type encoding mode.
dwManualEncoding	dwManualEncoding is used to magnetic encoding using print data. If the magnetic related function is used, magnetic encoding and reading will be done directly independent of printing.

	0 – Not use 1 – Use
strMSTrack1	ISO type track1 encoding data
strMSTrack2	ISO type track2 encoding date
strMSTrack3	ISO type track3 encoding data
strJISTrack	JIS II type encoding data

dwDocSupply	Defines hopper and a fixed value is “0”.
	
	`#define SMART_SUPPLY_AUTO 0x00`
	`#define SMART_SUPPLY_HOPPER 0x01`

dwDocTray	`#define SMART_TRAY_CR80 0x00`

dwDocRibbon	Defines which kind of ribbon is used.
	dwDocRibbon is automatically set when you open the printer with SmartDCL_OpenDevice2 function. If you want to change that value, read ribbon type with SmartComm_GetRibbonType and set a new value.
	
	`#define SMART_RIBBON_YMCKO 0x00 // YMCKO`
	`#define SMART_RIBBON_YMCKOK 0x01 // YMCKO.K`
	`#define SMART_RIBBON_hYMCKO 0x02 // half YMCKO`
	`#define SMART_RIBBON_KO 0x03 // KO`
	`#define SMART_RIBBON_K 0x04 // K`
	`#define SMART_RIBBON_BO 0x05 // BO`
	`#define SMART_RIBBON_B 0x06 // B`
	`#define SMART_RIBBON_BYMCKO 0x07 // BYMCKO`
	`#define SMART_RIBBON_YMCKFO 0x08 // YMCKFO`
	`#define SMART_RIBBON_REWRITABLE 0x09 // Rewritable Printer`
	`#define SMART_RIBBON_hYMCKOKO 0x0B // half YMCKO.KO`

dwDocResin	Defines the method when a print color image, extract black with resin panel and print with high density. Resin extract method has Black Objects, Black Texts, Black Dots and Not Use.
	Black Objects: Extract black text and black drawing automatically and print with resin panel
	Black Texts: Extract black text and print with resin panel
	Black Dot : Extract all blacks among print data and print with resin panels
	Not Use: Do not use resin panel
	
	`#define SMART_RESIN_BLACKOBJECT 0x00`
	`#define SMART_RESIN_BLACKTEXT 0x01`
	`#define SMART_RESIN_BLACKTDOT 0x02`
	`#define SMART_RESIN_NOTUSE 0x03`

dwDocQuality	Defines printing quality.

	Standard: High quality
	Fast : A little bit lower quality than standard but fast printing speed.
	Partial : Print only image exists.
	Semi-Partial : Print from image starts to card end.
	We recommend using “Standard” mode to maintain high printing quality.
	
	`#define SMART_QUALITY_STANDARD 0x00`
	`#define SMART_QUALITY_VERYHIGH 0x01`
	`#define SMART_QUALITY_PARTIAL 0x02`
	`#define SMART_QUALITY_SEMIPARTIAL 0x03`

dwDocColor	Defines whether print color or black and white when the color ribbon is used.
	
	`#define SMART_COLOR_COLOR 0x00`
	`#define SMART_COLOR_BLACKNWHITE 0x01`

dwDocDither	Defines which dithering algorithm will be used when print black and white with resin panel.
	Threshold: Defines black and white with standard density.
	Random: Use random function during dithering
	Diffusion: Use diffusion algorithm
	We recommend diffusion dither.
	
	`#define SMART_DITHER_THRESHOLD 0x00`
	`#define SMART_DITHER_RANDOM 0x01`
	`#define SMART_DITHER_DIFFUSION 0x02`

dwDocPrintSide	Defines single side print or dual side print
	The value which comes from “SmartDCL_Print” function to print side parameter is saved. If flipper is not installed, Front is used and the flipper is installed, choose “Front” when print front side and choose “Both” when printing dual side.
	You can check if flipper is installed or not call “ SmartComm_GetStatus” function.
	
	`#define SMART_PRINTSIDE_FRONT 0x00`
	`#define SMART_PRINTSIDE_BOTH 0x01`

dwDocMediaFront	Defines where not to print according to card type.
dwDocMediaBack	Defines where not to print according to card type.

dwDocLaminateSide	Defines laminate print side. If a laminator is not installed in the printer this value is not used. To check a laminator is installed, call SmartComm_GetStatus function.
	
	`#define SMART_LAMINATESIDE_NONE 0x00`
	`#define SMART_LAMINATESIDE_TOP 0x01`
	`#define SMART_LAMINATESIDE_BOTTOM 0x02`
	`#define SMART_LAMINATESIDE_BOTH 0x03`

strDocMediaUserFront	In dwDocMediaFront, when user-defined printing area is used, mask file is designated.
strDocMediaUserBack	In strDocMediaUserBack, when user-defined printing area is used, mask file is designated.

Followings are the structure definition of SMART51_SURFACE that is used in SMART-51 and 31.

```
typedef struct {
      DWORD     side;
      DWORD     orientation;
      DWORD     ribbon;
      DWORD     ribbon_type;
      DWORD     width;
      DWORD     height;
}     SMART51_SURFACE_PROPERTIES;


typedef struct {
      HDC       hdcColor;
      HDC       hdcResin;
      HDC       hdcFluorescent;
      HDC       hdcOverlay;
      HBITMAP hBmpColor;
      HBITMAP hBmpResin;
      HBITMAP hBmpFluorescent;
      HBITMAP hBmpOverlay;
      BYTE *    lpPRN;
}     SMART51_SURFACE_BITMAPS;

typedef struct {
      DWORD     dwASMain;
      DWORD     dwASYellow;
      DWORD     dwASMagenta;
      DWORD     dwASCyan;
      DWORD     dwASBlack;
      DWORD     dwASOverlay;
      DWORD     dwASCorrectColor;
      DWORD     dwASCorrectMono;
      DWORD     dwASCorrectOverlay;
```

```
DWORD     dwASBPText;
DWORD     dwASBPDot;
DWORD     dwASBPThreshold;
DWORD     dwASBPDitherDegree;
DWORD     dwASBPResin;
DWORD     dwASBPResinOnly;
DWORD     dwASErase;
DWORD     dwASFastAlignment;

DWORD     dwASWaitRFUse;
DWORD     dwASWaitRFSide;
DWORD     dwASWaitRFPos;
DWORD     dwASWaitRFTime;
DWORD     dwASWaitICUse;
DWORD     dwASWaitICSide;
DWORD     dwASWaitICPos;
DWORD     dwASWaitICTime;
BYTE      btAsReserved[40];

DWORD     dwIOSupply;
DWORD     dwIOTray;

DWORD     dwPrtUse;
DWORD     dwPrtSide;
DWORD     dwPrtColorFront;
DWORD     dwPrtColorBack;
DWORD     dwPrtFlipFront;
DWORD     dwPrtFlipBack;
DWORD     dwPrtMediaFront;
DWORD     dwPrtMediaBack;
WCHAR     strPrtMediaUserFront[1024];
WCHAR     strPrtMediaUserBack[1024];
DWORD     dwPrtRibbon;
DWORD     dwPrtSpeed;
DWORD     dwPrtQuality;
DWORD     dwPrtDither;
DWORD     dwPrtHeatControl;
DWORD     dwPrtRibbonSplit;
BYTE      btPrtReserved[40];

DWORD     dwLamUse;
DWORD     dwLamSide;
DWORD     dwLamOverlay;
BYTE      btLamReserved[40];

DWORD     dwEncUse;
DWORD     dwEncSide;
DWORD     dwEncCoer;
DWORD     dwEncMagRepeat;
DWORD     dwEncMagFlipBefore;
DWORD     dwEncMagFlipAfter;
```

```
    DWORD    dwEncUseUserOption;
    DWORD    dwEncUserHeadValue;
    DWORD    dwEncFormat1;
    DWORD    dwEncFormat2;
    DWORD    dwEncFormat3;
    DWORD    dwEncFormatJ;
    DWORD    dwEncDensity1;
    DWORD    dwEncDensity2;
    DWORD    dwEncDensity3;
    DWORD    dwEncDensityJ;
    DWORD    dwEncOption1;
    DWORD    dwEncOption2;
    DWORD    dwEncOption3;
    DWORD    dwEncOptionJ;
    WCHAR    strEncTrack1[1024];
    WCHAR    strEncTrack2[1024];
    WCHAR    strEncTrack3[1024];
    WCHAR    strEncJISTrack[1024];
    WCHAR    strEncMSST1[10];
    WCHAR    strEncMSST2[10];
    WCHAR    strEncMSST3[10];
    WCHAR    strEncMSSTJ[10];
    WCHAR    strEncMSET1[10];
    WCHAR    strEncMSET2[10];
    WCHAR    strEncMSET3[10];
    WCHAR    strEncMSETJ[10];
    BYTE     btEncReserved[40];

    BYTE     reserved[100];
} SMART51_SURFACE_OEMDEV;

typedef struct {
    SMART51_SURFACE_PROPERTIES   prop;
    SMART51_SURFACE_BITMAPS      bmp;
    SMART51_SURFACE_OEMDEV       oemdev;
}   SMART51_SURFACE;
```

SMART51_SURFACE_PROPERTIES	

side	Indicate printing page information
	
	`#define PAGE_FRONT 0 // front`
	`#define PAGE_BACK 1 // back`

orientation	Indicate printing direction
	Define at SmartDCL_OpenDevice2()
	
	`#define DMORIENT_PORTRAIT 1 // vertical`
	`#define DMORIENT_LANDSCAPE 2 // horizontal`

ribbon	Defines which kind of ribbon is used.
ribbon_type	Indicates Standard / Premium ribbon.
width	Indicates horizontal card width The width is determined by orientation value when SmartDCL_OpenDevice2 function is called.
height	Indicates vertical card height. The height is determined by orientation value when SmartDCL_ OpenDevice2 function is called.

SMART51_SURFACE_BITMAPS	
hdcColor	DC for color panel printing
hdcResin	DC for resin panel printing
hdcFluorescent	DC for fluorescent printing
hdcOverlay	DC for overlay panel printing
hBmpColor	Bitmap for color panel printing
hBmpResin	Bitmap for resin panel printing
hBmpFluorescent	Bitmap for fluorescent printing
hBmpOverlay	Bitmap for overlay panel printing
lpPRN	Printing data buffer-treated printing format

SMART51_SURFACE_OEMDEV	
dwASMain	Controls the all color density to adjust printing quality. Available value is from -100 to 100. The bigger number is the higher its density.
dwASYellow	Controls the yellow density. Available value is from -100 to 100. The bigger number is the higher its density.
dwASMagenta	Controls the Magenta density. Available value is from -100 to 100. The bigger number is the higher its density.
dwASCyan	Controls the Cyan density. Available value is from -100 to 100. The bigger number is the higher its density.
dwASBlack	Controls the Resin Black density. Available value is from -100 to 100. The bigger number is the higher its density.
dwASOverlay	Controls the Overlay density.

	Available value is from -100 to 100. The bigger number is the higher its density.
dwASCorrectColor	Controls the Color Correction. Available value is from -32 to 32. The bigger number is the higher its thickness.
dwASCorrectMono	Controls the Resin Black Correction. Available value is from -32 to 32. The bigger number is the higher its thickness.
dwASCorrectOverlay	Controls the Overlay Correction. Available value is from -32 to 32. The bigger number is the higher its thickness.
dwASBPText	Defines which level of density will be extracted to resin when extract text with resin panel. The default is to extract genuine black text. The index is from 0 to 100. The bigger number is the higher density.
dwASBPDot	Defines which level of density will be extracted to resin when extract dot with resin panel. The default is to extract genuine black dot. The index is from 0 to 100. The bigger number is the higher density.
dwASBPThreshold	Defines which level of color will be changed to black. The default value is “50”. The index is from 0 to 100. The bigger number is the higher density.
dwASBPDitherDegree	Defines the level of dithering when black dithering is used. When the number is big, the degree of dispersion is big and the printing looks soft. When the number is small, the degree of dispersion is small and the printing looks rough. The index is from 0 to 100.

dwASBPResin	Defines the method when a print color image, extract black with resin panel and print with high density. Resin extract method has Black Objects, Black Texts, Black Dots and Not Use.
	Black Objects: Extract black text and black drawing automatically and print with resin panel
	Black Texts: Extract black text and print with resin panel
	Black Dot : Extract all blacks among print data and print with resin panels
	Not Use: Do not use resin panel
	`#define SMART_RESIN_OBJECT 0`
	`#define SMART_RESIN_TEXT 1`
	`#define SMART_RESIN_DOT 2`
	`#define SMART_RESIN_NOTUSE 3`

dwASBPResinOnly	Prints Black Dot by Resin only if the value of dwASBPResin is SMART_RESIN_DOT and dwASBPResinOnly sets 1.

dwASErase	Controls the Erase density in the Rewritable printer. Available value is from -100 to 100. The default value is “50”.
dwASFastAlignment	Not use.
dwASWaitRFUse	Not use.
dwASWaitRFSide	Not use.
dwASWaitRFPos	Not use.
dwASWaitRFTime	Not use.
dwASWaitICUse	Not use.
dwASWaitICSide	Not use.
dwASWaitICPos	Not use.
dwASWaitICTime	Not use.
dwASReserved[10]	Not use.
dwIOSupply	Options for card supply. `#define SMART_SUPPLY_AUTO 0`
dwDocTray	Options for card type. `#define SMART_TRAY_CR80 0`
dwPrtUse	Indicates printing or not 0 – Not print 1 – Print

dwPrtSide	Defines single side print or dual side print
	The value which comes from “SmartDCL_Print” function to print side parameter is saved. If flipper is not installed, Front is used and the flipper is installed, choose “Front” when print front side and choose “Both” when printing dual side.
	You can check if flipper is installed or not by calling “SmartComm_GetStatus” function.
	
	`#define SMART51_PRINTSIDE_FRONT 0`
	`#define SMART51_PRINTSIDE_BOTH 2`

dwPrtColorFront	Defines whether print Color or Black & White when printing on front side.
	`#define SMART_COLOR_COLOR 0`
	`#define SMART_COLOR_BLACKNWHITE 1`

dwPrtColorBack	Defines whether print Color or Black & White when printing on back side.
dwPrtFlipFront	Defines the method of flipping when printing on the front side. 0 – Do not flip 1 – Flip vertical 2 – Flip horizontal

	3 – Flip vertical & horizontal The default value is “0”
dwPrtFlipBack	Defines the method of flipping when printing on back side.

dwPrtMediaFront	Defines the media mask type when printing on the front side.
	The default value is “0”.
	0 – Standard card.
	1 - Smart card.
	2 - Smart card (Right Only).
	3 - ISO Magnetic Stripe card.
	4 - JIS Magnetic Stripe card.
	5 - Smart card + ISO Magnetic Stripe card
	6 - No Overlay.
	7 - User Defined.
	
	`#define SMART_MASK_STANDARD 0`
	`#define SMART_MASK_SMART 1`
	`#define SMART_MASK_SMARTRIGHT 2`
	`#define SMART_MASK_MSISO 3`
	`#define SMART_MASK_MSJIS 4`
	`#define SMART_MASK_SMARTMSJIS 5`
	`#define SMART_MASK_NOOVERLAY 6`
	`#define SMART_MASK_USERDEFINED 7`

dwPrtMediaBack	Defines the media mask type when printing on back side.
strPrtMediaUserFront	Defines the path of the user-defined mask file when the User Defined(7) option is selected in dwPrtMediaFront
strPrtMediaUserBack	Defines the path of the user-defined mask file when the User Defined(7) option is selected in dwPrtMediaBack

dwPrtRibbon	Defines the ribbon type
	dwDocRibbon is automatically set when you open the printer with SmartDCL_OpenDevice2 function. If you want to change that value, read ribbon type with SmartComm_GetRibbonType or SmartComm_GetRibbonInfo and set a new value.
	
	`#define SMART_RIBBON_YMCKO 0 // YMCKO`
	`#define SMART_RIBBON_YMCKOK 1 // YMCKO.K`
	`#define SMART_RIBBON_hYMCKO 2 // half YMCKO`
	`#define SMART_RIBBON_KO 3 // KO`
	`#define SMART_RIBBON_K 4 // K`
	`#define SMART_RIBBON_BO 5 // BO`
	`#define SMART_RIBBON_B 6 // B`
	`#define SMART_RIBBON_BYMCKO 7 // BYMCKO`

	#define SMART_RIBBON_YMCKFO 8 // YMCKFO #define SMART_RIBBON_REWRITABLE 9 // Rewritable Printer #define SMART_RIBBON_hYMCKOKO 11 // half YMCKO.KO #define SMART_RIBBON_YMCKOKR 12 // YMCKO.KR
dwPrtSpeed	Defines printing speed High Quality : Good quality but slower Normal : High speed but worse quality We recommend High Quality. #define SMART51_SPEED_HIGHQUALITY 0x00 #define SMART51_SPEED_STANDARD 0x01
dwPrtQuality	Defines the area of printing on the card Standard : All area over the card Partial : an area that the image only exists in the card #define SMART51_QUALITY_STANDARD 0x00 #define SMART51_QUALITY_PARTIAL 0x01
dwPrtDither	Defines which dithering algorithm will be used when print black and white with resin panel. Threshold: Defines black and white with standard density. Random: Use random function during dithering Diffusion: Use diffusion algorithm We recommend diffusion dither. #define SMART_DITHER_THRESHOLD 0 #define SMART_DITHER_RANDOM 1 #define SMART_DITHER_DIFFUSION 2 #define SMART_DITHER_HALFTONE 3 #define SMART_DITHER_HALFTONE2 4
dwPrtHeatControl	Defines whether use Heat Control option or not The default value is “1”. 0 – Not use 1 - Use
dwPrtReserved	Not use
dwLamUse	Defines whether laminate or not. If a laminator is not installed in the printer this value is “0”. To check a laminator is installed, call SmartComm_GetStatus function. 0 – Not laminate. 1 - Laminate.
dwLamSide	Defines laminate print side. #define SMART_LAMINATESIDE_NONE 0x00 #define SMART_LAMINATESIDE_TOP 0x01

	#define SMART_LAMINATESIDE_BOTTOM 0x02 #define SMART_LAMINATESIDE_BOTH 0x03
dwLamOverlay	Defines whether print overlay or not before laminating 0 – Not print overlay 1 – Print overlay
dwLamReserved	Not use
dwEncUse	Defines whether magnetic stripe encoding or not. To check a Magnetic encoder is installed, call SmartComm_GetStatus function. 0 – Not encode on the magnetic stripe 1 - Encode on the magnetic stripe
dwEncSide	Options for the position of Magnetic stripe encoder in the printer 0 - Bottom 1 - Top
dwEncCoer	Options for coercivity to encode Magnetic stripe #define SMART_MAG_LOCO 0 #define SMART_MAG_HICO 1 #define SMART_MAG_SPCO 2 #define SMART_MAG_AUTO 3
dwEncMagRepeat	Defines the retry count of encoding when encoding is failed. #define SMART_MAG_NOREPEAT 0 #define SMART_MAG_REPEAT1 1 #define SMART_MAG_REPEAT2 2 #define SMART_MAG_REPEAT3 3 #define SMART_MAG_REPEAT4 4
dwEncMagFlipBefore	Defines whether flip a card or not before encoding 0 – Not flip 1 - Flip.
dwEncMagFlipAfter	Defines whether flip a card or not after encoding 0 – Not flip 1 - Flip.
dwEncUseUserOption	Defines the current value (mAh) when encoding 0 – Default value 1 – Use the user defined value in dwEncUseUserHeadValue
dwEncUseUserHeadValue	Defines the current value (mAh) by user directly when dwEncUseUserOption is 1. The default value is “300”. The index is from 0 to 800.
dwEncFormat1	Defines Format on track 1. The default value is “0”.

	0 - ISO IATA Format
	1 - ISO ABA Format
	2 - ISO MINS Format
	3 - JIS II Format
	4 - BITS Mode Format
	
	`#define SMART_MAG_FORMAT_IATA 0`
	`#define SMART_MAG_FORMAT_ABA 1`
	`#define SMART_MAG_FORMAT_MINS 2`
	`#define SMART_MAG_FORMAT_JIS2 3`
	`#define SMART_MAG_FORMAT_BITS 4`

dwEncFormat2	Defines Format on track 2. Default value is “1”.
dwEncFormat3	Defines Format on track 3 Default value is “2”.
dwEncFormatJ	Defines Format on JIS track Default value is “3”
dwEncDensity1	Defines Writing Density on track 1 Default value is “210”
dwEncDensity2	Defines Writing Density on track 2 Default value is “75”
dwEncDensity3	Defines Writing Density on track 3 Default value is “210”
dwEncDensityJ	Defines Writing Density on JIS track Default value is “210”
dwEncOption1	Defines encoding type on track 1 Default value is “0” 0 - ISO 1 - ISO Ballys 2 2 - ISO Ballys 3
dwEncOption2	Defines encoding type on track 2 Default value is “0”
dwEncOption3	Defines encoding type on track 1 Default value is “0”
dwEncOptionJ	Defines encoding type on track 1 Default value is “0”
strEncTrack1	ISO type track1 encoding data
strEncTrack2	ISO type track2 encoding date

strEncTrack3	ISO type track3 encoding data
strEncJISTrack	JIS II type encoding data
strEncMSST1	Defines the start string when encoding on track 1 ISO type by using a text string
strEncMSST2	Defines the start string when encoding on track 2 ISO type by using a text string
strEncMSST3	Defines the start string when encoding on track 3 ISO type by using a text string
strEncMSSTJ	Defines the start string when encoding on track JIS type by using a text string
strEncMSET1	Defines the end string when encoding on track 1 ISO type by using a text string
strEncMSET2	Defines the end string when encoding on track 2 ISO type by using a text string
strEncMSET3	Defines the end string when encoding on track 3 ISO type by using a text string
strEncMSETJ	Defines the end string when encoding on track JIS type by using a text string
dwEncReserved[10]	Not use
Reserved[100]	Not use

3 Function Reference

3.1 Return Values

If the functions in “SMART Printer SDK” are operated successfully, it will return to 0 (SM_SUCCESS). If there is an error, it returns a negative number. Each error is defined as below in SmartComm2.dll.h.

```
#define     SM_SUCCESS                0           // Success
#define     SM_FAIL                   -1          // Generic Failure
#define     SM_F_FOUNDNODEV           0x80000000  // there is no device to use.
#define     SM_F_INVALIDDEVIDX        0x80000001  // index of device is out of bound.
#define     SM_F_INVALIDBUFPOINTER    0x80000002  // invalid buffer pointer. (may be null)
#define     SM_F_NOTEXISTDEV          0x80000003  // device is not exist or not connected
#define     SM_F_INVALIDPARAM         0x80000004  // invalid parameter value.
#define     SM_F_DEVOPENFAILED        0x80000005  // device open failed.
#define     SM_F_DEVIO                0x80000006  // device io operation is failed.
#define     SM_F_FOUNDNODRV           0x80000007  // not found driver or cannot acquire
                                                  // DEVMODE from driver.
#define     SM_F_INVALIDHANDLE        0x80000008  // invalid handle value.
#define     SM_F_CARDISINSIDE         0x80000009  // card is already inside of device.
#define     SM_F_NOCARDISINSIDE       0x8000000A  // no card is inside of device.
#define     SM_F_HOPPEREMPTY          0x8000000B  // no cards are in hopper.
#define     SM_F_NOCARDONBOTH         0x8000000C  // no card both hopper and inside.
#define     SM_F_WAITTIMEOUT          0x8000000D  // timeout occured while wait...
#define     SM_F_CASEOPEN             0x8000000E  // case is opened.
#define     SM_F_ERRORSTATUS          0x8000000F  // current status has error flag.
#define     SM_F_CARDIN               0x80000010  // card-in action is failed.
#define     SM_F_CARDOUT              0x80000011  // card-out action is failed.
#define     SM_F_CARDOUTBACK          0x80000012  // card-back-out action is failed.
#define     SM_F_MOVE2MAG             0x80000013  // card move (to magnetic) is failed.
#define     SM_F_MOVE2IC              0x80000014  // card move (to IC) is failed.
#define     SM_F_MOVE2RF              0x80000015  // card move (to RF) is failed.
#define     SM_F_MOVE2ROT             0x80000016  // card move (to Flipper) is failed.
#define     SM_F_MOVE2DEV             0x80000017  // card move (from Flipper) is failed.
#define     SM_F_MAGRW                0x80000018  // magnetic read/write is failed,
#define     SM_F_NOPRINTDATA          0x80000019  // printer failed to receive print data.
#define     SM_F_PRINT                0x8000001A  // print failed.
#define     SM_F_SEEKRIBBON           0x8000001B  // seek-ribbon is failed.
#define     SM_F_MOVERIBBON           0x8000001C  // move-ribbon is failed.
#define     SM_F_EMPTYRIBBON          0x8000001D  // ribbon is empty.
#define     SM_F_ICHUP                0x8000001E  // ic-head up failed.
#define     SM_F_ICHDN                0x8000001F  // ic-head down failed.
#define     SM_F_ROTTOP               0x80000020  // rotate to top is failed.
#define     SM_F_ROTBOTTOM            0x80000021  // rotate to bottom is failed.
#define     SM_F_REQNOMAGTRACK        0x80000022  // requested no magnetic track.
#define     SM_F_REQMULTIMAGTRACK     0x80000023  // requested multiple mag. tracks
#define     SM_F_FILENOTFOUND         0x80000024  // file not found.
#define     SM_F_FIELDNOTFOUND        0x80000025  // field is not exist.
#define     SM_F_IMAGELOAD            0x80000026  // failed to load image.
#define     SM_F_CREATEDC             0x80000027  // dc creation is failed.
#define     SM_F_VERIFYDRV            0x80000028  // driver verification is failed. may the
                                                  // driver is not ours.
#define     SM_F_SPOOLING             0x80000029  // failed to make spool data.
#define     SM_F_DEVNOTOPENED         0x8000002A  // command is requested while device is not opened.
#define     SM_F_USEDBYOTHER          0x8000002B  // usb is temporarily blocked by other.
#define     SM_F_SOCKETCREATE         0x8000002C  // socket creation failed.
#define     SM_F_SOCKETCONNECT        0x8000002D  // socket connection failed.
#define     SM_F_SSLINIT              0x8000002E  // SSL initialization failed.
#define     SM_F_SSLCREATE            0x8000002F  // SSL creation failed.
#define     SM_F_SSLCONNECT           0x80000030  // SSL connection is failed.
#define     SM_F_RESERVED             0x80000031  // host is already reserved status.
```

```
#define     SM_F_INVALIDSOCKET          0x80000032   // socket fd is invalid.
#define     SM_F_LESSSENDED             0x80000033   // packet is sended less than requested.
#define     SM_F_LESSRECVED             0x80000034   // packet is received less than requested
#define     SM_F_SOCKETERROR            0x80000035   // socket error occured.
#define     SM_F_INVALIDPACKET          0x80000036   // packet is not valid.
#define     SM_F_PACKETSEQDIFFER        0x80000037   // received packet sequence/id is not equal
#define     SM_F_PACKETFLAGNOREPLY      0x80000038   // reply flag is not setted on received packet.
#define     SM_F_PACKETFLAGHEADER       0x80000039   // sent packet header is incorrect.
#define     SM_F_PACKETFLAGARGUMENT     0x8000003A   // argument is not valid on sent packet.
#define     SM_F_PACKETFLAGEXE          0x8000003B   // execution error flag is setted.
#define     SM_F_PACKETFLAGBADCMD       0x8000003C   // bad command flag is setted.
#define     SM_F_PACKETFLAGINIT         0x8000003D   //
#define     SM_F_PACKETFLAGHANDLE       0x8000003E   // invalid handle is given.
#define     SM_F_FILEOPEN               0x8000003F   // file open failed...
#define     SM_F_FILEREAD               0x80000040   // read from fale is failed.
#define     SM_F_NOTSUPPORTYET          0x80000041   // not support yet...
#define     SM_F_INSUFFICIENTBUF        0x80000042   // insufficient buffer.

#define     SM_F_ICESTABLISH            0x80000043   // SCardEstablish failed.
#define     SM_F_ICLISTREADER           0x80000044   // SCardListReaders failed.
#define     SM_F_ICCONNECT              0x80000045   // SCardConnect failed.
#define     SM_F_ICGETSTATUS            0x80000046   // SCardStatus failed.
#define     SM_F_ICDISCONNECT           0x80000047   // SCardDisconenct failed.
#define     SM_F_ICRELEASE              0x80000048   // SCardReleaseContext failed.

#define     SM_F_INVALIDNAME            0x80000200   // invalid or not support barcode name.
#define     SM_F_OBJECTNOTFOUND         0x80000203   // object is not exist.
#define     SM_F_SUBDLLLOADFAILED       0x80000204   // sub-dll load failed...
#define     SM_F_CONFIGFILEFAILED       0x80000205   // configuration file load/save failed...
#define     SM_F_OVERHEATED             0x80000206   // thermal-head is overheated...
#define     SM_F_NOPRINTHREAD           0x80000207   // no thermal-head
#define     SM_F_LOWERVERSION           0x80000208   //
#define     SM_F_HIGHERVERSION          0x80000209   //
#define     SM_F_OLDSERVER              0x8000020A   //
#define     SM_F_VERSIONNOTMATCH        0x8000020B   // version mismatched.
#define     SM_F_CHANGEPASSWORD         0x8000020C   // changing root/user password is failed.
#define     SM_F_UNLOCK                 0x8000020D   // unlock is failed.
#define     SM_F_LOCK                   0x8000020E   // lock is failed.
#define     SM_F_FILESIZEZERO           0x80000220   // required file size is 0
#define     SM_F_DEPRECATED             0x80000221   // function is deprecated.
#define     SM_F_SERIALNORESPONSE       0x80000222   // not responding for serial command.

#define     SM_F_NETAUTHFAIL            0x80000300   // Network Authentification failed.
#define     SM_F_NETREBOOTERROR         0x80000301   // Network Reboot command failed.
#define     SM_F_NETRELOADERROR         0x80000302   // Network Reload command failed.
#define     SM_F_NETRESETERROR          0x80000303   // Network Reset command failed.
#define     SM_F_NETUPGRADEERROR        0x80000304   // Network upgrade command failed.
#define     SM_F_NETGETSYSCFGERROR      0x80000305   // Network Get System Config command failed.
#define     SM_F_NETSETSYSCFGERROR      0x80000306   // Network Set System Config command failed.
#define     SM_F_NETGETSVCCFGERROR      0x80000307   // Network Get Service Config command failed.
#define     SM_F_NETSETSVCCFGERROR      0x80000308   // Network Set Service Config command failed.
#define     SM_F_NETLISTUSERERROR       0x80000309   // Network List User command failed.
#define     SM_F_NETADDUSERERROR        0x8000030A   // Network Add User command failed.
#define     SM_F_NETDELUSERERROR        0x8000030B   // Network Delete User command failed.
#define     SM_F_NETPASSWDUSERERROR     0x8000030C   // Network Change User Password command failed.
```

3.2 SmartComm_GetDeviceList

SmartComm_GetDeviceList is a function which collects the SMART Printer lists. Using this function, printers' information is collected and users can input the information to use other functions such as

SmartComm_OpenDevice and SmartDCL_OpenDevice.

SmartComm_GetDeviceList is for SDK 1 only. To use all functions of SDK 2, please use SmartComm_GetDeviceList2.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetDeviceList(
        DEVICEINFO* pDevList
        );
```

Parameters

pDevList

[out] A pointer of DEVICEINFO structure which is for the list of available printers. If the function is operated successfully, the number of SMART Printers is delivered at “nID” and printers’ IDs are input at “ID” array.

```
#define PRINTER_ID_MAX      32

typedef struct    {
        int       nID;
        WCHAR     ID[PRINTER_ID_MAX][16];
} DEVICEINFO;
```

Return Values

If the function is operated successfully, SM_SUCCESS will be returned.

Remarks

It collects printer lists from both USB and network. It’ll take a few seconds to collect printer list on the network.

Printer ID’s factory default setting is “SMART”. So if you want to use several SMART Printers on same PC or same network, you have to set different Printer ID using CardPrinterConfig Utility. For the setting method, refer to the SMART user manual.

3.3 SmartComm_GetDeviceList2

SmartComm_GetDeviceList2 is a function which collects the SMART Printer lists. Using this function,

printers' information is collected and users can input the information to use other functions such as SmartComm_OpenDevice2 and SmartDCL_OpenDevice2.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetDeviceList2(
        SMART_PRINTER_LIST* pDevList
        );
```

Parameters

pDevList

[out] A pointer for SMART_PRINTER_LIST structure which is for the list of available printers. SMART_PRINTER_LIST is defined as follows and if the function is operated successfully, the number of SMART Printers is delivered at "n" and printers' information are input at "item".

```
#define MAX_SMART_PRINTER 32

typedef struct {
        WCHAR    name[128];              // printer name
        WCHAR    id[64];                 // printer ID
        WCHAR    dev[64];                // device connection
        WCHAR    desc[256];              // description
        int      pid;                    // USB product ID
} SMART_PRINTER_ITEM;

typedef struct  {
        int                     n;
        SMART_PRINTER_ITEM      item[MAX_SMART_PRINTER];
} SMART_PRINTER_LIST;
```

Return Values

If the function is operated successfully, SM_SUCCESS will be returned.

Remarks

It collects printer lists from both USB and network. It'll take a few seconds to collect printer list on the network.

Printer ID's factory default setting is "SMART". So if you want to use several SMART Printers on same PC or same network, you have to set different Printer ID using CardPrinterConfig Utility. For the setting method, refer to the SMART user manual.

3.4 SmartComm_GetDeviceInfo

SmartComm_GetDeviceInfo is used to get detail information of a printer.

SmartComm_GetDeviceInfo is for SDK 1 only. To use all functions of SDK 2, please use SmartComm_GetDeviceInfo2.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetDeviceInfo(
        WCHAR* szDevice,
        SMART_LOOKUP_ITEM* pDevInfo, psmart
        );
```

Parameters

szDevID

[in] A printer ID to get a printer's information. To get the ID, SmartComm_GetDeviceList or SmartComm_GetDeviceList2 can be used

pDevInfo

[out] *szDevID* is SMART_LOOKUP_ITEM structure for printer's IP and Port information.

```
typedef struct {
        WCHAR    version[64];              // version info.
        WCHAR    id[64];                   // printer id.
        WCHAR    ip[64];                   // network ip address.
        int      pid;                      //
        int      port;                     // network port.
        int      b_ssl;                    // whether use of SSL protocol.
        int      type;                     // server type.
} SMART_LOOKUP_ITEM;
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

The printer model can be distinguished by pid. The range of pid is as below.

SMART-50 : 0x3850 ~ 0x386F

SMART-30 : 0x3870 ~ 0x388F

SMART-51 : 0x3510 ~ 0x351F

SMART-31 : 0x3310 ~ 0x331F

3.5 SmartComm_GetDeviceInfo2

SmartComm_GetDeviceInfo2 is used to get detail information of a printer.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetDeviceInfo2(
        SMART_PRINTER_INFO* pDevInfo,
        WCHAR* szDevice,
        int nDevType
        );
```

Parameters

pDevInfo

[out] A pointer of SMART_PRINTER_INFO structure to get a printer's information.

```
typedef struct {
        WCHAR    port[64];              // usb port
        WCHAR    link[256];             // symbolic link of usb port
        int      is_bridge;             // Network module bridge
}       SMART_PRINTER_PORT_USB;

typedef struct {
        WCHAR    ver[64];               // version of network protocol
        WCHAR    ip[64];                // ip address
        int      port;                  // tcp port
        int      is_ssl;                // ssl protocol
}       SMART_PRINTER_PORT_NET;

typedef struct {
        WCHAR    name[128];             // printer name
        WCHAR    id[64];                // printer ID
        WCHAR    dev[64];               // device connection
        int      dev_type;              // 1=USB, 2=NET
        int      pid;                   // USB product ID
```

```
        SMART_PRINTER_PORT_USB      usb;
        SMART_PRINTER_PORT_NET      net;
    } SMART_PRINTER_STANDARD;

    typedef struct {
        int      is_dual;                  // dual printer
        WCHAR    ic1[64];                  // internal contact encoder
        WCHAR    ic2[64];                  // external contact SIM encoder
        WCHAR    rf1[64];                  // internal contactless encoder
        WCHAR    rf2[64];                  // external contactless encoder
    } SMART_PRINTER_OPTIONS;

    typedef struct {
        SMART_PRINTER_STANDARD      std;
        SMART_PRINTER_OPTIONS       opt;
    } SMART_PRINTER_INFO;
```

szDevice

[in] Printer's Description or ID.

This parameter can be brought by SmartComm_GetDeviceList2 function.

nDevType

[in] A parameter for distinguish for *szDevice* whether it is Printer Description or Printer ID.

```
#define SMART_OPENDEVICE_BYID       0    // szDevice is Printer ID
#define SMART_OPENDEVICE_BYDESC     1    // szDevice is Printer Description
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

The printer model can be distinguished by pid of SMART_PRINTER_STANDARD. The range of pid is as below.

SMART-50 : 0x3850 ~ 0x386F

SMART-30 : 0x3870 ~ 0x388F

SMART-51 : 0x3510 ~ 0x351F

SMART-31 : 0x3310 ~ 0x331F

3.6 SmartComm_OpenDevice

SmartComm_OpenDevice is an opening function to initialize and receive a handle of SMART Printer. Printer ID which is from SmartComm_GetDeviceList or SmartComm_GetDeviceList2 functions, can be

used in this function.

SmartComm_OpenDevice is for SDK 1 only. To use all functions of SDK 2, please use SmartComm_OpenDevice2.

Operation Mode: USB, Network, Driver

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_OpenDevice(
        HSMART* pHandle,
        WCHAR* szID
        );
```

Parameters

pHandle

[out] A handle for Smart printer. After SmartComm_OpenDevice function this parameter will be set to non-zero value.

szDevice

[in] Printer's ID from SmartComm_GetDeviceList function

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If a printer is connected to USB, the other process can access the printer but if the printer is connected to the network, the other process can't access the printer and can't open the printer.

SmartComm_OpenDevice can be operated successfully only if the printer driver is installed on PC. If the printer driver is not installed and the user tries to open the printer, unexpected errors can occur.

3.7 SmartComm_OpenDevice2

SmartComm_OpenDevice2 is an opening function to initialize and receive a handle to use SMART Printer. Either printer description or printer ID which those are from SmartComm_GetDeviceList2 function can be used in this function.

Operation Mode: USB, Network, Driver

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_OpenDevice2(
        HSMART* pHandle,
        WCHAR* szDevice,
        int  nDevType
        );
```

Parameters

pHandle

[out] A handle for Smart printer. After SmartComm_OpenDevice2 function this parameter will be set as non-zero value .

szDevice

[in] Printer's description or ID from SmartComm_GetDeviceList2 function

nDevType

[in] This parameter distinguishs wheter it is printer description or printer's ID of *szDevice*

```
#define SMART_OPENDEVICE_BYID       0    // szDevice is Printer ID
#define SMART_OPENDEVICE_BYDESC     1    // szDevice is Printer Description
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If a printer is connected to USB, the other process can access the printer but if the printer is connected to the network, the other process can't access the printer and can't open the printer.

SmartComm_OpenDevice2 can be operated successfully only if the printer driver is installed on PC. If the printer driver is not installed and the user tries to open the printer, unexpected errors can occur.

3.8 SmartDCL_OpenDevice

SmartDCL_OpenDevice is a function to initialize and receive the handle to use SMART Printer as DCL mode. Printer ID from SmartComm_GetDeviceList or SmartComm_GetDeviceList2 can be used in this function.

SmartDCL_OpenDevice is for SDK 1 only. To use all functions of SDK 2, please use SmartDCL_OpenDevice2.

Operation Mode: USB, Network, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartDCL_OpenDevice(
      HSMART* pHandle,
      CHAR* szID,
      int nOrientation
      );
```

Parameters

pHandle

[out] A handle for Smart printer. After SmartDCL_OpenDevice function this parameter will be set to non-zero value.

szDevice

[in] Printer's ID from SmartComm_GetDeviceList function

nOrientation

[in] Orientation of the paper

```
#define DMORIENT_PORTRAIT              1
#define DMORIENT_LANDSCAPE             2
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If the printer driver is not installed in PC, SmartDCL_OpenDevice can be used to open printer. If you use this function, you should use SmartDCL_Print and SmartDCL_CloseDevice instead of SmartComm_Print and SmartComm_Close.

3.9 SmartDCL_OpenDevice2

SmartDCL_OpenDevice2 is a function to initialize and receive the handle to use SMART Printer as DCL mode. Either printer description or printer ID those are from SmartComm_GetDeviceList2 function, can be used in this function.

Operation Mode: USB, Network, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartDCL_OpenDevice2(
        HSMART* pHandle,
        WCHAR* szDevice,
        int nDevType,
        int nOrientation
        );
```

Parameters

pHandle

[out] A handle for Smart printer. After SmartDCL_OpenDevice2 function this parameter will be set to non-zero value.

szDevice

[in] Printer's description or ID from SmartComm_GetDeviceList2 function

nDevType

[in] This parameter distinguishs printer description and ID of *szDevice*

```
#define SMART_OPENDEVICE_BYID       0    // szDevice is Printer ID
#define SMART_OPENDEVICE_BYDESC     1    // szDevice is Printer Description
```

nOrientation

[in] Orientation of the paper

```
#define DMORIENT_PORTRAIT                1
#define DMORIENT_LANDSCAPE       2
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If the printer driver is not installed in PC, SmartDCL_OpenDevice2 can be used to open printer. If you use this function, you should use SmartDCL_Print and SmartDCL_CloseDevice instead of SmartComm_Print and SmartComm_Close.

3.10 SmartDCL_OpenDevice3

SmartDCL_OpenDevice3 is a function to initialize and receive the handle to use SMART Printer as DCL mode. If the C class of IP address of the network SMART printer is different from PC or the printer can't be searched by SmartComm_GetDevice2, this function can be used.

Operation Mode: USB, Network, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartDCL_OpenDevice3(
        HSMART* pHandle,
        WCHAR* szDevice,
        int nPort,
        BOOL bSSL,
        int nOrientation
);
```

Parameters

pHandle

[out] A handle for Smart printer. After SmartDCL_OpenDevice3 function this parameter will be set to non-zero value.

szDevice

[in] Printer's description from SmartComm_GetDeviceList3 function

Format: [SMART : *_ip_address_*]

nPort

[in] Port of the printer

bSSL

[in] Port of the printer. This parameter distinguishes whether SSL is used or not.

nOrientation

[in] Orientation of the paper

```
#define DMORIENT_PORTRAIT               1
#define DMORIENT_LANDSCAPE              2
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.11 SmartComm_CloseDevice

SmartComm_CloseDevice is close command when the use of SMART Printer is finished.

Operation Mode: USB, Network, Driver

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_CloseDevice(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] SMART Printer's handle which is opened.

Return Values

If the value is executed normally, SM_SUCCESS will be returned.

Remarks

The received handle, after the success of SmartComm_OpenDevice function, should be returned by SmartComm_CloseDevice. Especially in case of using the network, the handle should be returned as soon as possible that the printer can be used by other users. If the handle isn't returned, the assigned memory when calling OpenDevice can't be returned.

3.12 SmartDCL_CloseDevice

SmarDCL_CloseDevice is the function to close a printer when a user opened the printer using SmartDCL_OpenDevice.

Operation Mode: USB, Network, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartDCL_CloseDevice(
        HSMART hHandle
```

```
);
```

Parameters

hHandle

[in] A printer handle which is created by SmartDCL_OpenDevice function.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

The received handle, after the success of SmartDCL_OpenDevice function, should be returned by SmartDCL_CloseDevice. Especially in case of using the network, the handle should be returned as soon as possible that the printer can be used by other users. If the handle isn't returned, the assigned memory when calling OpenDevice can't be returned.

3.13 SmartComm_GetStatus

SMART Printer's status can be checked by using SmartComm_GetStatus.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetStatus(
        HSMART hHandle,
        __int64* piStatus
        );
```

Parameters

hHandle

[in] A handle of the opened printer

piStatus

[out] 64-bit value of the printer's status.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SMART-50 and SMART-30 Printer's statuses are defined in SmartComm2.dll.h.

```
// MOTION-PART
#define    SMSC_M_CARDIN           0x0000000000000001     // Under card in
#define    SMSC_M_CARDOUT          0x0000000000000002     // Under card card
#define    SMSC_M_MOVE_PRINT       0x0000000000000004     // Move to print position
#define    SMSC_M_MOVE_PRN2ROT     0x0000000000000008     // Move to roator from printer
#define    SMSC_M_MOVE_ROT2PRN     0x0000000000000010     // Moe to pritner from flipper
#define    SMSC_M_MOVE_IC          0x0000000000000020     // Move to IC position
#define    SMSC_M_MOVE_RF          0x0000000000000040     // Move to contactless IC position
#define    SMSC_M_MOVE_MAG         0x0000000000000080     // Move to the magnetic stripe position
#define    SMSC_M_THUP             0x0000000000000100     // Under Thermal head up
#define    SMSC_M_THDOWN           0x0000000000000200     // Under Thermal head down
#define    SMSC_M_ICHUP            0x0000000000000400     // Under IC head(Contactor) up
#define    SMSC_M_ICHDOWN          0x0000000000000800     // Under IC head(Contactor) down
#define    SMSC_M_PRINT            0x0000000000001000     // Under printing
#define    SMSC_M_MAGRW            0x0000000000002000     // Under Mag read/write
#define    SMSC_M_SEEKRIBBON       0x0000000000004000     // Under ribbon finding
#define    SMSC_M_MOVERIBBON       0x0000000000008000     // Under ribbon moving
#define    SMSC_M_ROTATORTOP       0x0000000000010000     // Card front side rotate to upper
#define    SMSC_M_ROTATORBOTTOM    0x0000000000020000     // Card front side rotate to down
#define    SMSC_S_HOPPERHASCARD    0x0000000000040000     // Card is in hopper
#define    SMSC_S_THUP             0x0000000000080000     // Thermal head up state
#define    SMSC_S_CARDIN           0x0000000000100000     // Card checking by card in sensor
#define    SMSC_S_CARDOUT          0x0000000000200000     // Card checking by card out sensor
#define    SMSC_S_ROTATORTOP       0x0000000000400000     // Card front is upside in flipper
#define    SMSC_S_EQUIPROTATOR     0x0000000000800000     // Flipper installed
#define    SMSC_M_RECVPRINTDATA    0x0000000001000000     // Uner receiving the printer buffer
#define    SMSC_S_HASPRINTBUFFER   0x0000000002000000     // Unser saving the printing data buffer
#define    SMSC_M_SBSRUNNING       0x0000000004000000     // Under executing SBS command
#define    SMSC_S_SBSMODE          0x0000000008000000     // SBS mode state
#define    SMSC_S_CASEOPEN         0x0000000010000000     // Case open state
#define    SMSC_M_INIT             0x0000000020000000     // Under initialization
#define    SMSC_S_TESTMODE         0x0000000080000000     // Under test mode.
// ERROR-PART
#define    SMSC_F_CARDIN           0x0000000100000000     // Card in error
#define    SMSC_F_MOVETOPRINT      0x0000000200000000      // Card moving error to the printing pos.
#define    SMSC_F_CARDOUT          0x0000000400000000     // Card out error
#define    SMSC_F_MOVETOMAG        0x0000000800000000     // Card moving error to the mag. pos.
#define    SMSC_F_MOVETOIC         0x0000001000000000     // Card moving error to IC position
#define    SMSC_F_MOVETORF         0x0000002000000000     // Card moving error to contactless pos.
#define    SMSC_F_MOVETOROTATOR    0x0000004000000000     // Fail to move to flipper
#define    SMSC_F_MOVEFROMROTATOR 0x0000008000000000      // Fail to move to printer from flipper
#define    SMSC_F_THUP             0x0000010000000000     // Thermal head up fail
#define    SMSC_F_THDOWN           0x0000020000000000     // Thermal head down fail
#define    SMSC_F_ICHUP            0x0000040000000000     // IC head(Contactor) up fail
#define    SMSC_F_ICHDOWN          0x0000080000000000     // IC head(Contactor) down fail
#define    SMSC_F_ROTATORTOP       0x0000100000000000     // failed to rotate the card facing up
#define    SMSC_F_ROTATORBOTTOM    0x0000200000000000     // failed to rotate the card facing down
#define    SMSC_F_PRINT            0x0000400000000000     // Fail in printing
```

```
#define    SMSC_F_MAGRW            0x0000800000000000    // Magnetic Read/Write fail
#define    SMSC_E_SEEKRIBBON       0x0001000000000000    // Ribbon search error
#define    SMSC_E_MOVERIBBON       0x0002000000000000    // Ribbon moving error
#define    SMSC_E_NOTH             0x0004000000000000    // Thermal head uninstalled
#define    SMSC_E_THOVERHEAT       0x0008000000000000    // Thermal head over heated
#define    SMSC_E_EMPTYRIBBON      0x0010000000000000    // No ribbon
#define    SMSC_F_DATA             0x0020000000000000    // Printing data error
#define    SMSC_F_CARDBACKOUT      0x0040000000000000    // card back out fail...
#define    SMSC_F_CARDERASE        0x0080000000000000    // Card data removing fail.
#define    SMSC_F_INCORRECT_PW     0x0100000000000000    // Wrong password
#define    SMSC_F_MAGREADT1        0x0200000000000000    // Failed to read from Mag. Track 1.
#define    SMSC_F_MAGREADT2        0x0400000000000000    // Failed to read from Mag. Track 2.
#define    SMSC_F_MAGREADT3        0x0800000000000000    // Failed to read from Mag. Track 3.
#define    SMSC_F_LOCKED           0x1000000000000000    // Device is currently locked.
#define    SMSC_F_SPOOLFULL        0x2000000000000000    // Printer's spool is full
#define    SMSC_F_SET              0x4000000000000000    // Setsting fail
```

If printer firmware version is higher than 1.00.60 the defines are changed as below

```
Printer Firmware version is 1.00.59 or less
#define    SMSC_S_THUP             0x0000000000080000    // Thermal head up.
#define    SMSC_S_ROTATORTOP       0x0000000000400000    // Card top side in Flipper

Printer Firmware 1.00.60 or more.
#define    SMSC_S_CLEANWARNING     0x0000000000080000    // Cleaning Warning.
#define    SMSC_S_EQUIPLAMINATOR   0x0000000000400000    // Laminator is equiped.
```

To check whether the flipper is equipped or not in the printer please look up the flags as below.

```
Printer Firmware version is 1.00.59 or less
If "SMSC_S_EQUIPROTATOR" flag is activated, Flipper is equipped.
Printer Firmware 1.00.60 or more.
If "SMSC_S_EQUIPROTATOR" or "SMSC_S_EQUIPLAMINATOR" flag is activated, Flipper is equipped.
```

SMART-51 Printer's statuses are defined in SmartComm2.dll.h.

```
// MOTION-PART
#define    S51PS_M_CARDIN          0x0000000000000001    // inserting card
#define    S51PS_M_CARDMOVE        0x0000000000000002    // moving card
#define    S51PS_M_CARDMOVEEXT     0x0000000000000004    // moving card between external
#define    S51PS_M_CARDEJECT       0x0000000000000008    // ejecting card
#define    S51PS_M_THEADLIFT       0x0000000000000010    // lifting up/down thermal head
#define    S51PS_M_ICLIFT          0x0000000000000020    // lifting up/down ic connector
#define    S51PS_M_RIBBONSEARCH    0x0000000000000040    // searching ribbon
#define    S51PS_M_RIBBONWIND      0x0000000000000080    // winding ribbon
#define    S51PS_M_MAGNETIC        0x0000000000000100    // processing magnetic
#define    S51PS_M_PRINT           0x0000000000000200    // printing
#define    S51PS_M_INIT            0x0000000000000400    // initializing
#define    S51PS_S_CONNHOPPER      0x0000000000000800    // hopper connected
```

```
#define    S51PS_S_CONNICENCODEER  0x0000000000001000    // ic encoder connected
#define    S51PS_S_CONNMAGNETIC    0x0000000000002000    // magnetic encoder connected
#define    S51PS_S_CONNLAMINATOR   0x0000000000004000    // laminator connected
#define    S51PS_S_CONNFLIPPER     0x0000000000008000    // flipper connected
#define    S51PS_S_FLIPPERTOP      0x0000000000010000    // flipper is top sided
#define    S51PS_S_COVEROPENED     0x0000000000020000    // cover is opened
#define    S51PS_S_DETECTIN        0x0000000000040000    // detect a card from in sensor
#define    S51PS_S_DETECTOUT       0x0000000000080000    // detect a card from out sensor
#define    S51PS_S_CARDEMPTY       0x0000000000100000    // card empty
#define    S51PS_S_RECVPRINTDATA   0x0000000000200000    // receiving print data
#define    S51PS_S_HAVEPRINTDATA   0x0000000000400000    // having print data
#define    S51PS_S_NEEDCLEANING    0x0000000004000000    // need cleaning
#define    S51PS_S_SWLOCKED        0x0000000008000000    // system locked (sw)
#define    S51PS_S_HWLOCKED        0x0000000010000000    // system locked (hw)
#define    S51PS_M_SBSCOMMAND      0x0000000020000000    // doing SBS command
#define    S51PS_S_SBSMODE         0x0000000040000000    // under SBS mode
#define    S51PS_S_TESTMODE        0x0000000080000000    // test mode

// ERROR-PART
#define    S51PS_F_CARDIN          0x0000000100000000    // failed to insert card
#define    S51PS_F_CARDMOVE        0x0000000200000000    // failed to move card
#define    S51PS_F_CARDMOVEEXT     0x0000000400000000    // failed to move card between external
#define    S51PS_F_CARDEJECT       0x0000000800000000    // failed to eject card
#define    S51PS_F_THEADLIFT       0x0000001000000000    // failed to lift up/down thermal head
#define    S51PS_F_ICLIFT          0x0000002000000000    // failed to lift up/down ic connector
#define    S51PS_F_RIBBONSEARCH    0x0000004000000000    // failed to search ribbon
#define    S51PS_F_RIBBONWIND      0x0000008000000000    // failed to wind ribbon
#define    S51PS_F_MAGNETIC        0x0000010000000000    // failed to read/write magnetic
#define    S51PS_F_READMAGT1       0x0000020000000000    // failed to read magnetic track 1
#define    S51PS_F_READMAGT2       0x0000040000000000    // failed to read magnetic track 2
#define    S51PS_F_READMAGT3       0x0000080000000000    // failed to read magnetic track 3
#define    S51PS_F_PRINT           0x0000100000000000    // error from printing
#define    S51PS_E_INIT            0x0000200000000000    // error from initializing
#define    S51PS_E_CONNEXT         0x0000400000000000    // device connection error –connect failed
#define    S51PS_E_CONNLAMINATOR   0x0000800000000000    // device connection error -laminator
#define    S51PS_E_CONNFLIPPER     0x0001000000000000    // device connection error -flipper
#define    S51PS_E_RIBBON0         0x0002000000000000    // ribbon remain 0
#define    S51PS_E_NORIBBON        0x0004000000000000    // no ribbon
#define    S51PS_E_NOTHEAD         0x0008000000000000    // no thermal head
#define    S51PS_E_OVERHEAT        0x0100000000000000    // thermal head overheat
#define    S51PS_F_INVALIDPRINTDATA0x0200000000000000    // invalid printing data format
#define    S51PS_F_INVALIDPASSWORD 0x0400000000000000    // invalid password
#define    S51PS_F_SET             0x4000000000000000    // failed to set
#define    S51PS_F_SPOOLFULL       0x8000000000000000    // fulled spool pool
```

3.14 SmartComm_GetRibbonType

SmartComm_GetRibbonType is for checking ribbon type installed in SMART Printer.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetRibbonType(
        HSMART hHandle,
        int* pnRibbonType
        );
```

Parameters

hHandle

[in] A handle of SMART Printer

pnRibbonType

[out] Ribbon type which is installed in Printer

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SMART Printer's ribbons have RFID tag so the ribbon type will be recognized automatically. Ribbons types are defined in SmartComm2.dll.h.

```
#define SMART_RIBBON_YMCKO              0
#define SMART_RIBBON_YMCKOK             1
#define SMART_RIBBON_hYMCKO             2
#define SMART_RIBBON_KO                 3
#define SMART_RIBBON_K                  4
#define SMART_RIBBON_BO                 5
#define SMART_RIBBON_B                  6
#define SMART_RIBBON_BYMCKO             7
#define SMART_RIBBON_YMCKFO             8
#define SMART_RIBBON_REWRITABLE         9
#define SMART_RIBBON_hYMCKOKO           11
#define SMART_RIBBON_YMCKOKR            12
```

SMART Rewritable Printer directly heat on the surface of the rewritable card so it does not use ribbon. However, the function takes it as using SMART_RIBBON_REWIRTABLE ribbon.

3.15 SmartComm_GetRibbonRemain

SmartComm_GetRibbonRemain is for the checking remained length of ribbon.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetRibbonRemain(
        HSMART hHandle,
        int* pnRibbonRemain
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pnRibbonRemain

[out] Ribbon's remains installed in Printer.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

In SMART Printer, ribbon's remains can be managed by RFID tag system. If the ribbon's remains are not enough, it can be alarmed and if there are no available remains, change notice can be indicated

3.16 SmartComm_GetRibbonInfo

SmartComm_GetRibbonInfo is for the retrieving information of ribbon.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetRibbonInfo(
        HSMART hHandle,
        int* pnRibbonType,
        int* pnRibbonMax,
        int* pnRibbonRemain,
        int* pnRibbonGrade
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pnRibbonType

[out] A pointer of ribbon type in SMART Printer. It can be NULL.

pnRibbonMax

[out] A pointer of a max of ribbon in SMART Printer. It can be NULL.

pnRibbonRemain

[out] A pointer of the remain of ribbon in SMART Printer. It can be NULL.

pnRibbonGrade

[out] A pointer of the grade of ribbon in SMART Printer. It can be NULL.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.17 SmartComm_ClearStatus

SmartComm_ClearStatus clears error status up when there are errors in SMART Printer.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_ClearStatus(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

Before using SMART Printer, this can be used to initialize the printer.

3.18 SmartComm_Reboot

SmartComm_Reboot is for rebooting SMART Printer.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_Reboot(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

After executing SmartComm_Reboot, the printer will be rebooted. So should return the handle by executing SmartComm_CloseDevice or SmartDCL_CloseDevice and wait until the printer is booting and have to reopen. At this moment, when the printer is available using SmartComm_GetDeviceList2,

do SmartComm_OpenDevice2 or SmartDCL_OpenDevice2.

3.19 SmartComm_SetLCDText

SmartComm_SetLCDText is for displaying user ’s message to LCD window of SMART Printer.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 51

Definition

```
int SmartComm_SetLCDText(
      HSMART hHandle,
      int nType,
      int nLine,
      WCHAR* szText
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nType

[in] Indicate the message type to print. Standard SMART Printer use 0.

nLine

[in] Sets the message display location to LCD window.

szText

[in] The content of the message which is display to LCD window.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

2 x 16 characters are used for LCD in SMART Printer. However in special specification, graphic LCD can be supported. *nType* is to set the display type of LCD display. LCD type is defined in SmartComm2.dll.h.

```
#define     LCD_TYPE_CH              0       // character LCD : Big Sized Text (2X16)
```

```
#define     LCD_TYPE_GSMALL          1       // graphic LCD : Small Sized Text (8X21)
#define     LCD_TYPE_GBIG            2       // graphic LCD : Big Sized Text (4X16)
```

3.20 SmartComm_Beep

SmartComm_Beep is a function of alarming sound.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50

Definition

```
int SmartComm_Beep(
        HSMART hHandle,
        int nBeep
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nBeep

[in] Sets how many times sound.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

This function is for the customized model so it cannot be operated in the standard model.

3.21 SmartComm_SBSStart

SmartComm_SBSStart is for setting to SBS (Step-By-Step) mode when SMART Printer is in normal mode.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_SBSStart(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

This SBS mode is to control the operation of SMART Printer per each step. The main difference between SBS mode and Normal mode is that the printing data through spool is not preceded automatically but it is preceded by SmartComm_DoPrint command. And to prevent wrong printing, if receive SmartComm_SBSStart command, the data in spool will be removed. So if you want to develop the program using SMART Printer SDK, it is recommended to initialize by SmartComm_SBSStart after SmartComm_OpenDevice.

3.22 SmartComm_SBSEnd

SmartComm_SBSEnd is for converting to normal mode after finish SBS mode.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_SBSEnd(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If you want to use SMART Printer in SBS mode and close, it is recommended to convert to normal mode from SBS mode by SmartComm_SBSEnd. If you close without SBS mode finishing, SMART Printer is in SBS state so the printing data will be not proceeded and save at spool. Up to 3 printing data can be saved at the spool.

3.23 SmartComm_CardIn

SmartComm_CardIn is for inserting a card to inside of printer from hopper.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_CardIn(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.24 SmartComm_CardOut

SmartComm_CardOut is for sending a card from inside of printer to stacker.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_CardOut(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.25 SmartComm_CardOutBack

SmartComm_CardOutBack is for sending the card from inside of printer to back side of printer.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_CardOutBack(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

It will operate dual side printer (with Flipper)

3.26 SmartComm_CardOutBackAngle

SmartComm_CardOutBackAngle is for setting the angle of the card out to printer back side.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_CardOutBackAngle(
        HSMART hHandle,
        int angle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

angle

[out] The angle of Card-Out, Setting value is -127 to 127 °(degree)

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

This function is only available for SMART Printer, the flipper is installed.

3.27 SmartComm_Move

SmartComm_Move is for moving the card to someplace.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51,31

Definition

```
int SmartComm_Move(
        HSMART hHandle,
        int pos
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pos

[out] Nominates the card moving position.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

To proceed prinitng, magnetic stripe encoding, contact IC encoding, contactless IC encoding, a card should be placed each related place. Using SmartComm_Move, you can move the card to the place.

```
#define    CARDPOS_PRINT          0      // Print position
#define    CARDPOS_MAGNETIC       1      // Magnetic encoding position
#define    CARDPOS_IC             2      // Contact smart card encoding position
#define    CARDPOS_TOROT          4      // From main body to flipper
#define    CARDPOS_FROMROT        5      // From flipper to main body
#define    CARDPOS_RF2            6      // Contactless smart card encoding position
#define    CARDPOS_IC2            7      // Contact smart card encoding position
```

3.28 SmartComm_MoveFromIn

SmartComm_MoveFromIn is for moving a card to the specified place from In-Sensor.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30

Definition

```
int SmartComm_MoveFromIn(
      HSMART hHandle,
      int dist
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

dist

[out] The range of value is from -127 to +127, and the unit is mm.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.29 SmartComm_MoveFromOut

SmartComm_MoveFromOut is for moving a card to place from Out-Sensor.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30

Definition

```
int SmartComm_MoveFromOut(
      HSMART hHandle,
      int dist
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

dist

[out] Distances to move from Out-Sensor. The range of value is from -127 to +127, and the unit is mm.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.30 SmartComm_MoveFromRotateIn

SmartComm_MoveFromRotaterIn is for moving a card to place from flipper In sensor.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30

Definition

```
int SmartComm_MoveFromRotateIn(
        HSMART hHandle,
        int dist
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

dist

[out] Distances to move from Out-Sensor. The range of value is from -127 to +127, and the unit is mm.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

This function is only available for SMART Printer, the flipper is installed.

3.31 SmartComm_MoveSensor

SmartComm_MoveSensor is for moving a card to place from given sensor.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-51, 31

Definition

```
int SmartComm_MoveSensor(
        HSMART hHandle,
        int sensor,
        int accel,
        int dist,
        int speed
```

```
);
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

sensor

[in] Indicate the basic sensor when moving a card

```
#define SMART51_SENSOR_NONE       0
#define SMART51_SENSOR_CENTER     1
#define SMART51_SENSOR_REAR       2
#define SMART51_SENSOR_FLIPMOTOR  3
#define SMART51_SENSOR_FLIPCENTER 4
#define SMART51_SENSOR_FLIPREAR   5
#define SMART51_SENSOR_FLIPANGLE  6
```

Accel

[in] Indicate whether moving accelerated or moving at a constant velocity

```
#define SMART51_ACCEL_NOTUSE      0
#define SMART51_ACCEL_USE                 1
```

dist

[in] Indicate the basic sensor when moving a card The range of value is from -1,600 to 1,600, and the unit is 0.1 mm. If the sensor value is SMART51_SENSOR_FLIPANGLE, this value indicates the angle and the range of value is from -180° to 180°.

speed

[in] Indicate the speed when moving a card. The range of value is from 100 to 300, and the unit is mm/s.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If the sensor value is SMART51_SENSOR_FLIPMOTOR, SMART51_SENSOR_FLIPCENTER, SMART51_SENSOR_FLIPREAR or SMART51_SENSOR_FLIPANGLE, it is only available when the flipper is installed.

3.32 SmartComm_MovingScan

SmartComm_MovingScan is for scanning a card to move from flipper In sensor.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30

Definition

```
int SmartComm_MovingScan(
        HSMART hHandle,
        WORD dist,
        WORD speed,
        WORD speed2
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

dist

[in] Distances to move from In-Sensor of Flipper. The range of value is from 0 to 160, and the unit is mm.

speed

[in] Speeds to move from In-Sensor of Flipper. The range of value is from 100 to 1000, and the unit is PPS (pulse per second). In this speed value, the resolution of the pulse is 300. The formula which converts mm/s to PPS is following :

speed = 300 * mm/s / 25.4

speed2

[in] Speeds to move from In-Sensor of Flipper. The range of value is from 100 to 1000, and the unit is PPS (pulse per second). In this speed value, the resolution of the pulse is 225. The formula which converts mm/s to PPS is following :

speed = 225 * mm/s / 25.4

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.33 SmartCommEx_GetPanelDensity

SmartCommEx_GetPanelDensity is for getting a printing density value by ribbon panel.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_GetPanelDensity(
      HSMART hHandle,
      BYTE panel,
      int * pdensity
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

panel

[in] The panel value for getting printing density value.

pdensity

[out] The address of buffer for getting printing density.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.34 SmartCommEx_SetPanelDensity

SmartCommEx_SetPanelDensity is for setting a printing density value by ribbon panel.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_SetPanelDensity(
        HSMART hHandle,
        BYTE panel,
        int  density,
        WCHAR * pw
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

panel

[in] The panel value for getting printing density value.

density

[int] The printing density value.

The range of density value is different by the panel and its range is following.

Color Panel : -500 ~ +500

Black Panel : -1500 ~ +1000

Overlay Panel : -500 ~ +500

pw

[int] The administrator password for changing density value.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.35 SmartComm_ICHContact

SmartComm_ICHContact is for contacting IC module's head with IC of the card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_ICHContact(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SMART Printer supports 2 contact smart card encoders. One is the internal encoder for CR-80 size card and the other is external encoder using SIM slot on printer's front side. SmartComm_ICHContact is to move IC head to contact with IC of the card. In SIM, if it is inserted, it is contacted.

You need to execute SmartComm_ICHContact for physical contact before connect with contact smart card. And after disconnect with contact smart card, you have to execute SmartComm_ICHDiscontact to avoid the physical contact.

3.36 SmartComm_ICHDiscontact

SmartComm_ICHDiscontact is for moving IC head to not contact with IC of the card after finished IC encoding.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_ICHDiscontact(
        HSMART hVALUESHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If you use SmartComm_ICHContact to contact with IC of the card, you have to proceed SmartComm_ICHDiscontact to moving IC head from IC of the card. If not, it will be the reason of card moving problem or other problem.

3.37 SmartComm_Rotate

SmartComm_Rotate is for rotating a card. If receive SmartComm_Rotate command, the card is moving to flipper and it will be rotated 180 degrees and return to the inside of the printer.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_Rotate(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_Rotate is for only SMART Printer with flipper option. You can check the flipper option by call SmartComm_GerStatus and read the status value.

3.38 SmartComm_MagReadAction

SmartComm_MagReadAction is for reading magnetic stripe. If reading is succeeded, you have to operate SmartComm_MagGetBuffer to bring the data.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagReadAction(
        HSMART hHandle,
        int nTrackID
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nTrackID

[in] Nominate the reading track.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_MagReadAction is reading the magnetic data for the only nominated track. *nTrackID* is possible to read one or more tracks.

```
#define   MAG_T1     1        // ISO Track 1
#define   MAG_T2     2        // ISO Track 2
#define   MAG_T3     4        // ISO Track 3
#define   MAG_JIS    8        // JIS Track
```

3.39 SmartComm_MagReadAction2

SmartComm_MagReadAction2 is for reading magnetic stripe by normal but also bit mode. If the reading is succeeded, you have to operate SmartComm_MagGetBuffer to bring the data.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagReadAction2(
        HSMART hHandle,
        int nTrackID,
```

```
    int nOpt
    );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nTrackID

[in] Nominates the reading track.

nOpt

[in] Defines the option.

```
#define   MAG_NORMAL          0x00     // Normal mode
#define   MAG_BITMODE         0x80     // Bit Mode, OR operation with the nTrackID value
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_MagReadAction2, it is reading the magnetic data for the nominated track. *nTrackID* is possible to read one or more tracks.

```
#define   MAG_T1                    1        // ISO Track 1
#define   MAG_T2                    2        // ISO Track 2
#define   MAG_T3                    4        // ISO Track 3
#define   MAG_JIS                   8        // JIS Track
```

n0pt will be given as an option when you read.

When you read normal mode, it uses MAG_NORMAL.

The bit mode is only available on the ISO track.

3.40 SmartComm_MagGetBuffer

SmartComm_MagGetBuffer is for bringing the reading data which is executed in SmartComm_ MagReadAction.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagGetBuffer(
        HSMART hHandle,
        int nTrackID,
        BYTE* pOutBuf,
        int* pnInOutLen
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nTrackID

[in] Nominate the reading track.

pOutBuf

[out] Buffer to receive the reading data.

pnInOutLen

[in/out] Sets the buffer's size and return the number of reading data.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

Only one track information can be read in SmartComm_MagGetBuffer. To read more track information, have to use SmartComm_MagGetAllBuffer. If you read by bit mode, it will bring HexString data.

If JIS track is read, character array of WCHAR type will be returned at *pOutBuf*.

3.41 SmartComm_MagGetAllBuffer

SmartComm_MagGetAllBuffer is for bring all the reading data which is got by SmartComm_MagReadAction.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagGetAllBuffer(
        HSMART hHandle,
        BOOL bGetT1,
        BYTE* pBufT1,
        int* pnT1Len,
        BOOL bGetT2,
        BYTE* pBufT2,
        int* pnT2Len,
        BOOL bGetT3,
        BYTE* pBufT3,
        int* pnT3Len
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

bGetT1

[in] A parameter for reading Track 1. If it is TRUE, read.

pBufT1

[out] Track 1 Buffer to input data.

pnT1Len

[in/out] A parameter for a buffer size of Track1 and it is returned the number of reading data.

bGetT2

[in] A parameter for reading Track 2. If it is TRUE, read.

pBufT2

[out] Track 2 Buffer to input data.

pnT2Len

[in/out] A parameter for a buffer size of Track2 and it is returned the number of reading data.

bGetT3

[in] A parameter for reading Track 3. If it is TRUE, read.

pBufT3

[out] Track 3 Buffer to input data.

pnT3Len

[in/out] A parameter for a buffer size of Track3 and it is returned the number of reading data.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If you read in BitMode, it will bring Hex String data.

3.42 SmartComm_MagConfig

SmartComm_MagConfig is for setting the configuration values of magnetic stripe encoding such as format or density. Use this function before writing or reading Magnetic stripe.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-51, 31

Definition

```
int SmartComm_MagConfig(
        HSMART hHandle,
        MAGCONFIG* pMagCfg
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pMagCfg

[in] MAGCONFIG is the structure which defines the configuration value of magnetic encoding. MAGCONFIG is explained as below.

```
typedef  struct {
         int  ver;               // structure version (= MAGCONFIG_VER_1)
         struct {
              int  ballys_count; // 0 ~ 2    : 0(not use ballys)
              int  backward;     // forward(normal) backward(reverse)
              int  format;       // data format
              int  bpi;          // write bpi (bits per inch)
```

```
    } t1;
    struct {
        int  ballys_count;  // 0 ~ 2   : 0(not use ballys)
        int  backward;      // forward(normal) backward(reverse)
        int  format;        // data format
        int  bpi;           // write bpi (bits per inch)
    } t2;
    struct {
        int  ballys_count;  // 0 ~ 2   : 0(not use ballys)
        int  backward;      // forward(normal) backward(reverse)
        int  format;        // data format
        int  bpi;           // write bpi (bits per inch)
    } t3;
    struct {
        int  ballys_count;  // 0 ~ 2   : 0(not use ballys)
        int  backward;      // forward(normal) backward(reverse)
        int  format;        // data format
        int  bpi;           // write bpi (bits per inch)
    } jis;
} MAGCONFIG;
```

ver

Indicates the version of structure

```
#define   MAGCFG_VER_1              1
```

ballys_count

The retry count in case of BALLYS mode

```
#define   MAGCFG_BALLYS_NOTUSE      0
#define   MAGCFG_BALLYS_1           1
#define   MAGCFG_BALLYS_2           2
```

backward

The direction of writing

```
#define   MAGCFG_FORWARD            0
#define   MAGCFG_BACKWARD           1
```

format

The format on each track

```
#define   MAGCFG_FORMAT_IATA        0   // use on ISO 1 track generally
#define   MAGCFG_FORMAT_ABA         1   // use on ISO 2 track generally
#define   MAGCFG_FORMAT_MINS        2   // use on ISO 3 track generally .
#define   MAGCFG_FORMAT_JIS2        3   // use on JIS II track generally
#define   MAGCFG_FORMAT_BITS        4   // Bit Mode
```

bpi

Defines the density of writing

The value of ISO Track 1, 3 and JIS is 210, The value of ISO Track 2 is 75

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.43 SmartComm_MagWriteAction

SmartComm_MagWriteAction is for writing buffered data to the magnetic stripe. Before doing SmartComm_MagWriteAction, you have to proceed SmartComm_MagSetBuffer or SmartComm_MagSetAllBuffer to buffer the data.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagWriteAction(
      HSMART hHandle,
      int nTrackID,
      BOOL bHighCo
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nTrackID

[in] Nominates the writing track.

bHighCo

[in] Decides to encode to HiCo magnetic stripe. If it is TRUE, HiCo is used and FALSE is LoCo.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

LoCo magnetic stripe was used usually but nowadays, the damage can be happened frequently by

a magnet so it becomes changing to HiCo. SMART Printer supports HiCo / LoCo and it can be encoded by software. Please be sure that which type or magnetics you are going to use.

3.44 SmartComm_MagWriteAction2

SmartComm_MagWriteAction2 is for writing buffered data to the magnetic stripe by Bit Mode.

Before doing SmartComm_MagWriteAction, you have to proceed SmartComm_MagSetBuffer or SmartComm_MagSetAllBuffer to buffer the data.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagWriteAction2(
        HSMART hHandle,
        int nTrackID,
        int nOpt,
        int nT1BPI,
        int nT2BPI,
        int nT3BPI
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nTrackID

[in] Nominates writing track.

nOpt

[in] Decides encoding options of magnetic stripe.

```
#define   MAG_LOWCO              0x00     // Low-co write
#define   MAG_HIGHCO             0x10     // High-co write
#define   MAG_SUPERCO            0x20     // Super-co write, SMART-51,31 is used
#define   MAG_USERCO             0x30     // user defined coercivity to write
#define   MAG_BITMODE            0x80     // bit mode write, OR operation with the above value
```

nT1BPI

[in] Bit Mode recording, BPI will be used in ISO Track 1, if it is not Bit Mode, it will be ignored.

nT2BPI

[in] Bit Mode recording, BPI will be used in ISO Track 2, if it is not Bit Mode, it will be ignored.

nT3BPI

[in] Bit Mode recording, BPI will be used in ISO Track 3, if it is not Bit Mode, it will be ignored.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

LoCo magnetic stripe was used usually but nowadays, the damage can be happened frequently by a magnet so it becomes changing to HiCo. SMART Printer supports HiCo / LoCo and it can be encoded by software. Please be sure that which type or magnetics you are going to use. If you want to record data in Bit Mode, you can use MAG_BITMODE option at *nOpt.* You have to set BPI value for each track to Bit Mode Recording. You can acknowledge BPI value by SmartComm_MagBitModeGetBPI.

3.45 SmartComm_MagBitModeGetBPI

If you record with bit mode, SmartComm_MagBitModeGetBPI will bring accurate BPI.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagBitModeGetBPI(
        int nBPI,
        int* pNewBPI
        );
```

Parameters

nBPI

[in] BPI value.

pNewBPI

[out] *nBPI* value will be brought the accurate value for use.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If you record bit mode, BPI value will be used between 40 ~210. BPI value decides maximum data length. You can use SmartComm_MagBitModeGetMaxSize to know the Maximum bit data length.

3.46 SmartComm_MagBitModeGetMaxSize

If you record by bit mode, SmartComm_MagBitModeGetMaxSize will bring the maximum data length from BPI.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagBitModeGetMaxSize(
        int nBPI,
        int* pnMaxSize
        );
```

Parameters

nBPI

[in] BPI value.

pnMaxSize

[out] when you use the *nBPI,* it will bring maximum bit data length.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.47 SmartComm_MagSetBuffer

SmartComm_MagSetBuffer is for tranfering data which is used in SmartComm_MagWriteAction or SmartComm_MagWriteAction2.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagSetBuffer(
        HSMART hHandle,
        int nTrackID,
        BYTE* pInBuf,
        int nLen
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nTrackID

[in] Nominates the writing track.

pInBuf

[in] Buffer for sending data.

nLen

[in] Display the length to write data.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_MagSetBuffer is for sending the data to one track buffer. So if you want to use more tracks, use SmartComm_MagSetAllBuffer. When you record in Bit Mode, you arrange the bit value by Hex String without space.

When you try to write on JIS track, use WCHAR type for *pInBuf*.

3.48 SmartComm_MagSetAllBuffer

SmartComm_MagSetAllBuffer is for sending all track's data which is encoded SmartComm_ MagWriteAction.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagSetAllBuffer(
        HSMART hHandle,
        BOOL bSetsT1,
        BYTE* pBufT1,
        int nT1Len,
        BOOL bSetsT2,
        BYTE* pBufT2,
        int nT2Len,
        BOOL bSetsT3,
        BYTE* pBufT3,
        int nT3Len
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

bSetsT1

[in] Decides to send Track1 buffer. If it is TRUE, send.

pBufT1

[in] Track 1 Buffer to send the data.

pnT1Len

[in] Data length of track 1 which is transferred.

bSetsT2

[in] Decides to send Track2 buffer. If it is TRUE, send.

pBufT2

[in] Track 2 Buffer to send the data.

pnT2Len

[in] Data length of track 2 which is transferred.

bSetsT3

[in] Decides to send Track3 buffer. If it is TRUE, send.

pBufT3

[in] Track 3 Buffer to send the data.

pnT3Len

[in] Data length of track 3 which is transferred.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

When you record in Bit Mode, arrange the bit value by Hex String without space.

3.49 SmartComm_MagGetCryptoBuffer

SmartComm_MagGetCryptoBuffer is for bringing the data which is read at SmartComm_MagReadAction by using the AES128 encrypted algorithm.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagGetCryptoBuffer(
      HSMART hHandle,
      int nTrackID,
      BYTE* pOutBuf,
      int* pnInOutLen,
      char* pKey
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nTrackID

[in] Nominates reading track.

pOutBuf

[out] Buffer to input the reading data.

pnInOutLen

[in/out] Sets buffer size and return the number of reading data.

pKey

[in] Nominates encryption key.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

Only one track information can be read by SmartComm_MagGetCryptoBuffer. To read more track information, use SmartComm_MagGetAllCryptoBuffer. In case *pKey* is used and it's shorter than 16 bytes.

3.50 SmartComm_MagGetAllCryptoBuffer

SmartComm_MagGetAllCryptoBuffer is for bring all the encrypted track data which is read at SmartComm_MagReadAction by using AES128.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagGetAllCryptoBuffer(
        HSMART hHandle,
        BOOL bGetT1,
        BYTE* pBufT1,
        int* pnT1Len,
        BOOL bGetT2,
        BYTE* pBufT2,
        int* pnT2Len,
        BOOL bGetT3,
        BYTE* pBufT3,
        int* pnT3Len,
```

```
    char* pKey
    );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

bGetT1

[in] Decides whether read the Track1 buffer. If it is TRUE, read.

pBufT1

[out] Track 1 Buffer to input data.

pnT1Len

[in/out] Buffer size of Track1 and return the number of the reading data size.

bGetT2

[in] Decides whether read the Track2 buffer. If it is TRUE, read.

pBufT2

[out] Track 2 Buffer to input data.

pnT2Len

[in/out] Buffer size of Track2 and return the number of the reading data size.

bGetT3

[in] Decides whether read the Track3 buffer. If it is TRUE, read.

pBufT3

[out] Track 3 Buffer to input data.

pnT3Len

[in/out] Buffer size of Track3 and return the number of the reading data size.

pKey

[in] Nominates encryption key.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If *pKey* is lower than 16 bytes, NULL will be recorded after the data and if bigger, only use 16 bytes.

3.51 SmartComm_MagSetCryptoBuffer

SmartComm_MagSetCryptoBuffer is for sending encryped(AES128) data which is used at SmartComm_MagWriteAction

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagSetCryptoBuffer(
        HSMART hHandle,
        int nTrackID,
        BYTE* pInBuf,
        int nInLen,
        char* pKey
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nTrackID

[in] Nominates writing track.

pInBuf

[in] Buffer for writing data.

nLen

[in] Displays the number of writing data.

pKey

[in] Nominates encryption key.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

The data is transfer to only one track buffer at SmartComm_MagSetCryptoBuffer. If you want to use more tracks at the same time, use SmartComm_MagSetAllCryptoBuffer.

If *pKey* is smaller than 16 byte, NULL will be recorded and use it as a code key. And if it is bigger, only use 16 bytes.

3.52 SmartComm_MagSetAllCryptoBuffer

SmartComm_MagSetAllCryptoBuffer is for tranferring the encrypted(AES128) data which will be encoded at SmartComm_MagWriteAction.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_MagSetAllCryptoBuffer(
        HSMART hHandle,
        BOOL bSetsT1,
        BYTE* pBufT1,
        int nT1Len,
        BOOL bSetsT2,
        BYTE* pBufT2,
        int nT2Len,
        BOOL bSetsT3,
        BYTE* pBufT3,
        int nT3Len,
        char* pKey
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

bSetsT1

[in] Decides whether send Track1 buffer. If it is TRUE, send.

pBufT1

[in] Track 1 Buffer to send the data.

pnT1Len

[in] Data length of track 1.

bSetsT2

[in] Decides whether send Track2 buffer. If it is TRUE, send.

pBufT2

[in] Track 2 Buffer to send the data.

pnT2Len

[in] Data length of track 2.

bSetsT3

[in] Decides whether send Track3 buffer. If it is TRUE, send.

pBufT3

[in] Track 3 Buffer to send the data.

pnT3Len

[in] Data length of track 3.

pKey

[in] Nominate encryption key.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If *pKey* is shorter than 16 byte, NULL will be recorded after the data. If the bigger, it uses only 16 bytes as encryption key.

3.53 SmartComm_OpenDocument

SmartComm_OpenDocument is for printing CSD file which is made in Smart Design program provided.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_OpenDocument(
        HSMART hHandle,
        const WCHAR* szCSDName
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

szCSDName

[in] The name of CSD file

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

Smart Design has not only the function of the designing card but also has the function of synchronizing field s which is connected with DB (Database). So to set fields, SmartComm_OpenDocument is needed.

In DCL mode, if a CSD file is opened by SmartComm_OpenDocument or SmartComm_DrawXXX, SMART_SURFACE DC data which is modified by the user will be ignored.

3.54 SmartComm_CloseDocument

SmartComm_CloseDocument is for closing a CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_CloseDocument(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.55 SmartComm_GetFieldCount

SmartComm_GetFieldCount is for checking the field number from CSD file which is opened by SmartComm_OpenDocument.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetFieldCount(
      HSMART hHandle,
      int* pnCount
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pnCount

[out] Returns the field number which is defined in CSD file.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.56 SmartComm_GetFieldName

SmartComm_GetFieldName is for bring the nominated field from CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetFieldName(
        HSMART hHandle,
        int idx,
        WCHAR* szName,
        DWORD dwBufLen
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

idx

[in] Field name's index in CSD

szName

[out] Buffer which the nominated field's name will be returned in.

dwBufLen

[in] Buffer size for the field name.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.57 SmartComm_GetFieldValue

SmartComm_GetFieldValue is for bring the nominated field value from CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetFieldValue(
        HSMART hHandle,
        const WCHAR* szName,
        WCHAR* szValue
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

szName

[in] A field name.

szValue

[out] Buffer for the name of the field.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.58 SmartComm_SetFieldValue

SmartComm_SetFieldValue is for nominating the field value in CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_SetFieldValue(
        HSMART hHandle,
        const WCHAR* szName,
        const WCHAR* szValue
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

szName

[in] Field's name which is to be set.

szValue

[in] Setting value for the name field.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.59 SmartComm_GetPrinterSettings

SmartComm_GetPrinterSetsting is for bringing the printing setup value of SMART Printer. The printing setup value is including all value which can be changed in the printer properties of Windows control board.

Operation Mode: USB, Network, Driver

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetPrinterSetstings(
        HSMART hHandle,
        CHASM_DEVMODE* pDm
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pDM

[out] A pointer for SMART Printer's printing properties.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

CHASM_DEVMODE is the structure which defines SMART printer's properties. CHASM_DEVMODE is explained at SmartComm2.dll.h. For each detailed explanation, please, refer to the device driver manual.

3.60 SmartComm_GetPrinterSettings2

SmartComm_GetPrinterSetsting2 is for bringing the printing setup value of SMART Printer. The printing setup value is including all value which can be changed in the printer properties of Windows control board.

Operation Mode: USB, Network, Driver

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetPrinterSettings2(
        HSMART hHandle,
        void* pDm,
        int* plen
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pDM

[out] A pointer for SMART Printer's printing properties.

CHASM_DEVMODE structure is used in the SMART-50, 30 and SMART51_DEVMODE structure is used in the SMART-51. The size of structure for the current printer is returned to plen if this value is NULL.

plen

[in] the size of pDM structure

[out] The size of structure for the current printer is returned if this value is NULL.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

CHASM_DEVMODE is the structure which defines SMART-50, 30 printer's properties.

SMART51_DEVMODE is the structure which defines SMART-51 printer's properties. If this CHASM_DEVMODE and SMART51_DEVMODE are explained at SmartComm2.dll.h. For each detailed explanation, please, refer to the device driver manual.

3.61 SmartComm_SetPrinterSettings

SmartComm_SetPrinterSetsting is for setting the printer setup value. The printing setup value is including all value which can be changed in the printer's properties of Windows control board.

Operation Mode: USB, Network, Driver

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_SetPrinterSetstings(
        HSMART hHandle,
        CHASM_DEVMODE* pDm
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pDM

[in] A pointer to the printing properties of SMART Printer.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.62 SmartComm_SetPrinterSettings2

SmartComm_SetPrinterSetsting2 is for setting the printer setup value. The printing setup value is including all value which can be changed in the printer's properties of Windows control board.

Operation Mode: USB, Network, Driver

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_SetPrinterSettings2(
        HSMART hHandle,
        void* pDm,
        int len
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pDM

[in] A pointer for the printing properties of SMART Printer.

CHASM_DEVMODE structure is used in the SMART-50, 30 and SMART51_DEVMODE structure is used in the SMART-51. The size of structure for the current printer is returned to plen if this value is NULL.

len

[in] the size of pDM structure

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.63 SmartComm_DrawImage

SmartComm_DrawImage is for printing an image directly not using the CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_DrawImage(
        HSMART hHandle,
```

```
BYTE page,
BYTE panel,
int x,
int y,
int cx,
int cy,
WCHAR* szImgPath,
RECT* prcArea
);
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] Sets the side printed. (Front : 0, Back : 1)

panel

[in] Sets the image's printing section among color, resin, overlay.

x

[in] Image's X-axis starting position. Left side is 0 and max is 1027.

y

[in] Image's Y-axis starting position. Up side is 0 and max is 635.

cx

[in] Width of image scaling. 0 is image's original size.

cy

[in] Height of image scaling. 0 is original size.

szImgPath

[in] Sets the image file's location.

prcArea

[out] Return value of printed area where the image is drawn on a card.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

In case of using SmartComm_DrawImage in local, the image file is used directly. but if it is used in the network, the image file will be transferred and print it. Panel to print is defined in SmartComm2.dll.h as below.

```
#define   PANELID_COLOR            1        // YMC color panel
#define   PANELID_BLACK            2        // Resin black  panel
#define   PANELID_OVERLAY          4        // Overlay panel
#define   PANELID_UV               8        // UV panel
```

In DCL Mode, if a CSD file is opened by SmartComm_OpenDocument or print data is added by SmartComm_DrawXXX, the data which is drawn in SMART SURFACE DC will be ignored.

3.64 SmartComm_DrawBitmap

SmartComm_DrawBitmap is the function of drawing images directly not using CSD files and it uses HBITMAP.

Operation Mode: USB, Network, Driver, DCL
Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_DrawBitmap(
        HSMART hHandle,
        BYTE page,
        BYTE panel,
        int x,
        int y,
        int cx,
        int cy,
        HBITMAP hbmp,
        RECT* prcArea
        );
```

Parameters

hHandle

[in] A handle of an opened printer.

page

[in] Sets the printed side. (Front : 0, Back : 1)

panel

[in] Sets the image's printing type among color, resin, overlay.

x

[in] Image's X-axis starting position. Left side is 0 and max is 1027.

y

[in] Image's Y-axis starting position. Up side is 0 and max is 635.

cx

[in] It means the width when the image is scaling. 0 is image's original size.

cy

[in] It means the height when the image is scaling. 0 is original size.

hbmp

[in] A handle of the bitmap which will be printed

prcArea

[out] Return value of printed area where the image is drawn on a card.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

The print panel is defined in SmartComm2.dll.h as followings.

```
#define   PANELID_COLOR             1        // YMC color panel
#define   PANELID_BLACK             2        // Resin black  panel
#define   PANELID_OVERLAY           4        // Overlay panel
#define   PANELID_UV                8        // UV panel
```

If a CSD file is opened using SmartComm_OpenDocument in DCL mode, or data is added by SmartComm_DrawXXX function then DC data, which is drawn by the user, in SMART_SURFACE will be ignored.

3.65 SmartCommEx_DrawImage

SmartCommEx_DrawImage is for printing the image directly not using the CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_DrawImage(
        HSMART hHandle,
        BYTE page,
        BYTE panel,
        int x,
        int y,
        int cx,
        int cy,
        int nScaleMethod,
        BYTE nAlign,
        WCHAR* szImgPath,
        RECT* prcArea
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] Sets the printed side. (Front : 0, Back : 1)

panel

[in] Sets the image's printing type among color, resin, overlay.

x

[in] Image's X-axis starting position. Left side is 0 and max is 1027.

y

[in] Image's Y-axis starting position. Up side is 0 and max is 635.

cx

[in] It means the width when the image is scaling. 0 is image's original size.

cy

[in] It means the height when the image is scaling. 0 is original size.

nScaleMethod

[in] Decides to image maximize/minimize in the frame.

```
#define   IMGSCALE_FITHORZ     0x00    // hold the width x vertical rate and range will be
                                           matched cx.
#define   IMGSCALE_FITVERT     0x01    // hold width x vertical rate and height will be
                                           matched cy.
#define   IMGSCALE_FITFRAME    0x02    // without hold the size, range and height will be
                                           matched cx,cy.
```

nAlign

[in] Decides the image alignment in the frame.

```
#define   OBJ_ALIGN_LEFT     0x00   // left alignment, match with left side in the frame.
#define   OBJ_ALIGN_CENTER   0x01   // center alignment, match with center in the frame.
#define   OBJ_ALIGN_RIGHT    0x02   // right alignment, match with right side in the frame.
#define   OBJ_ALIGN_TOP      0x00   // top alignment, match with upper side in the frame.
#define   OBJ_ALIGN_MIDDLE   0x10   // middle alignment, match with middle in the frame.
#define   OBJ_ALIGN_BOTTOM   0x20   // bottom alignment, match with bottom in theframe.
```

szImgPath

[in] Sets the image file's location.

prcArea

[out] Return value of printed area where the image is drawn on a card.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

Panel to print is defined in SmartComm2.dll.h.

```
#define   PANELID_COLOR              1        // Color panel which use YMC
#define   PANELID_BLACK              2        // Black panel which use Resin
#define   PANELID_OVERLAY            4        // Transparency panel which use Overlay
#define   PANELID_UV                 8        // Transparency panel which use Fluorescent
```

When a CSD file is opened by SmartComm_OpenDocument under DCL mode or is added some printing data with SmartComm_Draw*XXX*, the data in DC of SMART_SURFACE which is drawn by user will be ignored.

3.66 SmartCommEx_DrawBitmap

SmartCommEx_DrawBitmap is for printing images not using the CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_DrawBitmap(
        HSMART hHandle,
        BYTE page,
        BYTE panel,
        int x,
        int y,
        int cx,
        int cy,
        int nScaleMethod,
        BYTE nAlign,
        HBITMAP hbmp,
        RECT* prcArea
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] Decides the text printing side. (Front : 0, Back : 1)

panel

[in] Decides the text printing type among color, resin, overlay.

x

[in] Text's X-axis starting position. Left side is 0 and max is 1027.

y

[in] Text's Y axis starting position. Up side is 0 and max is 635.

cx

[in] Width when the image is scaling. 0 = original size.

cy

[in] Height when the image is scaling. 0 = original size.

nScaleMethod

[in] Decides to image maximize/minimize in the frame.

```
#define   IMGSCALE_FITHORZ     0x00    // hold width - vertical rates and width is
                                           matched to cx.
#define   IMGSCALE_FITVERT     0x01    // hold width - vertical rates and height is
                                           matched to cy.
#define   IMGSCALE_FITFRAME    0x02    // without hold the size, range and height will be
                                           matched cx,cy.
```

nAlign

[in] Decides the image alignment in the frame.

```
#define   OBJ_ALIGN_LEFT     0x00   // left alignment, match with left side in the frame.
#define   OBJ_ALIGN_CENTER   0x01   // center alignment, match with center in the frame.
#define   OBJ_ALIGN_RIGHT    0x02    // right alignment, match with right side in the frame.
#define   OBJ_ALIGN_TOP      0x00   // top alignment, match with upper side in the frame.
#define   OBJ_ALIGN_MIDDLE   0x10   // middle alignment, match with middle in the frame.
#define   OBJ_ALIGN_BOTTOM   0x20   // bottom alignment, match with bottom in theframe.
```

hbmp

[in] A handle of BMP.

prcArea

[out] Return value of printed area where the image is drawn on a card.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

Panel to print is defined in SmartComm2.dll.h.

```
#define   PANELID_COLOR              1        // Color panel which use YMC
#define   PANELID_BLACK              2        // Black panel which use Resin
#define   PANELID_OVERLAY            4        // Transparency panel which use Overlay
#define   PANELID_UV                 8        // Transparency panel which use Fluorescent
```

When a CSD file is opened by SmartComm_OpenDocument under DCL mode or is added some printing data with SmartComm_Draw*XXX*, the data in DC of SMART_SURFACE which is drawn by user will be ignored.

3.67 SmartComm_DrawText

SmartComm_DrawText is for printing text without using CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_DrawText(
        HSMART hHandle,
        BYTE page,
        BYTE panel,
        int x,
        int y,
        WCHAR* szFontName,
        int nFontSize,
        BYTE nFontStyle,
        WCHAR* szText,
        RECT* prcArea
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] Decides the text printing side. (Front : 0, Back : 1)

panel

[in] Decides the text printing type among color, resin, overlay.

x

[in] Text's X-axis starting position. Left side is 0 and max is 1027.

y

[in] Text's Y axis starting position. Up side is 0 and max is 635.

szFontName

[in] Font name for the text printing.

szFontSize

[in] Font size for the text printing.

nFontStyle

[in] Font type for the text printing.

szText

[in] The text contents to print.

prcArea

[out] Return value of printed area where the image is drawn on a card.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.68 SmartComm_DrawText2

SmartComm_DrawText2 is for printing the text without using CSD file. It has more option than SmartComm_DrawText.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_DrawText2(
        HSMART hHandle,
        BYTE page,
        BYTE panel,
        DRAWTEXT2INFO* pdt2info,
        WCHAR* szText
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] Decides the side printed. (Front : 0, Back : 1)

panel

[in] Decides the printing type among color, resin or overlay.

Pd12into

[in] Records the specification of Text.

```
typedef struct {
    int x;
    int y;
    int cx;
    int cy;
    int rotate;
    int align;
    int fontHeight;
    int fontWidth;
    int style;
    COLORREF color;
    int option;
    WCHAR szFaceName[32];
} DRAWTEXT2INFO;
```

x, y

[in] It is the initial coordinate of the frame that Text will be drawn.

cx, cy

[in] It is width and height of the text frame that will be drawn.
The text will be displayed in the frame only.

rotate

[in] It will set the rotation angle of text.
The value must be one of the “0, 90, 180, 270” align

align

[in] Defines the method of text alignment in the frame.

```
#define    OBJ_ALIGN_LEFT       0x00    // width-left alignment.
#define    OBJ_ALIGN_CENTER     0x01    // width-cemter alignment.
#define    OBJ_ALIGN_RIGHT      0x02    // Width-right alignment.
#define    OBJ_ALIGN_JUSTIFY    0x03    // width-justfy alignment.
#define    OBJ_ALIGN_TOP        0x00    // length-top alignment.
#define    OBJ_ALIGN_MIDDLE     0x10    // length- middle alignment.
#define    OBJ_ALIGN_BOTTOM     0x20    // length-bottom alignment.
```

Width and length alignment will be defined by bit OR operation.

When you set the OBJ_ALIGN_JUSTIFY, all text will be written in the frame. But it could be lapped.

fontHeight

[in] Defines the text height.

Fontwidth

[in] Defines the text width. When you set the “0”, length will be defined according to fontHeight.

It can be used for the different length of the text.

style

[in] Defines the text style.

```
#define   FONT_NORMAL       0x00
#define   FONT_BOLD         0x01
#define   FONT_ITALIC       0x02
#define   FONT_UNDERLINE    0x04
```

Each style can be duplicated by bit OR operation.

color

[in] Defines the text color. If the panel is defined as PANELID_BLACK, it will be printed with dithering. If the panel is defined as PANELID_COLOR, it will be printed with defined color.

option

[in] Defines text option.

```
#define   TEXT_NOFIT        0       // drawing normal
#define   TEXT_AUTOSIZE     4       // text autosizing for frame.
```

If you use TEXT_AUTOSIZE, the text will be resized according to the frame. *fontHeight* and *fontWidth* will be ignored.

SzFaceName

[in] Defines the text font name.

szText

[in] Contents of text.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

In DCL Mode, if a CSD file is opened by SmartComm_OpenDocument or data is added by SmartComm_Draw*XXX*, the data which is drawn in SMART SURFACE DC will be ignored.

3.69 SmartComm_DrawRect, 31

SmartComm_DrawRect is for printing rectangle without using CSD files.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51

Definition

```
int SmartComm_DrawRect(
        HSMART hHandle,
        BYTE page,
        BYTE panel,
        int x,
        int y,
        int cx,
        int cy,
        COLORREF col,
        RECT* prcArea
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] Decides the side of printing rectangle. (Front : 0, Back : 1)

panel

[in] Decides printing type among color, resin, overlay.

x

[in] Rectangle's X axis starting position. Left side is 0 and max is 1027.

y

[in] Text's Y axis starting position. Up side is 0 and max is 635.

cx

[in] Width of the rectangle.

cy

[in] Height of rectangle.

col

[in] Color of the rectangle.

prcArea

[out] Return value where the rectangle is printed on the card.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

In DCL Mode, if a CSD file is opened by SmartComm_OpenDocument or print data is added by SmartComm_Draw*XXX*, the data which is drawn in SMART SURFACE DC will be ignored.

3.70 SmartComm_DrawBarcode

SmartComm_DrawBarcode is for printing barcodes without using CSD files.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_DrawBarcode(
        HSMART hHandle,
        BYTE page,
        BYTE panel,
        int x,
        int y,
        int cx,
        int cy,
        COLORREF col,
        RECT* prcArea,
        const WCHAR* szName,
        int nSize,
        const WCHAR* szData,
        const WCHAR* szPost
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] Decides printing side. (Front : 0, Back : 1)

panel

[in] Decides printing area among color, resin, overlay.

x

[in] Rectangle's X axis starting position. Left side is 0 and max is 1027.

y

[in] Text's Y axis starting position. Up side is 0 and max is 635.

cx

[in] Width of the rectangle.

cy

[in] Height of rectangle.

col

[in] Color of the rectangle.

prcArea

[out] Return value where the rectangle is printed on the card.

szName

[in] A name of the barcode.

nSize

[in] Size of barcode.

szData

[in] Data of barcode.

szPost

[in] Post data of barcode. If the *szName* is “MaxiCode” it is postcode data, otherwise using NULL.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

In DCL Mode, if a CSD file is opened by SmartComm_OpenDocument or print data is added by SmartComm_Draw*XXX*, the data which is drawn in SMART SURFACE DC will be ignored.

SMART Printer SDK supports both 1D and 2D Barcodes.

// 1D Barcode	// 2D Barcode
"Code39(1:2)"	"PDF 417"
"Code39(1:3)"	"MicroPDF 417"
"Code39Ext(1:2)"	"DataMatrix"
"Code39Ext(1:3)"	"QR Code"
"EAN8"	"Maxicode"
"EAN13"	
"ITF"	
"Code128(A)"	
"Code128(B)"	
"Code128(C)"	
“Codabar”	

3.71 SmartComm_GetBarcodeTypeCount

SmartComm_GetBarcodeTypeCount is for checking the number of barcodes those are supported by DLL.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetBarcodeTypeCount(
        HSMART hHandle,
        int* pnCount
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pnCount

[out] The number of Barcodes type those are supported through DLL.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.72 SmartComm_GetBarcodeTypeName

SmartComm_GetBarcodeTypeName is for checking the barcode type.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetBarcodeTypeName(
        HSMART hHandle,
        int idx,
        WCHAR* szName,
        int nBufLen
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

idx

[in] Index of a barcode.

szName

[out] Barcode type.

nBufLen

[in] Buffer size of szName.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.73 SmartDCL_GetSurface

SmartDCL_GetSurface is a function to get a pointer of the SMART_SURFACE structure. You can draw directly on the DC for the panel(color, resin, overlay, fluorescence).

Operation Mode: USB, Network, DCL

Operation Device: SMART-50, 30

Definition

```
int SmartDCL_GetSurface(
        HSMART hHandle,
        SMART_SURFACE** ppFrontSurface,
        SMART_SURFACE** ppBackSurface
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

ppFrontSurface

[out] A pointer of the front side page.

ppBackSurface

[out] A pointer of the back side page

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.74 SmartDCL_GetSurface2

SmartDCL_GetSurface2 is a function to get a pointer of SMART_SURFACE or SMART51_SURFACE structure. You can draw directly on the DC for the panel(color, resin, overlay, fluorescence).

Operation Mode: USB, Network, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartDCL_GetSurface2(
      HSMART hHandle,
      int page,
      void** ppSurface,
      int* plen
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] A page that is returned.

ppSurface

[out] A pointer that the page indicates

A pointer of the SMART_SURFACE structure is returned in case of the SMART-50, 30 and

A pointer of the SMART51_SURFACE structure is returned in case of the SMART-51, 31.

plen

[in] the size of the structure that ppSurface indicates

[out] The size of structure for the current printer is returned if this value is NULL.

A size of the SMART_SURFACE structure is returned in case of the SMART-50, 30 and

A size of the SMART51_SURFACE structure is returned in case of the SMART-51, 31.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.75 SmartComm_Print

SmartComm_Print is for sending the printing data to the spool.

Operation Mode: USB, Network, Driver

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_Print(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If SmartComm_Print is executed when SMART Printer is the normal mode, a card is taken inside and is printed and is sent out as the same as a general program. However, in SBS mode, it will only be transferred to the spool of SMART Printer. To print in SBS mode, SmartComm_DoPrint command should be followed.

3.76 SmartDCL_Print

SmartDCL_Print is for sending the printing data to the printer.

Operation Mode: USB, Network, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartDCL_Print(
        HSMART hHandle,
        int  nPrintSide
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nPrintSide

[in] Print side.

```
#define   SMART_PRINTSIDE_FRONT    0        // Print front side only.
#define   SMART_PRINTSIDE_BOTH     1        // Print both side.

In case of SMART-51 and SMART-31, use the below values
#define   SMART51_PRINTSIDE_FRONT  0        // Print front side only.
#define   SMART51_PRINTSIDE_BOTH   2        // Print both side.
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If SmartComm_Print is executed when SMART Printer is the normal mode, a card is taken inside and is printed and is sent out as the same as a general program. However, in SBS mode, it will only be transferred to the spool of SMART Printer. To print in SBS mode, SmartComm_DoPrint command should be followed. If a CSD file is opened by SmartComm_OpenDocument or print data is added by SmartComm_Draw*XXX*, the data which is drawn in SMART SURFACE DC will be ignored.

3.77 SmartComm_DoPrint

SmartComm_DoPrint is for printing the data in SBS mode.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_DoPrint(
        HSMART hHandle
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_DoPrint is used in SBS mode only. If this command is sent, it will be printed on a card but card out operation will not be done.

3.78 SmartComm_GetPreviewBitmap

SmartComm_GetPreviewBitmap is for bringing bitmap data as DIB type which is created by SmartComm_OpenDocument or SmartComm_DrawText command.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetPreviewBitmap(
        HSMART hHandle,
        BYTE page,
        BITMAPINFO ** const ppbi
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

page

[in] Sets a side of a card. (0 : forward, 1 : back)

ppbi

[out] A bitmap data pointer in DIB format.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

ppbi delivers a bitmap pointer from internal of DLL. Do not delete *ppbi* after using it, because it is controlled by DLL.

3.79 SmartComm_GetUnitInfo

SmartComm_GetUnitInfo is for getting the object's data(text, barcode, image) from linked Field.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetUnitInfo(
        HSMART hHandle,
        UNITINFO* pUnit,
        int dir
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pUnit

[out] A pointer of UNITINFO structure to get object's data.

dir

[in] Sets the direction to get data. If it is UNITINFO_FIRST, it searches drawing an object from the first part of list linked to the field. If it is UNITINFO_NEXT, it searches drawing an object from the next part where the object was brought the last time.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

```
// unit type definition
#define   UNITTYPE_IMAGE            0x0010
#define   UNITTYPE_TEXT             0x0020
#define   UNITTYPE_BARCODE          0x0040

// unit info direction
#define   UNITINFO_FIRST            0
#define   UNITINFO_NEXT             1

// text/image alignment
#define   OBJ_ALIGN_LEFT            0x00
#define   OBJ_ALIGN_CENTER          0x01
#define   OBJ_ALIGN_RIGHT           0x02
#define   OBJ_ALIGN_JUSTIFY 0x03
#define   OBJ_ALIGN_HNOALIGN0x04
#define   OBJ_ALIGN_TOP             0x00
#define   OBJ_ALIGN_MIDDLE          0x10
#define   OBJ_ALIGN_BOTTOM          0x20
#define   OBJ_ALIGN_VNOALIGN0x30
#define   OBJ_ALIGN_HORZMASK0x0F
#define   OBJ_ALIGN_VERTMASK0xF0

// font information of Text
typedef   struct _tagFontInfo
{
      int          size;             // font size
      BYTE         style;            // font style
      COLORREF     color;            // font color
      WCHAR        name[LF_FACESIZE];// font name
} FontInfo;

// border information
typedef struct _tagBorder
{
      USHORT       type;             // using PenStyle. PS_SOLID, ...
      USHORT       width;            // line thickness. mm type.
      COLORREF     color;            // border color of frame.
```

```
} Border;

// background information
typedef   struct _tagBackGround
{
      BOOL         fill;             // fill background of frame ?
      COLORREF     color;            // fill color.
      BYTE         transparency;     // not use.
} BackGround;

// text object information.
typedef   struct
{
      short        leftMargin;       // left side space of inside of text object
      short        topMargin;        // top side space
      short        rightMargin;      // right side space
      short        bottomMargin;     // bottom side space
      BYTE         align;            // text horizontal and vertical alignment.
      FontInfo     font;
      WCHAR        field[33];        // linked field name.
} UNITTEXT;

// image object information.
typedef   struct
{
      int          widthZoom;        // horizontal zoom rate. 10000 multiplied value.
      int          heightZoom;       // vertical zoom rate. 10000 multiplied value.
      short        contrast;// contrast value.
      short        brightness;       // brightness value.
      BOOL         grayscale;        // is grayscaled ?
      USHORT       align;            // image alignment on frame.
      POINT        offset;           // start position of draw.
      BYTE         scaleMethod;      // how to scale
      int          round;            // rounding value.
      WCHAR        field[33];        // linked field name
} UNITIMAGE;

// barcode object information.
typedef   struct
{
      int          type;             // barcode type index
      int          size;             // barcode size
      int          option;           // option
      SIZE         opt2D;            // 2D barcode option
      COLORREF     barColor;// barcode color
      WCHAR        field[33];        // linked field name
} UNITBAR;
```

```
// object unit information.
typedef   struct
{
      int          index;            // zero-based index from list.
      int          type;             // object type. (UNITTYPE_TEXT, …)
      BYTE         page;             // page information.
      BYTE         panel;            // panel information.
      int          left;             // left position of frame.
      int          top;              // top position of frame.
      int          width;            // width of frame.
      int          height;           // height of frame.
      int          rotate;           // rotation value.
      Border       border;           // border information of frame.
      BackGround   back;             // background information of frame.

      union
      {
          UNITTEXT txt;     // information of text unit.
          UNITIMAGEimg;     // information of image unit.
          UNITBAR           bar;     // information of barcode unit.
      };
} UNITINFO;
```

3.80 SmartComm_GetFieldLinkedUnitInfo

SmartComm_GetFieldLinkedUnitInfo is for getting the object's data(text, barcode, image) from linked fields.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetFieldLinkedUnitInfo(
      HSMART hHandle,
      WCHAR* szField,
      UNITINFO* pUnit,
      int dir
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

szField

[in] Field name.

pUnit

[out] A pointer of UNITINFO structure to get object's data.

dir

[in] Sets the direction to get data. If it is UNITINFO_FIRST, it searches drawing an object from the first part of list linked to the field. If it is UNITINFO_NEXT, it searches drawing an object from the next part where the object was brought the last time.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.81 SmartComm_SetUnitInfo

SmartComm_SetUnitInfo is for updating object's data(text, barcode, image).

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_SetUnitInfo(
      HSMART hHandle,
      UNITINFO* pUnit
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pUnit

[in] Information of drawing object.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.82 SmartComm_SetICG

SmartComm_SetICG command defines inter character gap/space of text.

It effects to SmartComm_DrawText, SmartComm_DrawText2 and CSD file.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_SetICG(
        HSMART hHandle,
        int page,
        int nICG
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Page

[in] page number which sets inter character gap/space.

nICG

[in] Inter character gap/space. The default value is “0”. When the value is negative, the gap will be narrower. When the value is positive, the gap will be wider.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.83 SmartComm_GetICG

SmartComm_GetICG command is for bringing the inter character gap/space value in the current page.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_GetICG(
        HSMART hHandle,
        int page,
        int* pnICG
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

Page

[in] Page number which sets the inter character gap/space.

pnICG

[out] Get the inter character gap/space value.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.84 SmartComm_IsBackEnable

SmartComm_IsBackEnable command will check that the back side is activated or not.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_IsBackEnable(
        HSMART hHandle,
```

```
        BOOL* pbEnabled
        );
```

Parameters

hHandle

[in] handle of the smart printer which is opened.

pbEnabled.

[out] Checks the back side whether it is activated or not.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.85 SmartCommEx_SerialCmdSend

SmartCommEx_SerialCmdSend is a function for sending commands through printer serial port.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_IsBackEnable(
        HSMART hHandle,
        Byte* pbtSend
        int nSendLen
        );
```

Parameters

hHandle

[in] handle of the smart printer which is opened.

pbtSend

[in] the pointer of sending packets.

nSendLen

[in] the length of sending packets.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

This function is only available when the serial port is connected inside of the printer.

After calling SmartCommEx_SerialCmdSend and then call SmartCommEx_SerialCmdRecv.

3.86 SmartCommEx_SerialCmdRecv

SmartCommEx_SerialCmdRecv is a function for receiving commands through printer serial port.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_IsBackEnable(
      HSMART hHandle,
      Byte* pbtRecv
      int nRecvLen
      );
```

Parameters

hHandle

[in] handle of the smart printer which is opened.

pbtRecv

[in] the pointer of receiving buffer.

nSendLen

[in] the size of receiving buffer.

[out] the length of data is in receiving buffer.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

This function is only available when the serial port is connected inside of the printer.

3.87 SmartComm_RFPowerOn

SmartComm_RFPowerOn is to operate functions for the contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_RFPowerOn(
        HSMART hHandle,
        int nDev,
        int* pnCardType,
        DWORD* pdwOutLen,
        BYTE* pOutBuf
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish the device (1: Internal, 2: External)

pnCardType

[in] not use.

pdwOutLen

[out] Length of contactless smart card's ATR value.

pOutBuf

[out] The length of contactless smart card's ATR value.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.88 SmartComm_RFPowerOff

SmartComm_RFPowerOff is for stopping the use of the contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_RFPowerOff(
        HSMART hHandle,
        int nDev
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish the device (1: Internal, 2: External)

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.89 SmartComm_RFInput

SmartComm_RFInput is PC/SC and is for sending APDU command to the contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_RFInput(
        HSMART hHandle,
        int nDev,
```

```
    DWORD dwTXSize,
    BYTE* lpbTxData,
    DWORD* pdwRXSize,
    BYTE* plbRxData
    );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish the device (1: Internal, 2: External)

dwInLen

[in] Size of APDU input buffer

pInBuf

[in] Input APDU buffer

pdwOutLen

[in/out] Size of output data.

pOutBuf

[out] Output buffer returned.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_RFInput function will not be supported in the future so please use SmartComm_RFTransmit function instead of SmartComm_RFInput.

3.90 SmartComm_RFOutput

SmartComm_RFOutput is PC/SC and is for sending APDU command to the contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_RFOutput(
        HSMART hHandle,
        int nDev,
        DWORD dwInLen,
        BYTE* pInBuf,
        DWORD* pdwOutLen,
        BYTE* pOutBuf
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish the device (1: Internal, 2: External)

dwInLen

[in] Input APDU buffer' size

pInBuf

[in] Input APDU buffer

pdwOutLen

[in/out] Size of output buffer

pOutBuf

[out] Output buffer which is returned

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_RFOutput function will not be supported in the future so please use SmartComm_RFTransmit function instead of SmartComm_RFOutput.

3.91 SmartComm_RFTransmit

SmartComm_RFTransmit is PC/SC and is sending APDU command to Contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_RFTransmit(
        HSMART hHandle,
        int nDev,
        DWORD dwTXSize,
        BYTE* lpbTXData,
        DWORD* pdwRXSize,
        BYTE* plbRXData
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish the device (1: Internal, 2: External)

dwTXSize

[in] APDU buffer' size

lpbTXData

[in] APDU buffer

pdwRXSize

[in/out] Size of the output buffer

plbTXData

[out] Output buffer which is returned

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_RFInput and SmartComm_RFOutput function will not be supported in the future so

please use SmartComm_RFTransmit function.

3.92 SmartCommEx_RFPCSC_GetReaderName.

SmartCommEx_RFPCSC_GetReaderName command is to know the name of RF reader PC/SC which is connected to the printer.

Operation Mode: USB, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_RFPCSC_GetReaderName(
        HSMART hHandle,
        int nWhich,
        WCHAR* szName
        );
```

Parameters

hHandle

[in] handle of the smart printer which is opened.

nWhich

[in] Defines the device which is used (0: internal RF reader, 1: extender RF reader)

szName

[out] Buffer for bringing RF reader's name.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.93 SmartComm_ICPowerOn

SmartComm_ICPowerOn is for operating the contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_ICPowerOn(
        HSMART hHandle,
        int nDev,
        DWORD* pdwOutLen,
        BYTE* pOutBuf
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish devices (1: Internal, 2: External)

pdwOutLen

[out] The length of contactless smart card's ATR.

pOutBuf

[out] Returns the contactless smart card's ATR.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

In case of using the internal module, the card should be moved to IC position and proceed the contacting between the card and reader by SmartComm_ICHContact and execute SmartComm_ICPowerOn.

3.94 SmartComm_ICPowerOff

SmartComm_ICPowerOff is for stopping the use of the contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_ICPowerOff(
        HSMART hHandle,
        int nDev
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] Number to distinguish the device (1: Internal, 2: External)

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

If it uses the internal module, the card and reader should be separated using SmartComm_ICHDiscontact, after using of SmartComm_ICPowerOff.

3.95 SmartComm_ICInput

SmartComm_ICInput is for sending APDU command set to the contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_ICInput(
        HSMART hHandle,
        int nDev,
        DWORD dwInLen,
        BYTE* pInBuf,
        DWORD* pdwOutLen,
        BYTE* pOutBuf
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish devices (1: Internal, 2: External)

dwInLen

[in] Size of input buffer

pInBuf

[in] Input buffer

pdwOutLen

[in/out] Size of output buffer

pOutBuf

[out] Output buffer of data read.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_ICInput function will not be supported in the future so please use SmartComm_ICTransmit function.

3.96 SmartComm_ICOutput

SmartComm_ICOutput is for sending APDU command set to the contactless smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_ICOutput(
        HSMART hHandle,
        int nDev,
        DWORD dwInLen,
```

```
    BYTE* pInBuf,
    DWORD* pdwOutLen,
    BYTE* pOutBuf
    );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish the device (1: Internal, 2: External)

dwInLen

[in] Input buffer' size

pInBuf

[in] Input buffer

pdwOutLen

[in/out] Size of output buffer

pOutBuf

[out] Output buffer

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_ICOutput function will not be supported in the future so please use SmartComm_ICTransmit function.

3.97 SmartComm_ICTransmit

SmartComm_ICTransmit is for sending APDU commands to Contact smart card.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartComm_ICTransmit(
        HSMART hHandle,
        int nDev,
        DWORD dwInLen,
        BYTE* pInBuf,
        DWORD* pdwOutLen,
        BYTE* pOutBuf
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

nDev

[in] The number to distinguish the device (1: Internal, 2: External)

dwInLen

[in] Input buffer' size

pInBuf

[in] Input buffer

pdwOutLen

[in/out] Size of output buffer

pOutBuf

[out] Output buffer

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

SmartComm_ICInput and SmartComm_ICOutput function will not be supported in the future so please use SmartComm_ICTransmit function.

3.98 SmartCommEx_IC_GetICReaderInfo

SmartCommEx_IC_GetICReaderInfo command is to know the name of PC/SC of IC reader.

Operation Mode: USB, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_IC_GetICReaderInfo(
        HSMART hHandle,
        int* pnIdx,
        WCHAR* szName
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pnIdx

[in] Index of SMART Printer in DEVICEINFO.

szName

[out] Buffer for IC reader's name.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.99 SmartCommEx_IC_ GetSIMReaderInfo

SmartCommEx_IC_GetSIMReaderInfo command is to know the name of SIM reader PC/SC.

Operation Mode: USB, Driver, DCL

Operation Device: SMART-50, 30

Definition

```
int SmartCommEx_IC_GetSIMReaderInfo(
        HSMART hHandle,
        int* pnIdx,
```

```
        WCHAR* szName
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pnIdx.

[in] Index of SMART Printer in DEVICEINFO.

szName

[out] Buffer for SIM reader's name

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

3.100 SmartCommEx_SetPassword

SmartCommEx_SetPassword command is to set the password of the administrator in SMART Printer.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_SetPassword(
        HSMART hHandle,
        WCHAR* szOldPW,
        WCHAR* szNewPW
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

szOldPW

[in] Old password. New printer does not have the password. The password can be set as

maximum 8 digits.

szNewPW

[in] New password. The maximum is 8 digits.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

The return value of this function is to know whether the command is sent successfully. To know whether the password is changed or not, check the printer status by SmartComm_GetStatus commands. If the password is incorrect or generate Errors, SMSC_F_INCORRECT_PW code will be turned “On” when SMART-50 or SMART-30 is connected, S51PS_F_INVALIDPASSWORD code will be turned “On” when SMART-51 or SMART-31 is connected.

In the CardPrinterConfig, when Root Auth. is activated in Security and the printer is turned on, the printer will be locked. To use the printer, you have to do certification process using SmarCommEx_ UnLockPrinter2.

3.101 SmartCommEx_SetUserPassword

SmartCommEx_SetUserPassword command is to set the user password.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_SetUserPassword(
      HSMART hHandle,
      WCHAR* szRootPW,
      WCHAR* szUserPW
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

szRootPW

[in] Administrator’s password which is set at the moment. New printer does not have the

password. You can input the password with maximum 8 digits.

szUserPW

[in] User's password which will be set in the printer. You can input the password with maximum 8 digits.

Return Value

SM_SUCCESS is returned if the function is operated successfully.

Remarks

The return value of this function is to know whether the command is sent successfully. To know whether the password is changed or not, check the printer status by SmartComm_GetStatus commands. If the password is incorrect or generate Errors, SMSC_F_INCORRECT_PW code will be turned "On" when SMART-50 or SMART-30 is connected, S51PS_F_INVALIDPASSWORD code will be turned "On" when SMART-51 or SMART-31 is connected.

In the CardPrinterConfig, when Root Witness is activated in PC Authentification and the printer is turned on, the printer will be locked. To use the printer, you have to do certification process using SmarCommEx_UnLockPrinter2.

3.102SmartCommEx_UnlockPrinter2

SmartCommEx_UnlockPrinter2 command is to unlock a printer when the printer is locked by Root Witness (Root Auth.) or User Witness (User Auth.).

Operation Mode: USB, Network, Driver, DCL
Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_UnlockPrinter2(
      HSMART hHandle,
      WCHAR* szPW
      );
```

Parameters

hHandle

[in] A handle of SMART printer which is opened.

szPW

[in] Password of administrator or user which will be unlocked.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

The return value of this function is to know whether the command is sent successfully. To know whether the password is changed or not, check the printer status by SmartComm_GetStatus commands. If the password is incorrect or generate Errors, SMSC_F_INCORRECT_PW code will be turned “On ” when SMART-50 or SMART-30 is connected, S51PS_F_INVALIDPASSWORD code will be turned “On” when SMART-51 or SMART-31 is connected.

.

3.103SmartCommEx_LockPrinter2

SmartCommEx_LockPrinter2 command is to lock a printer from unlocking status.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartCommEx_LockPrinter2(
        HSMART hHandle,
        WCHAR* szPW
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

szPW

[in] Password of administrator or user.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remark

The return value of this function is to know whether the command is sent successfully. To know

whether the password is changed or not, check the printer status by SmartComm_GetStatus commands. If the password is incorrect or generate Errors, SMSC_F_INCORRECT_PW code will be turned “On” when SMART-50 or SMART-30 is connected, S51PS_F_INVALIDPASSWORD code will be turned “On” when SMART-51 or SMART-31 is connected.

3.104SmartLami_GetStatus

SmartLami_GetStatus command is for getting laminator’s status.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51, 31

Definition

```
int SmartLami_GetStatus(
        HSMART hHandle,
        BYTE* pstatus
        int* pnlen
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pstatus

[in] the pointer of buffer for getting laminator status

SMART-50 Laminator Status Structure

offset	size	type	description
0	8	__int64	Laminator status information Please refer to below Remark section.
8	1	BYTE	Film type.
9	2	char[2]	reserved
11	1	BYTE	A maximum number of laminating film. Multiply this value by 10 for the actual number.
12	2	USHORT	Remained number of films.
14	2	char[2]	reserved

16	2	char[2]	reserved
18	1	BYTE	Laminator local code.
19	1	BYTE	Laminator vendor code.
20	1	BYTE	Film local code.
21	1	BYTE	Film vendor code.

SMART-51 Laminator Status Structure

offset	size	type	description
0	8	__int64	Laminator status information Please refer to below Remark section.
8	1	BYTE	Film type.
9	1	BYTE	Film maker.
10	2	USHORT	A maximum number of laminating film.
12	2	USHORT	Remained number of films.
14	2	char[2]	reserved
16	2	char[2]	reserved
18	1	BYTE	Laminator local code.
19	1	BYTE	Laminator vendor code.
20	1	BYTE	Film local code.
21	1	BYTE	Film vendor code.

pnlen

[in] the size of pstatus buffer. Must give the size of pstatus buffer, unless it will give an error.

[out] the length of data written in pstatus buffer.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remark

This function is only available when laminator is equipped in case of the SMART-50 printer and printer firmware is more than 1.00.60. This function is only available when laminator is equipped in case of the SMART- 51 printer.

Below is the simple code for calling this function.

```
__int64 flipStatus = 0;
int len = sizeof( flipStatus );
SmartLami_GetStatus( hsmart, (BYTE*)&flipStatus, &len );
```

SMART-50 Printer Laminator's statuses are defined in SmartComm2.dll.h.

```
// MOTION-PART
```

```
#define    LMSC_M_HEATHDRLIFTUP     0x0000000000000001     // Heat-header is being lifted up
#define    LMSC_M_HEATHDRLIFTDOWN   0x0000000000000004     // Heat-header is being lifted down
#define    LMSC_M_CARDIN            0x0000000000000010     // Card In
#define    LMSC_M_MOVE_LAMINATE     0x0000000000000040     // Moving to laminate position
#define    LMSC_M_FRONTCARDOUT      0x0000000000000100     // Card is being moved out to front.
                                                           (Printer side)
#define    LMSC_M_REARCARDOUT       0x0000000000000200     // Card is being moved out to rear.
                                                           (stacker side)
#define    LMSC_M_ROTATE            0x0000000000000400     // Rotating
#define    LMSC_S_WAIT              0x0000000000000800     // Waiting
#define    LMSC_S_CMDRUN            0x0000000000002000     // Command is being executed.
#define    LMSC_M_HEATING           0x0000000000004000     // Heating Heat-header
#define    LMSC_S_CASEOPEN          0x0000000000008000     // Top cover is opened
#define    LMSC_M_LAMINATING        0x0000000000010000     // Laminating
#define    LMSC_S_CARDINSENSOR      0x0000000020000000     // Caught by card in sensor
#define    LMSC_S_CARDOUTSENSOR     0x0000000040000000     // Caught by card out sensor
#define    LMSC_S_OUTDOORSENSOR     0x0000000080000000     // Caught by stacker open sensor
// ERROR-PART
#define    LMSC_E_HEATHDRLIFTUP     0x0000000100000000     // Error while lifting up Heat-header
#define    LMSC_E_HEATHDRLIFTDOWN   0x0000000400000000     // Error while lifting down Heat-header
#define    LMSC_E_CARDIN            0x0000001000000000     // Error while moving card in
#define    LMSC_E_MOVE_LAMINATE     0x0000004000000000     // Error while moving to Laminate position.
#define    LMSC_E_FRONTCARDOUT      0x0000010000000000     // Error while front card out (printer side)
#define    LMSC_E_REARCARDOUT       0x0000020000000000     // Error while rear card out (stacker side)
#define    LMSC_E_ROTATE            0x0000040000000000     // Error while rotating
#define    LMSC_E_INIT              0x0000800000000000     // Error while initializing laminator
#define    LMSC_E_EMPTYFILM         0x0001000000000000     // Film remain count is 0.
#define    LMSC_E_NOFILM            0x8000000000000000     // No film is installed, or cannot recognize
```

SMART-51 Printer Laminator's statuses are defined in SmartComm2.dll.h.

```
// MOTION-PART
#define    S51LS_S_READY              0x0000000000000001  // Ready
#define    S51LS_S_BUSY               0x0000000000000002  // Command is being executed
#define    S51LS_M_CARDMOVE           0x0000000000000004  // Card is being moved
#define    S51LS_M_CARDIN             0x0000000000000008  // Card In
#define    S51LS_M_CARDEJECT          0x0000000000000010  // Card Out
#define    S51LS_M_THEADLIFT          0x0000000000000020  // Laminator head is being lifted up/down
#define    S51LS_M_LAMINATE           0x0000000000000040  // Laminating
#define    S51LS_M_FLIPTRAYMOVE       0x0000000000000080  // Card is being moved to flipper
#define    S51LS_S_FLIPTRAYTOPSIDED   0x0000000000000100  // Flipper is up state
#define    S51LS_M_HEATHEADHEAT       0x0000000000010000  // Heating laminator head
#define    S51LS_M_FILMMOTOR          0x0000000000020000  // Film Motor is operating
#define    S51LS_M_HEADMOTOR          0x0000000000040000  // Head Motor is operating
#define    S51LS_M_CARDMOVESTEPMOTOR  0x0000000000080000  // Step Motor is operating
#define    S51LS_M_FLIPSTEPMOTOR      0x0000000000100000  // Flip Motor is operating
#define    S51LS_S_DETECTENCODERCNTINC 0x0000000000200000 // Sensing the encoder counter increased
#define    S51LS_S_DETECTENCODERCNTDEC 0x0000000000400000 // Sensing the encoder counter decreased
#define    S51LS_S_COVEROPENED        0x0000000000800000  // Case Open
```

```
#define    S51LS_S_LOCKSENSOR          0x0000000001000000  // Locked
#define    S51LS_S_CARDCENTERSENSOR    0x0000000002000000  // Caught by Card Center sensor
#define    S51LS_S_CARDOUTSENSOR       0x0000000004000000  // Caught by Card Out sensor
#define    S51LS_S_FLIPPERCENTERSENSOR 0x0000000008000000  // Caught by Flipper Center sensor
#define    S51LS_S_FLIPPERFLIPSENSOR   0x0000000010000000  // Caught by Flipper flip sensor
#define    S51LS_S_HEADSENSOR          0x0000000020000000  // Caught by Head up/down sensor
#define    S51LS_S_FILMMARKMAINSENSOR  0x0000000040000000  // Caught by Flim mark main sensor
#define    S51LS_S_FILMMARKSUBSENSOR   0x0000000080000000  // Caught by Flim mark sub sensor

// ERROR-PART
#define    S51LS_F_CARDIN              0x0000000100000000  // Error while moving card in
#define    S51LS_F_CARDMOVE            0x0000000200000000  // Error while moving card
#define    S51LS_F_CARDEJECT           0x0000000400000000  // Error while ejecting card
#define    S51LS_F_HEADLIFT            0x0000000800000000  // Error while head up/down
#define    S51LS_F_LAMINATE            0x0000001000000000  // Error while laminating
#define    S51LS_F_COMMAND             0x0000002000000000  // Error while executing command
#define    S51LS_F_FLIPTRAYMOVE        0x0000004000000000  // Error while moving card in the flip tray
#define    S51LS_E_INIT                0x0001000000000000  // Error while initialzing laminator
#define    S51LS_E_FILMSEARCH          0x0002000000000000  // Error while searching a film position
#define    S51LS_E_FILM0               0x0004000000000000  // Film remain count is 0
#define    S51LS_E_NOFILM              0x0008000000000000  // No film is installed or can not recognize
#define    S51LS_E_HEADOVERHEAT        0x0010000000000000  // Head is overheated or fault
```

3.105 SmartFlip_GetStatus

SmartFlip_GetStatus command is for getting flipper's status.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-51, 31

Definition

```
int SmartFlip_GetStatus(
      HSMART hHandle,
      BYTE* pstatus
      int* pnlen
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

pstatus

[in] the pointer of buffer for getting flipper status

SMART-51, 31 Flipper Status Structure

offset	size	type	description
0	8	__int64	Flipper status information Please refer to below Remark section.

pnlen

[in] the size of pstatus buffer. Must give the size of pstatus buffer, unless it will give an error.

[out] the length of data written in pstatus buffer.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remark

This function is only available when the flipper is equipped in case of the SMART- 51 and SMART-51 printer. The installation of flipper can be checked by the return value of calling SmartComm_GetStatus.

The SM_F_NOTSUPPORTYET is returned if the printer is SMART-50 and SMART-30.

Below is the simple code for calling this function.

```
__int64 flipStatus = 0;
int len = sizeof( flipStatus );
SmartFlip_GetStatus( hsmart, (BYTE*)&flipStatus, &len );
```

SMART-51,31 Printer flipper's statuses are defined in SmartComm2.dll.h.

```
// MOTION-PART
#define   S51FS_S_READY              0x0000000000000001  // ready
#define   S51FS_S_BUSY               0x0000000000000002  // busy (doing something)
#define   S51FS_M_CARDMOVE           0x0000000000000004  // moving card
#define   S51FS_M_CARDIN             0x0000000000000008  // inserting card
#define   S51FS_M_CARDEJECT          0x0000000000000010  // ejecting card
#define   S51FS_M_FLIP               0x0000000000000020  // flipping
#define   S51FS_S_FLIPTRAYTOPSIDED   0x0000000000000100  // flip tray top sided
#define   S51FS_S_ACTIVATEDREARSENSOR 0x0000000002000000  // activated rear sensor
#define   S51FS_M_CARDMOVESTEPMOTOR  0x0000000004000000  // working card move step motor
#define   S51FS_M_FLIPSTEPMOTOR      0x0000000008000000  // working flip step motor
#define   S51FS_S_COVERCLOSED        0x0000000010000000  // cover is opened
#define   S51FS_S_CENTERSENSOR       0x0000000020000000  // center sensor
#define   S51FS_S_REARSENSOR         0x0000000040000000  // rear sensor
#define   S51FS_S_FLIPSENSOR         0x0000000080000000  // flip sensor

// ERROR-PART
```

```
#define    S51FS_F_CARDIN            0x0000000100000000  // failed to card insert
#define    S51FS_F_CARDMOVE          0x0000000200000000  // failed to card move
#define    S51FS_F_CARDEJECT         0x0000000400000000  // failed to card eject
#define    S51FS_F_MOVEFLIPTRAY      0x0000000800000000  // failed to move flip tray
#define    S51FS_F_COMMAND           0x0000001000000000  // failed while process command
#define    S51FS_E_INIT              0x0001000000000000  // error from initializing
```

3.106 SmartLami_GetVersion

SmartLami_GetVersion command is for getting laminator's version.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 51

Definition

```
int SmartLami_GetVersion(
      HSMART hHandle,
      WCHAR* szVer
      int* pcblen
      );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

szVer

[Out] A pointer of the Unicode buffer of Laminator version

pcblen

[in] the size of szVer.

[out] the length of character set in szVer buffer.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remark

This function is only available when laminator is equipped in case of the SMART-50 printer and printer firmware is more than 1.00.60. This function is only available when laminator is equipped in case of the SMART- 51 printer. To check laminator is installed, call SmartComm_GetStatus and check

that SMSC_S_EQUIPLAMINATOR is set.

3.107 SmartCommEx_GetConfig

SmartCommEx_GetConfig command is for getting the configuration of Smart SDK.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51,31

Definition

```
int SmartCommEx_GetConfig(
        HSMART hHandle,
        int cfgId,
        cfgval* pvalue
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

cfgId

[in] Configuration ID of SDK is set

```
#define  SMEX_CFG_STATUS51TO50     0x0E    // flag of Converted SMART-51 status code to SMART-50's
#define  SMEX_CFG_RESOLUTION       0x10    // Resolution of printing
```

SMEX_CFG_STATUS51TO50 uses SmartComm_GetStatus, SmartLami_GetStatus. You can check whether the status code of SMART-51 printer and laminator is converted to SMART-50's.

SMEX_CFG_RESOLUTION is the resolution value of printing.

pvalue

[out] the current configuration value of *cfgId*

If *cfgid* is SMEX_CFG_STATUS51TO50, the returned *pvalue* is as below.

```
#define  SMEX_CFG_USE              0       // use the configuration ID function
#define  SMEX_CFG_NOTUSE           1       // not use the configuration ID function
```

If *pvalue* is SMEX_CFG_USE, the status code for SMART-51 is converted to SMART-50's.

If *cfgid* is SMEX_CFG_RESOLUTION, the returned *pvalue* is as below.

```
#define  SMEX_CFG_RES300X300       0       // 300x300 dpi
#define  SMEX_CFG_RES300X600       1       // 300x600 dpi
#define  SMEX_CFG_RES300X1200      2       // 300x1200 dpi
```

Default value is SMEX_CFG_RES300X300.

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

Even though SMEX_CFG_RES300X600 or SMEX_CFG_RES300X1200 is set, the resolution range for X, Y position of objects is same. (1012x636 or 636x1012)

3.108 SmartCommEx_SetConfig

SmartCommEx_GetConfig command is for setting configuration of Smart SDK.

Operation Mode: USB, Network, Driver, DCL

Operation Device: SMART-50, 30, 51,31

Definition

```
int SmartCommEx_SetConfig(
        HSMART hHandle,
        int cfgId,
        cfgval value
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

cfgId

[in] Configuration ID of SDK is set

```
#define  SMEX_CFG_STATUS51TO50     0x0E    // flag of Converted SMART-51 status code to SMART-50's
#define  SMEX_CFG_RESOLUTION       0x10    // Resolution of printing
```

SMEX_CFG_STATUS51TO50 uses SmartComm_GetStatus, SmartLami_GetStatus. You can check whether the status code of SMART-51 printer and laminator is converted to SMART-50's.

SMEX_CFG_RESOLUTION is the resolution value of printing.

value

[in] the configuration value to set of *cfgId*

If *cfgid* is SMEX_CFG_STATUS51TO50, the returned *pvalue* is as below.

```
#define  SMEX_CFG_USE              0       // use the configuration ID function
#define  SMEX_CFG_NOTUSE           1       // not use the configuration ID function
```

If the *value* is SMEX_CFG_USE, the status code of SMART-51 and 31 is converted to SMART-50's.

If the *value* is SMEX_CFG_NOTUSE, the returned status code is not coverted. It means that the status code of SMART-50 is returned when the SMART-50 and 30 are connected, and the status code of SMART-51 is returned when the SMART-51 and 31 is connected. So, the program should check the connected printer before checking a status code when SMEX_CFG_NOTUSE is set.

If *cfgid* is SMEX_CFG_RESOLUTION, the returned *value* is as below.

```
#define  SMEX_CFG_RES300X300       0       // 300x300 dpi
#define  SMEX_CFG_RES300X600       1       // 300x600 dpi
#define  SMEX_CFG_RES300X1200      2       // 300x1200 dpi
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remarks

Even though SMEX_CFG_RES300X600 or SMEX_CFG_RES300X1200 is set, the resolution range for X, Y position of objects is same. (1012x636 or 636x1012)

3.109 SmartKiosk_Hopper

SmartKiosk_Hopper command is for retrieving the information of Multi hopper.

Operation Mode: USB, Driver, DCL

Operation Device: SMART-51, 31

Definition

```
int SmartKiosk_Hopper(
        HSMART hHandle,
        int hopper,
        int opt,
        int* pvalue
```

```
);
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

hopper

[in] Hopper number to get information

```
#define    KIOSK_HOPPER_AUTO      0     // Get a card automatically among hoppers
#define    KIOSK_HOPPER_1         0x01  // No.1 Hopper
#define    KIOSK_HOPPER_2         0x02  // No.2 Hopper
#define    KIOSK_HOPPER_3         0x04  // No.3 Hopper
#define    KIOSK_HOPPER_4         0x08  // No.4 Hopper
#define    KIOSK_HOPPER_5         0x10  // No.5 Hopper
#define    KIOSK_HOPPER_6         0x20  // No.6 Hopper
```

opt

[in] information type to get.

```
#define    KIOSK_HOPPER_SCAN      0
#define    KIOSK_HOPPER_STATUS    1
```

Scans the hoppers if *opt* is KIOSK_HOPPER_SCAN. Scanned hopper information is returned to *pvalue* by the bit-or operation. In this case, the value of hopper is ignored

Gets the information of hopper value if *opt* is KIOSK_HOPPER_STATUS. The status of the hopper is as below.

```
#define    KIOSK_HOPPER_READY          0              // Hopper is ready
#define    KIOSK_HOPPER_ERROR          0x8000000F     // Error in the hopper
#define    KIOSK_HOPPER_EMPTY          0x8000000B     // No card in the hopper
#define    KIOSK_HOPPER_NOTEXIST       0x80000003     // That hopper doesn't exist
```

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remark

This function is only available when multi-hopper is equipped in the SMART-51, 31 printers.

3.110 SmartKiosk_CardIn2

SmartKiosk_CardIn2 is for inserting a card to inside of printer from multi-hopper.

Operation Mode: USB, Driver, DCL

Operation Device: SMART-51, 31

Definition

```
int SmartKiosk_CardIn2(
        HSMART hHandle,
        int hopper,
        int opt
        );
```

Parameters

hHandle

[in] A handle of SMART Printer opened.

hopper

[in] Hopper number to insert a card

```
#define     KIOSK_HOPPER_AUTO     0                  // Get a card automatically among hoppers
#define     KIOSK_HOPPER_1        0x01               // No.1 Hopper
#define     KIOSK_HOPPER_2        0x02               // No.2 Hopper
#define     KIOSK_HOPPER_3        0x04               // No.3 Hopper
#define     KIOSK_HOPPER_4        0x08               // No.4 Hopper
#define     KIOSK_HOPPER_5        0x10               // No.5 Hopper
#define     KIOSK_HOPPER_6        0x20               // No.6 Hopper
```

opt

[in] Not use

Return Values

SM_SUCCESS is returned if the function is operated successfully.

Remark

This function is only available when multi-hopper is equipped in the SMART-51, 31 printers. To check multi-hopper is installed, call SmartKiosk_Hopper and check the status of the multi-hopper before using this function

4 Migration from SDK 1 to SDK 2

SMART SDK2 is compatible with previous version SDK1 so that programmers can use the program which is set to SDK1 without any modifications. However, it is recommended to change the program according to SDK2 when a network printer is used or new functions are used.

Please refer the followings to set your program according to SDK2.

4.1 Integration of DLL files

In SDK 1, two DLL files (SmartCommon.dll and SmartCommCl.dll) are used for different connections, local and network. However, these 2 DLL files have been integrated as SmartComm2.dll in SDK2, so both local printer and network printer can be used through the DLL and it makes your work easier and convenient.

4.2 Calling printer list

The structures for calling printer list are updated in SDK2.

The names of functions are still the same but the names of structures and contents inside are modified.

SDK 1:

```
typedef struct {
        int nID;
        WCHAR ID[32][16];
} DEVICEINFO;

DEVICEINFO list = {0, };
int nres = SmartComm_GetDeviceList( &list );
```

SDK 2:

```
typedef struct {
        WCHAR   name[128];      // printer name
        WCHAR   id[64];         // printer ID
        WCHAR   dev[64];        // device connection
        WCHAR   desc[256];      // description
        int     pid;            // USB product ID
} SMART_PRINTER_ITEM;

typedef struct {
        int     n;
        SMART_PRINTER_ITEM      item[32];
} SMART_PRINTER_LIST;

SMART_PRINTER_LIST      list;
int nres = SmartComm_GetDeviceList2( &list );
```

4.3 To distinguish between local and network

There are 2 methods to distinguish printer connections as follows

SDK 2: **// By SmartComm_GetDeviceList2 function.**

```
SMART_PRINTER_LIST      list;
int nres = SmartComm_GetDeviceList2( &list );
if( wcsncmp(list.item[0].dev, L"USB", 3) == 0 ) {
        // Local USB Printer
} else {
        // Network Printer
}
```

SDK 2: **// By SmartComm_GetDeviceInfo2 function.**

```
typedef struct {
        WCHAR   port[64];               // usb port
        WCHAR   link[256];              // symbolic link of usb port
        int     is_bridge;              // Network module bridge
}       SMART_PRINTER_PORT_USB;

typedef struct {
        WCHAR   ver[64];                // version of network protocol
        WCHAR   ip[64];                 // ip address
        int     port;                   // tcp port
        int     is_ssl;                 // ssl protocol
}       SMART_PRINTER_PORT_NET;

typedef struct {
        WCHAR   name[128];              // printer name
        WCHAR   id[64];                 // printer ID
        WCHAR   dev[64];                // device connection
        int     dev_type;               // 1=USB, 2=NET
        int     pid;                    // USB product ID

        SMART_PRINTER_PORT_USB        usb;
        SMART_PRINTER_PORT_NET        net;
} SMART_PRINTER_STANDARD;

typedef struct {
        int     is_dual;                // dual printer
        WCHAR   ic1[64];                // internal contact encoder
        WCHAR   ic2[64];                // external contact SIM encoder
        WCHAR   rf1[64];                // internal contactless encoder
        WCHAR   rf2[64];                // external contactless encoder
} SMART_PRINTER_OPTIONS;

typedef struct {
        SMART_PRINTER_STANDARD        std;
        SMART_PRINTER_OPTIONS         opt;
} SMART_PRINTER_INFO;


SMART_PRINTER_LIST      list;
int nres = SmartComm_GetDeviceList2( &list );
```

```
SMART_PRINTER_INFO      info;
nres = SmartComm_GetDeviceInfo2( &info, list.item[0].desc,
                                    SMART_OPENDEVICE_BYDESC );
if( info.std.dev_type == 1 ) {
        // Local USB Printer
} else {
        // Network Printer
}
```

4.4 Connecting to the printer

In SDK1, only printer ID is used for connecting to the printer, however, both printer ID and Description can be used for connecting to the printer, in SDK2.

SDK 1:

```
DEVICEINFO list = {0, };
SmartComm_GetDeviceList( &list );df

HSMART          hsmart = NULL;
int             nres = SM_SUCCESS;
nres = SmartComm_OpenDevice( &hsmart, list.ID[0] );
// or
nres = SmartDCL_OpenDevice( &hsmart, list.ID[0], DMORIENT_LANDSCAPE );
```

SDK 2:

```
SMART_PRINTER_LIST      list;
SmartComm_GetDeviceList2( &list );

HSMART          hsmart = NULL;
int             nres = SM_SUCCESS;
nres = SmartComm_OpenDevice2( &hsmart, list.item[0].id, SMART_OPENDEVICE_BYID );
// or
nres = SmartComm_OpenDevice( &hsmart, list.item[0].desc,
                             SMART_OPENDEVICE_BYDESC );
// or
nres = SmartDCL_OpenDevice2( &hsmart, list.item[0].id,
                              SMART_OPENDEVICE_BYID, DMORIENT_LANDSCAPE );
// or
nres = SmartDCL_OpenDevice2( &hsmart, list.item[0].desc,
                              SMART_OPENDEVICE_BYDESC, DMORIENT_LANDSCAPE );
```

4.5 Sending APDU command to Contact/Contactless Smart Card

SmartComm_ICInput, SmartComm_ICOutput in SDK1, are the same function for sending APDU command to Contact Smart Card Encoder.

These 2 functions have been updated as one single SmartComm_ICTransmit function in SDK2.

SmartComm_ICInput, SmartComm_ICOutput will not be supported in the future.

SmartComm_RFInput, SmartComm_RFOutput in SDK1, are the same function for sending APDU command to Contactless Smart Card Encoder.

These 2 functions have been updated as one single SmartComm_RFTransmit function in SDK2. SmartComm_RFInput, SmartComm_RFOutput will not be supported in the future.

SDK 1:

```
BYTE btAPDU[] = { … };
int  nAPDULen = sizeof btAPDU;
BYTE btRecv[512];
int   nRecvLen = sizeof btRecv;
int   nres = SmartComm_ICInput( hsmart, INTERNALDEV,
                                nAPDULen, btAPDU, RecvLen, btRecv );
```

SDK 2:

```
BYTE btAPDU[] = { … };
int  nAPDULen = sizeof btAPDU;
BYTE btRecv[512];
int   nRecvLen = sizeof btRecv;
int   nres = SmartComm_ICTransmit( hsmart, INTERNALDEV,
                                   nAPDULen, btAPDU, RecvLen, btRecv );
```

4.6 The attentive point when use DCL mode.

When you use DCL mode and SmartComm_GetSurface please check the below.

There's a difference of SMART_SURFACE structure between SDK 1 and SDK 2.

SMART_SURFACE_PROPERTIES structure in SMART_SURFACE is

SDK 1:

```
typedef struct {
        DWORD   side;
        DWORD   orientation;
        DWORD   width;
        DWORD   height;
} SMART_SURFACE_PROPERTIES;
```

SDK 2:

```
typedef struct {
        DWORD   side;
        DWORD   orientation;
        DWORD   ribbon;
        DWORD   ribbon_type;
        DWORD   width;
        DWORD   height;
} SMART_SURFACE_PROPERTIES;
```

In SDK 2, `DWORD ribbon` and `DWORD ribbon_type` members are newly included in SMART_SURFACE_PROPERTIES.

Therefore, please change the structure of SMART_SURFACE_PROPERTIES first, before changing from SDK 1 to SDK 2.

5 Migration from SMART-50 to SMART-51

The next model of SMART-50 and 30 is SMART-51 and 31. They basically operate similarly to SMART-50 and 30 but some parts are changed. So, please refer to the followings to use the customized program by SDK for SMART-50 and 30 with the SMART-51 and 31.

5.1 Conversion of status codes for printer

The status code of SMART-50, 30 is different from SMART-51,31's. So, if the SMART-51, 31 is connected without modification, it doesn't work because the wrong status code is returned. To solve it, there are two ways to convert the status code for SMART-51,31 to SMART-50's.

5.1.1 Use the configuration file (SmartComm2.ini)

The SmartComm2.dll uses the configuration value in the SmartComm2.ini when it is loaded. Make a SmartComm2.ini and insert the below strings to convert a status code automatically.

```
[DLL]
TranslateStatus=50
```

* Smartcomm2.dll tries to search Smartcomm2.ini in the same directory.
If it can't, it searches in the 'C:\'.

5.1.2 Use the DLL function

Use the SmartCommEx_SetConfig to convert the status codes for SMART-51,31 to SMART-50,30's.

```
HSMART hsmart;


…
SmartCommEx_SetConfig( hsmart, SMEX_CFG_STATUS51TO50, SMEX_CFG_USE );
```

5.2 Conversion of configuration structure for a printer

The configuration for SMART-50, 30 is in the SMART_DEVMODE (SHASM_DEVMODE) structure. On the other hand, the configuration for SMART-51, 31 is in the SMART-51_DEVMODE structure and two structures are different. But, even if the SMART_DEVMODE is used with SMART-51,31, it can work without modification.

But for setting a parameter in SMART51_DEVMODE, please modify to use SMART51_DEVMODE

when the SMART-51, 31 is connected. Please use SmartComm_GetPrinterSettings2, SmartComm_SetPrinterSettings2 for using SMART51_DEVMODE.

5.3 Modification of SURFACE structure

For drawing directly on the DC for the panel on DCL mode, please get and use the pointer of the SMART_SURFACE structure with SMART-50,30 or SMART51_SURFACE structure with SMART-51, 31. Please use SmartDCL_GetSurface2 for using both structures.

```
SMART_SURFACE * pSurfFront = NULL;
SMART_SURFACE * pSurfBack = NULL;
int surflen = sizeof( SMART_SURFACE );

SmartDCL_GetSurface2( hsmart, PAGE_FRONT, (void**) &pSurfFront, &surflen );
SmartDCL_GetSurface2( hsmart, PAGE_BACK, (void**) &pSurfBack, &surflen );

if( pSurfFront )
    FillRect( pSurfFront->bmp.hdcColor, … );
```

```
SMART51_SURFACE * pSurf51Front = NULL;
SMART51_SURFACE * pSurf51Back = NULL;
int surf51len = sizeof( SMART51_SURFACE );

SmartDCL_GetSurface2( hsmart, PAGE_FRONT, (void**) &pSur51fFront, &surflen );
SmartDCL_GetSurface2( hsmart, PAGE_BACK, (void**) &pSurf51Back, &surflen );

if( pSurf51Front )
    FillRect( pSurf51Front->bmp.hdcColor, … );
```

6 Sample Code

In “SMART Printer SDK”, sample program and source codes are included for developers to use SDK. The sample program is composed of simple functions so you can follow to use each function and program easily.

6.1 VC++ Source: SmartCommonTest

SmartCommonTest in “SMART Printer SDK” is an example of SMART Printer using SmartComm2.dll. SmartCommonTest is developed in VC of Microsoft Visual Studio 2005.

If executes SmartCommonTest, the window such as figure 5 will be displayed. Printer descriptions are displayed on “Printer” list and you can choose a printer which will be used. After choosing a printer, you can test SMART Printer with this program.

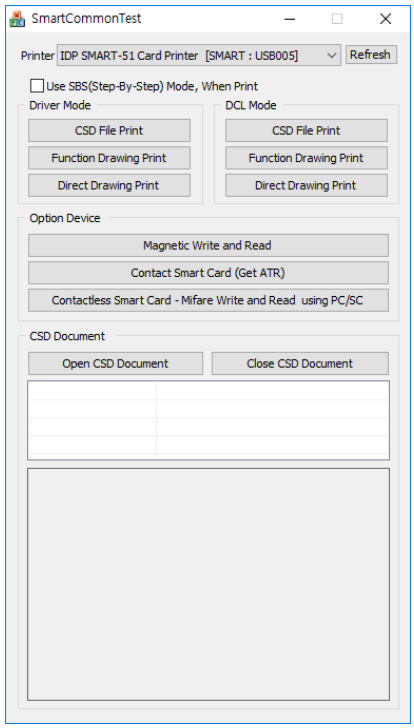

Figure 5 SmartCommonTest

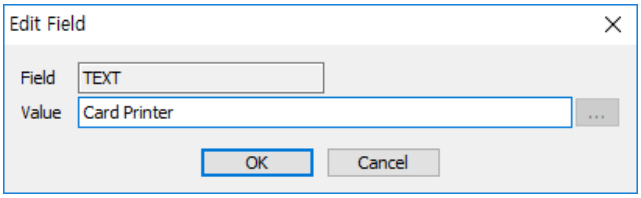

Figure 6 Edit Field Value

“Use SBS(Step-By-Step) Mode, When Print” checkbox decides whether you will use SBS mode or not. If you check the box, SBS mode will be set and it is applied during the printing process.

There are three buttons for each operating mode (Driver, DCL).

There are 3 ways to print in Driver mode.

“CSD File Print” is for printing CSD file which is made by SmartID. It is executed on OnBnClickedButtonDriverCsd () of SmartCommonTestDlg.cpp file.

“Function Drawing Print” is for printing the image and text using SmartComm_DrawXXX functions. It is executed on OnBnClickedButtonDriverFunction of SmartCommonTestDlg.cpp.

“Direct Drawing Print” is for printing image and text by drawing directly. It is executed on OnBnClickedButtonDriverDirect() of SmartCommonTestDlg.cpp.

There are 3 ways to print in DCL mode.

“CSD File Print” is for printing CSD file which is made by SmartID. It is executed on OnBnClickedButtonDclCsd () of SmartCommonTestDlg.cpp file.

“Function Drawing Print” is for printing the image and text using SmartComm_DrawXXX functions. It is executed on OnBnClickedButtonDclFunction of SmartCommonTestDlg.cpp.

“Direct Drawing Print” is for printing image and text by drawing directly. It is executed on OnBnClickedButtonDclDirect() of SmartCommonTestDlg.cpp.

There are buttons to test option devices at the bottom.

“Magnetic Write and Read” is for testing the read/write of the magnetic stripe. This is executed on OnBnClickedButtonMagnetic () of SmartCommonTestDlg.cpp.

“Contact Smart Card (Get ATR)” is for reading ATR value from the smart card by internal contact smart card encoder. This is executed on OnBnClickedButtonIc () of SmartCommonTestDlg.cpp.

“Contactless Smart Card – Mifare Write and Read using PC/SC” write and read in Mifare card using an internal contactless smart card reader. This is executed on OnBnClickedButtonRf () of SmartCommonTestDlg.cpp.

And CSD Document group is for loading CSD files and shows printing images.

If you click “Open CSD Document” button, it reads the CSD file using SDK function. The contents read are displayed at the bottom of the main window, and if the CSD file contains fields, the fields are listed in the list control. Figure 6 is appeared by double click on the list. When you insert the value on that window, the value is automatically applied and updated contents are displayed at the bottom of the main window.

“Close CSD Document” button closes the CSD file which is read by ‘Open CSD Document”

If you see the source code of SmartCommonTest, there are various examples for issuing cards such as printing and encoding so it will be helpful to understand “SMART Printer SDK”.

As the examples, “SMART Printer SDK” provides the same interface for local issuance and network

issuance. So if you develop the issuance program, the network issuance can be possible without additional development.

6.2 VB Source : SmartDLL.VB

In the “SMART Printer SDK”, SmartDLL.VB is using SMART Printer by SmartComm2.dl. SmartDLL.VB is based on Microsoft Visual Studio 6.0.

If executes SmartDLL.VB, the window such as figure 7 will be displayed. Printer descriptions are displayed on “Printer” list and you can choose a printer which will be used. After choosing a printer, you can test SMART Printer with this program.

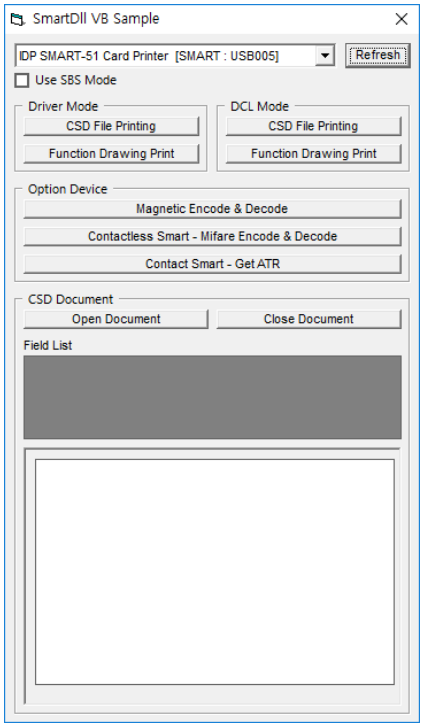

Figure 7 SmartDLL.VB

“Use SBS(Step-By-Step) Mode, When Print” checkbox decides whether you will use SBS mode or not. If you check the box, SBS mode will be set and it is applied during the printing process.

There are three buttons for 2 operating modes (Driver, DCL).

There are 3 ways to print in Driver mode.
“CSD File Print” is for printing CSD file which is made by SmartID.
“Function Drawing Print” is for printing the image and text using SmartComm_DrawXXX functions.

There are 3 ways to print in DCL mode.

“CSD File Print” is for printing CSD file which is made by SmartID.

“Function Drawing Print” is for printing the image and text using SmartComm_DrawXXX functions.

There are buttons to test option devices at the bottom.

“Magnetic Encode & Decode” button will open a device and record magnetic data to a card and compare magnetic data and memorized data and take the card out.

“Contactless Smart – Mifare Encode & Decode” button will open a device and record data to Mifare card by PC/SC protocol and take the card out.

“Contact Smart – Get ATR” button will open the device and display the IC card ATR value and take the card out.

And CSD Document group is for loading CSD files and shows printing images.

If you click “Open Document” button, it reads the CSD file using SDK function. The contents read are displayed at the bottom of the main window such as figure 8, and if the CSD file contains fields, the fields are listed in the list control. Figure 9 is appeared by double click on the list. When you insert the value on that window, the value is automatically applied and updated contents are displayed at the bottom of the main window.

“Close Documents” button closes the CSD file which is read by “Open Document”

If you see the source code of SmartDll VB Sample, there are various examples for issuing cards such as printing and encoding so it will be helpful to understand “SMART Printer SDK”.

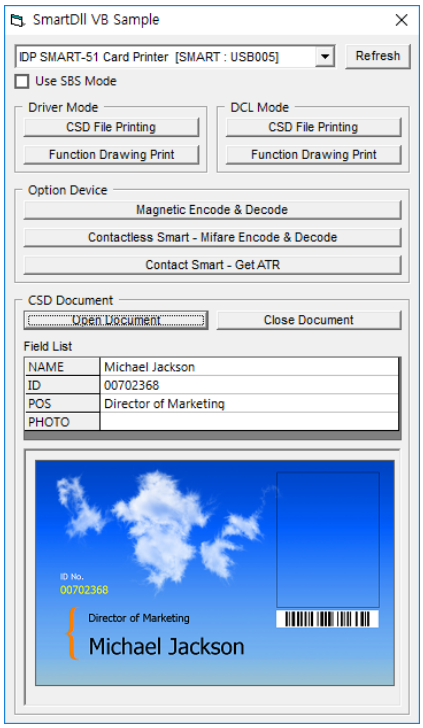

Figure 8 Open Document

As the examples, “SMART Printer SDK” provides the same interface for local issuance and network issuance. So if you develop the issuance program, the network issuance can be possible without additional development.

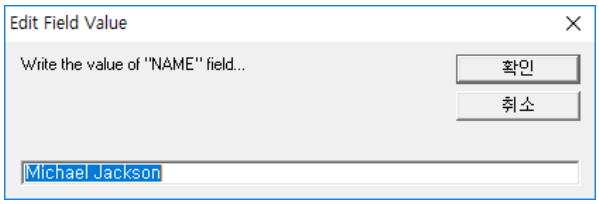

Figure 9 Input Field value

6.3 VB.NET Source : vb.NET Sample

vb.NET Sample in “SMART Printer SDK” is for operating SMART Printer using SmartComm2.dll. vb.NET Sample is developed in Microsoft Visual Studio 2005 VB Dot NET. Please be sure that you should use 32-bit platform because SmartComm2.dll is developed as a 32bit platform.

Entire sample is similar to VC++ sample program.

If you start the program, a window will be displayed as figure 10 and SMART Printer will be displayed in combo box at the top.

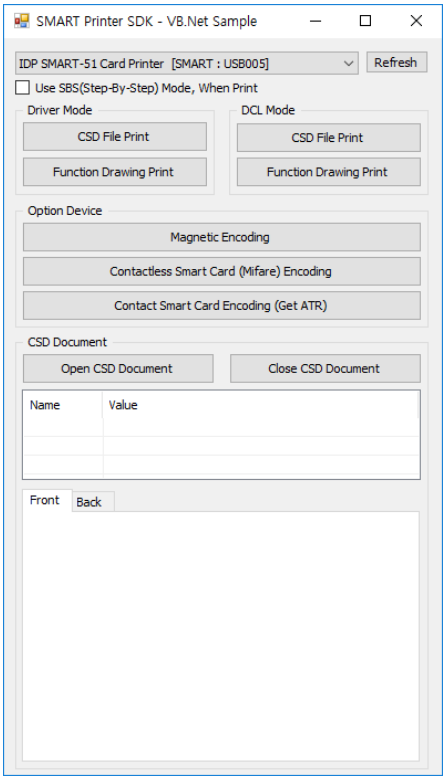

Figure 10 vb.Net Sample

After choosing a printer from the list, you can click buttons below and check that the functions work well.

“Use SBS(Step-By-Step) Mode, When Print” is for using SBS(Step-By-Step) mode during the printing process. If you click this checkbox, SBS mode is applied

There are two buttons for each operating mode (Driver, DCL).

There are two ways to print in Driver mode.

“CSD File Print” is for printing CSD file which is made by SmartID. It is executed on btnDriverPrintCSD_Click() of Form1.vb file.

“Function Drawing Print” is for printing the image and text using SmartComm_DrawXXX functions. It is executed on btnDriverPrintDrawFunc_Click() of Form1.vb file.

There are two ways to print in DCL mode.

“CSD File Print” is for printing CSD file which is made by SmartID. It is executed on btnDCLPrintCSD_Click() of Form1.vb file.

“Function Drawing Print” is for printing the image and text using SmartComm_DrawXXX functions. It is executed on btnDCLPrintDrawFunc_Click() of Form1.vb file.

There are buttons to test option devices at the bottom.

“Magnetic Encoding” is for testing the read/write of the magnetic stripe. This is executed on btnMagneticEncoding_Click() of Form1.vb file.

“Contactless Smart Card (Mifare) Encoding” write and read in Mifare card using an internal contactless smart card reader. This is executed on btnContactlessEncoding_Click() of Form1.vb file.

“Contact Smart Card Encoding (Get ATR)” is for reading ATR value from the smart card by internal contact smart card encoder. This is executed on btnContactEncoding_Click() of Form1.vb file.

And CSD Document group is for loading CSD files and shows printing images.

If you click “Open CSD Document” button, it reads the CSD file using SDK function. The contents read are displayed at the bottom of the main window, and if the CSD file contains fields, the fields are listed in the list control. Figure 11 is appeared by double-clicking on the list. When you insert the value on that window, the value is automatically applied and updated contents are displayed at the bottom of the main window.

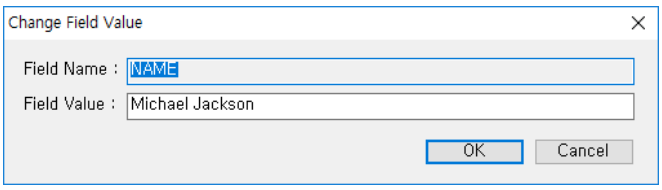

Figure 11 Edit Field Value

"Close CSD Document" button closes the CSD file which is read by "Open CSD Document"

If you see the source code of vb.NET, there are various examples for issuing cards such as printing and encoding so it will be helpful to understand "SMART Printer SDK".

As the examples, "SMART Printer SDK" provides the same interface for local issuance and network issuance. So if you develop the issuance program, the network issuance can be possible without additional development.

6.4 Java Source : SmartJavaClass

SmartJavaClass.jar Sample in "SMART Printer SDK" is for operating SMART Printer using SmartComm2.dll and SmartCommJNI.dll. SmartCommJNI.dll calls function those are included in SmartComm2.dll and enables the users can use the functions in Java program. This DLL is developed in Microsoft Visual Studio 2005 VC++. SmartJavaClass.jar Java sample is developed in NetBeans.

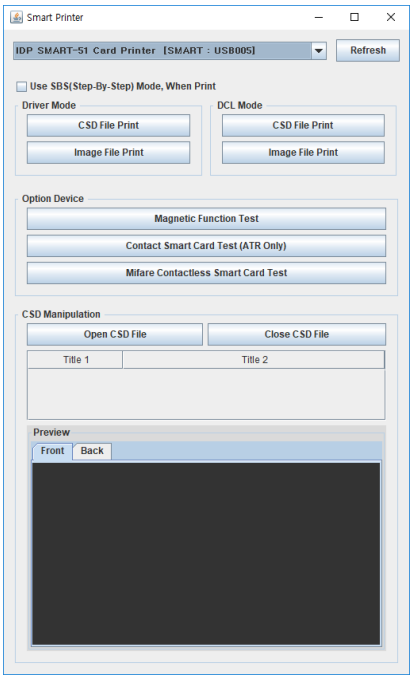

Figure 12 SmartJavaClass.jar

If you start SmartJavaClass.jar, a window will be displayed as figure 12 and SMART Printer will be displayed in combo box at the top. You can choose a printer from the list and test with the buttons below.

There are two buttons for each operating mode (Driver, DCL).

There are two ways to print in Driver mode.

“CSD File Print” is for printing CSD file which is made by SmartID. It is executed on jButtonCSDFilePrintActionPerformed() of SmartCommonTest.java file.

“Function Drawing Print” is for printing the image and text using SmartComm_DrawXXX functions. It is executed on jButtonFunctionDrawingPrintActionPerformed () of SmartCommonTest.java file.

There are two ways to print in DCL mode.

“CSD File Print” is for printing CSD file which is made by SmartID. It is executed on jButtonDCLCsdFilePrintActionPerformed () of SmartCommonTest.java file.

“Function Drawing Print” is for printing the image and text using SmartComm_DrawXXX functions. It is executed on jButtonDCLFunctionDrawingPrintActionPerformed () of SmartCommonTest.java file.

There are buttons to test option devices at the bottom.

“Magnetic Function Test” is for testing the read/write of the magnetic stripe. This is executed on jButtonMagneticWriteAndReadActionPerformed () of SmartCommonTest.java file.

“Contact Smart Card Test (ATR Only)” is for reading ATR value from the smart card by internal contact smart card encoder. This is executed on jButtonContactSmartCardTestActionPerformed () of SmartCommonTest.java file.

“Mifare Contactless Smart Card Test” writes and reads in Mifare card using an internal contactless smart card reader. This is executed on jButtonContaclessSmartCardTestActionPerformed () of SmartCommonTest.java file.

If you see the source code of SmartJavaTest.java, there are various examples for issuing cards such as printing and encoding so it will be helpful to understand “SMART Printer SDK”.