**Lithograph Module Dynamic Library Interface File**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **History of Revision** | | | | | |
| **No.** | **Date** | **Description of Revision** | **Version** | **Revised by** | **Checked by** |
| 1 | 20191126 | Initial draft | V1.0 | Fan Chen |  |
| 2 | 20191212 | 1. Revision of printing interface and additional content rotating parameters  2. Revision of Printer\_Open parameter description  3. Revision of Print\_Rotate interface function description  4. Additional description for call of complete printing process interface | V1.1 | Fan Chen |  |
| 3 | 20191226 | Additional card-retaining interface; redefined structural body and additional domain to prevent conflict | V1.2 | Fan Chen |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Table of Contents**

[1. Interface documentation 4](#_Toc29217780)

[2. Export interface 4](#_Toc29217781)

[3. Device type interface CLithographPrinter 5](#_Toc29217782)

[3.1 Device on 5](#_Toc29217783)

[3.2 Device off 5](#_Toc29217784)

[3.3 Resetting the device 5](#_Toc29217785)

[3.4 Get firmware version information 6](#_Toc29217786)

[3.5 Card enabled 6](#_Toc29217787)

[3.6 Card canceled 6](#_Toc29217788)

[3.7 Card returned to card slot 6](#_Toc29217789)

[3.8 Card retained in printer collection box 6](#_Toc29217790)

[3.9 Reading the track 7](#_Toc29217791)

[3.10 Writing the track 7](#_Toc29217792)

[3.11 IC card powered on 8](#_Toc29217793)

[3.12 IC card powered off 8](#_Toc29217794)

[3.13 IC card data exchange 8](#_Toc29217795)

[3.14 Dispensing cards from card box 8](#_Toc29217796)

[3.15 Get card box status 9](#_Toc29217797)

[3.16 Get printer status 9](#_Toc29217798)

[3.17 Printing initialization 9](#_Toc29217799)

[3.18 Pre-printing image 10](#_Toc29217800)

[3.19 Pre-printing text 10](#_Toc29217801)

[3.20 Pre-printing barcode 10](#_Toc29217802)

[3.21 Start printing 11](#_Toc29217803)

[3.22 Rotating 11](#_Toc29217804)

[3.23 Turning off sleep mode 11](#_Toc29217805)

[3.24 Expansion interface 12](#_Toc29217806)

[4. Printing instructions 12](#_Toc29217807)

[5. Definition of header file 13](#_Toc29217808)

# 1. Interface documentation

(1) The dynamic library name of lithograph module is unified as (vendor's name + "\_" + device name + "\_" + LithographPrinter.dll)，such as Evolis device modeled 656, which should be Evolis\_656\_LithographPrinter.dll.

(2) An error code list is to be provided. An error code contains 4 digits, in addition to description of the error, with unified format as (vendor's name + "\_" + device name + "\_" + ErrorCode.xml), such as Evolis\_656\_ErrorCode.xml, which is formated as the following:

<?xml version="1.0" encoding="utf-8"?> //format UTF8

<Evolis\_656> //vendor's name + "\_" + device name

<ErrorCode value ="0000"> //device error code

<ErrorDescription value ="successful"/> //error code description

</ErrorCode>

<ErrorCode value ="0001">

<ErrorDescription value ="loading dynamic library failed"/>

</ErrorCode>

...

...

...

</ Evolis\_656>

(3) For the type of interface achieved, interface characteristic description should be provided. For non-supportable interfaces or non-supportable parameters in an interface, they can be neglected, but further description should be provided.

(4) All interfaces are synchronous.

(5) The timeout length of all interfaces should be the default value. The timeout length of each vendor's interface should be provided with documentation.

# 2. Export interface

1. LPVOID WINAPI CreateInstance (LPVOID lpReserve);

Function：Create device instance; return device instance pointer and unify the device type name as CLithographPrinter"

An example of call：

typedef*LPVOID*(*WINAPI*\*fnCreateInstance)(LPVOID lpReserve); fnCreateInstance m\_pCreateInstance;

CLithographPrinter \* m\_pPrinter;

*HMODULE*hDll=*LoadLibrary*("XXX.dll");

if(*NULL*!=hDll)

{

pCreateInstance=(fnCreateInstance)*GetProcAddress*(hDll,"CreateInstance"); pPrinter = (CLithographPrinter \*)m\_ pCreateInstance(NULL);

}

1. void WINAPI FreeInstance(LPVOID lpDev);

Function: Release device instance pointer

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| LPVOID lpDev | IN | Device instance pointer |

# 3. Device type interface CLithographPrinter

class CLithographPrinter ()

{

CLithographPrinter () {};

~ CLithographPrinter () {}；

}

## 3.1 Device on

int Print\_Open (char \*pPort, char \*pPortParam, char \*pszRcCode);

Note: Successful return can be made when the device calls Open several times. (if already opened, do not throw any error)

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| pPort | IN | Port No., U port device: USB or usb; Serial port device: COM+ number of serial ports, such as COM1; Network port device: IP |
| pPortParam | IN | Port parameter, U port device: USB device parameter, VID\PID, or printer name, such as "Evlois"; Serial device：format being Baud Rate + parity check bit + start bit + stop bit，such as 57600,n,8,1; network device: IP address， such as 172.0.0.1 |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.2 Device off

int Print\_Close (char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.3 Resetting the device

int Print\_Reset (long lTimeout, int nResetAction, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| nResetAction | IN | 1-Reset - the card is retained in the device；2-Reset - the card is returned to the card slot；3-Reset - the card is retained into the printer's collection box (to be implemented if the hardware is supportable) |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.4 Get firmware version information

int Print\_GetVersion (long lTimeout, char \*pVersionInfo, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pVersionInfo | OUT | Firmware version information (firmware information shall include the vendor + device name + version info + date, such as Evolis656\_v1.0.1\_20190211) |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.5 Card enabled

= click on set on printer lcd to enable card insertion in right unit feed mode

int Print\_EnableCard(long lTimeout, int nCheckMode, char \*pszRcCode);

Note: the function is returned immediately. Successful return indicates card enabled.

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| nCheckMode | IN | Check magnetic track? 0- No; 1- Yes (to be implemented if it is supportable, or ignore the parameters)= check if the card is a mag one or not. (in case user inserts bad card type or wrong side inserted) no need, cf sean |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.6 Card canceled

= needed in case the card was not properly inserted (bug / error case) to avoid a reset on the printer.

int Print\_DisableCard(long lTimeout, char \*pszRcCode);

Note: the function is returned immediately. Successful return indicates card enabled. Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.7 Card returned to card slot

= eject when card is in stand by mode

int Print\_Eject (long lTimeout, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.8 Card retained in printer collection box

= reject, box number = exit reject / error setting

int Print\_Retract(long lTimeout, int nBoxNo, char \*pszRcCode)

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| nBoxNo | IN | Card collection box No. |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.9 Reading the track

int Print\_ReadTrack(long lTimeout, WORD wTrackId, MagCardInfo \*pTrackInfo, char\*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| wTrackId | IN | Track to read, 0x0001- read track 1; 0x0002- read track 2; 0x0004- read track 3. To read multiple tracks, xor will do. For example, to read tracks 1, 2 and 3, input 0x0007. |
| pTrackInfo | OUT | For specific track readings see Definition of structural body. |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

Definition:

typedef struct \_MagCard\_

{

unsigned char track1\_len; //TRACK1 length

unsigned char track2\_len; //TRACK2 length

unsigned char track3\_len; //TRACK3 length

unsigned char track1\_data[80]; //TRACK1 data

unsigned char track2\_data[80]; //TRACK2 data

unsigned char track3\_data[170]; //TRACK3 data

WORDwTrack1Method; //TRACK1 writing method, low resistance 0x0002, high resistance 0x0004, auto 0x0008

WORDwTrack2Method; //TRACK2 writing method, low resistance 0x0002, high resistance 0x0004, auto 0x0008

WORDwTrack3Method; //TRACK3 writing method, low resistance 0x0002, high resistance 0x0004, auto 0x0008

}MagCardInfo,\*LPMagCardInfo;

## 3.10 Writing the track

int Print\_WriteTrack (long lTimeout, WORD wTrackId, MagCardInfo \*pTrackInfo, char\*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| wTrackId | IN | Track to write, 0x0001- write track 1; 0x0002-write track 2; 0x0004-write track 3. To write multiple tracks, xor will do. For example, to write tracks 1, 2 and 3, input 0x0007. |
| pTrackInfo | IN | Track data to write and definition of the structural body are the same as for track-reading interface. |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.11 IC card powered on

3.11, 12,13 means that if they do not use an Evolis encoder, we just have to send the card in encoding position instead of the three functions below

int Print\_IcPowerOn(long lTimeout, BYTE \*pOutAtr, int &nAtrlen, char \*pUID, char\*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pOutAtr | OUT | ATR data HEX |
| nAtrlen | OUT | ATR data length |
| pUID | OUT | UID should be returned when non-connection is used, or null is returned. |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.12 IC card powered off

int Print\_IcPowerOff (long lTimeout, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.13 IC card data exchange

int Print\_IcExchange (long lTimeout, BYTE \*pIndata, int nInDataLen, BYTE \*pOutData, int &nOutLen, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pIndata | IN | Input command HEX |
| nInDataLen | IN | Input command length |
| pOutData | OUT | Response data HEX |
| nOutLen | OUT | Response data length |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.14 Dispensing cards from card box

= send from a source, to IC contact or contactless, but not possible to mag / print

int Print\_Dispense(long lTimeout, int nBox, int nDispPos, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| nBox | IN | Card collection box No. |
| nDispPos | IN | Car dispensing position  1- Card dispensed to track-reading position;  2- Card dispensed to contact IC card position;  3- Card dispensed to non-contact IC card position;  4- Card dispensed to printing position. |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.15 Get card box status

int Print\_GetBoxStatus (long lTimeout, *LPBOXINFO*&lpBoxInfo, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| lpBoxInfo | OUT | For structural body of card box see the Definition. |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

Definition:

typedef struct \_LITHOGRAPHBOXUNIT

{

BYTE byType; //card box type 0:unknown，1：card dispenser = feeder，2：collection box = reject

BYTE byBoxNumber; //card box No.

BYTE byStatus; //card box status, 0：normal; 1：less cards; 2： no card; 3：non-operable; 4：absent; 5：high(nearly full)；6：full； 7： unknown

BOOL bHardwareSensor; //Support failure state sensor? TRUE： Yes; FALSE：No

}LITHOGRAPHBOXUNIT,\*LPLITHOGRAPHBOXUNIT;

typedef struct \_LITHOGRAPHBOXINFO

{

int nCount; //Total card boxes (all card boxes)

LPLITHOGRAPHBOXUNIT lpplist; //card box information structure pointer

}LITHOGRAPHBOXINFO,\*LPLITHOGRAPHBOXINFO;

## 3.16 Get printer status

int Print\_Status(long lTimeout, *LPPRINTERSTATUS&lpStatus*, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| *lpStatus* | OUT | For structural body of printer see the Definition. |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

Definition:

typedef struct \_LITHOGRAPHSTATUS

{

WORD fwDevice; //Printer status, 0- online；1- operating；2- offline；3- failure

WORD fwMedia; //medium status，0- no card；1- card at door；2- card held inside；3- card power-on，4-card beyond the gate；5- card seized；6- card unknown（It is returned according to hardware characteristics. It must support checking the presence of card. ）

WORD fwToner; //lithograph tape status, 0-FLLL; 1-LOW; 2-OUT; 3-NOTSUPP; 4-UNKNOW

}LITHOGRAPHSTATUS,\*LPLITHOGRAPHSTATUS;

## 3.17 Printing initialization

= clear any error status and have the printer ready to print (in most cases, the printer will already be ready, just in case door opened for example, or need to synchronize ribbon)

Note: It is called before printing any image or text and it is only called once before each printing.

int Print\_InitPrint(long lTimeout, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.18 Pre-printing image

= Adding image data to the print job

int Print\_PrintImage (long lTimeout, char \*pImage, int nAngle, float fxPos, float fyPos, float fHeight, float fWidth, char \*pszRcCode);

Note: Printing must be initialized.

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| *pImage* | IN | Full path of image |
| nAngle | IN | Clockwise rotation angle of image, such as 90. This parameter does not rotate the card. |
| *fxPos* | IN | Print coordinate X |
| *fyPos* | IN | Print coordinate Y |
| *fHeight* | IN | Printing height |
| *fWidth* | IN | Printing width |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.19 Pre-printing text

int Print\_PrintText (long lTimeout, char \*pText, int nAngle, float fxPos, float fyPos, char \*pFontName, int nFontSize, int nFontStyle, int nColor, char \*pszRcCode);

Note: Printing must be initialized.

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| *pText* | IN | Print text content. |
| nAngle | IN | Clockwise rotation angle, used for printing horizontal or longitudinal text. |
| *fxPos* | IN | Print coordinate X |
| *fyPos* | IN | Print coordinate Y |
| *pFontName* | IN | Printing font, such as "Song typeface" |
| *nFontSize* | IN | Font size |
| *nFontStyle* | IN | Font style：1- regular；2- bold；4- italic；8-boldface |
| *nColor* | IN | Print color, RGB color value |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.20 Pre-printing barcode

= transform to barcoide to be given by GWI

int Print\_PrintBarcode(long lTimeout, int nAngle, char \*pBarcodeType, *loat fxPos, float fyP*os, int nMultiplier, int nHeight, int nIsActive, char \*pBarcodeText, char\*pszRcCode);

Note: Printing must be initialized.

Returned value：0- Successful；1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| nAngle | IN | Barcode text rotation angle, horizontal printing barcode or longitudinal printing |
| pBarcodeType | IN | Type of barcode, such as "c39", "2/5", "c128", "c128c", "ean13", etc. Please note it in the documentation if other types of barcode are supported. |
| *fxPos* | IN | Coordinate X of barcode |
| *fyP*os | IN | Coordinate Y of barcode |
| nMultiplier | IN | Clock multiplier factor |
| nHeight | IN | Barcode height |
| nIsActive | IN | Data value activation, 0 deactivate, 1 activate |
| pBarcodeText | IN | Printed barcode data |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.21 Start printing

Note: Start printing by submitting image and text printing task.

int Print\_StartPrint (long lTimeout, char \*pszRcCode);

Returned value: 0-Successful; 1-Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.22 Rotating

Note: Used for card rotation

int Print\_Rotate (long lTimeout, int nAngle, *char \*pszRcCode*);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| nAngle | IN | Used for turning the card. Generally, it only turn the card by 180 degrees. |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.23 Turning off sleep mode

int Print\_CloseSleepMode(long lTimeout, char \*pszRcCode);

Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| lTimeout | IN | Timeout period in ms |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

## 3.24 Expansion interface

int Print\_ExtraCommand(long lTimeout, char \*pCommand, LPVOID lpCmdIn, LPVOID &lpCmdOut, char \*pszRcCode) Returned value: 0- Successful; 1- Failed

|  |  |  |
| --- | --- | --- |
| Parameter | IN/OUT | Description |
| pCommand | IN | Command character string |
| lpCmdIn | IN | Command parameter |
| lpCmdOut | OUT | Output data |
| pszRcCode | OUT | A 4-bit error code will be returned if failed and "0000" if it is successful. |

1. Get the ability to support or not printing on both sides at the same time.

Input parameter: pCommand, the parameter value is " PrintBothSideSameTime "

Input parameter：lpCmdIn：the value is NULL

Output parameter：lpOutValue：parameter type \*BOOL，TRUE indicates support

Call instance：

LPBOOL pbPrintBothSide;

int nResult = Print\_ExtraCommand(" PrintBothSideSameTime ",NULL,

pbPrintBothSide,pszRcCode);

# 4. Printing instructions

1. Coordinate instructions

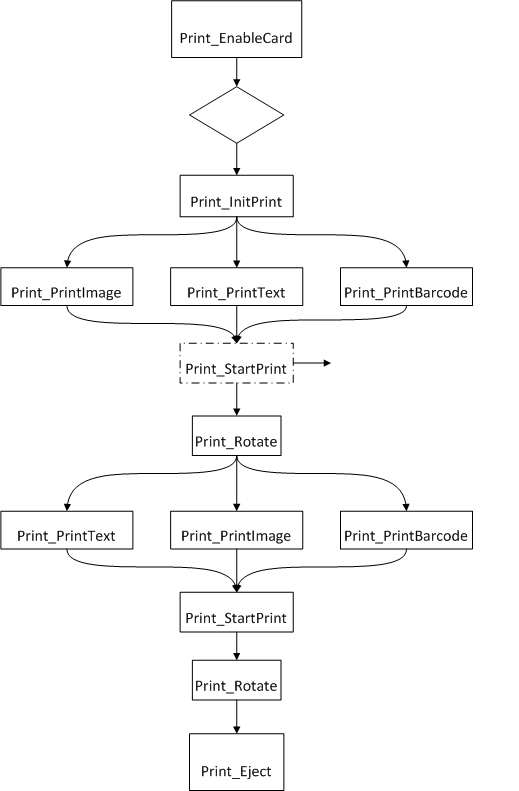
Card width：85.60mm / 3.37inch / 1024 point 🡪 1016x648 / 2048x1300 px

Card height：53.98mm / 2.13inch / 640 point

Front：Coordinates from（0，0）to（1024, 640）

Back：Coordinates from （0, 700）to（1024, 1340）

2. Printing process



Pre-printing image (multi-calls)

Printing initializing

Pre-printing barcode (multi-calls)

Pre-printing text (multi-calls)

Start printing

Card rotation

If printing on both front and back sides is supported, it is unnecessary to call Print\_StartPrint interface. Printing on the back side is determined via Print\_Rotate.

Pre-printing barcode (multi-calls)

Pre-printing text (multi-calls)

Pre-printing image (multi-calls)

Start printing

Card rotation

Card ejected

Card detected

Card enabled

# 5. Definition of header file



LithographPrinter.h