



SEIKO INSTRUMENTS USA INC.

SMART LABEL PRINTER

SOFTWARE DEVELOPMENT KIT

1.2

DOCUMENTATION

SLP 100/410

SLP 200/420

SLP 240/430

SLP 440

SLP 450

© 2005 Seiko Instruments USA Inc.

Contributors

Written by Sanford Selznick and Dave Koziol
Production by Selznick Scientific Software, LLC
Engineering contributions by Dave Koziol and Sanford Selznick

This SII software is supplied to you by Seiko Instruments USA Inc. ("SII") in consideration of your agreement to the following terms, and your use, installation, modification or redistribution of this SII software constitutes acceptance of these terms. If you do not agree with these terms, please do not use, install, modify or redistribute this SII software.

In consideration of your agreement to abide by the following terms, and subject to these terms, SII grants you a personal, non-exclusive license, under SII's copyrights in this original SII software (the "SII Software"), to use, reproduce, modify and redistribute the SII Software, with or without modifications, in source and/or binary forms; provided that if you redistribute the SII Software in its entirety and without modifications, you must retain this notice and the following text and disclaimers in all such redistributions of the SII Software. Neither the name, trademarks, service marks or logos of SII may be used to endorse or promote products derived from the SII Software without specific prior written permission from SII. Except as expressly stated in this notice, no other rights or licenses, express or implied, are granted by SII herein, including but not limited to any patent rights that may be infringed by your derivative works or by other works in which the SII Software may be incorporated.

The SII Software is provided by SII on an "AS IS" basis. SII MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE SII SOFTWARE OR ITS USE AND OPERATION ALONE OR IN COMBINATION WITH YOUR PRODUCTS.

IN NO EVENT SHALL SII BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) ARISING IN ANY WAY OUT OF THE USE, REPRODUCTION, MODIFICATION AND/OR DISTRIBUTION OF THE SII SOFTWARE, HOWEVER CAUSED AND WHETHER UNDER THEORY OF CONTRACT, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, EVEN IF SII HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Seiko Instruments USA Inc.
www.siibusinessproducts.com

Table of Contents

Contributors	2
Table of Contents	3
About this Document.....	4
Revision History.....	4
Compatibility	4
Installation	6
Distribution Checklist.....	7
SIISDK API Quick Start.....	8
SII SDK API Details	10
SIISLP_Initialize.....	10
SIISLP_ChangeCallbackPtr.....	11
SIISLP_Finalize	11
SIISLP_FindNext.....	11
SIISLP_GetIDFromReference	12
SIISLP_GetReferenceFromID	13
SIISLP_GetTextDescription	13
SIISLP_Open	14
SIISLP_Close.....	14
SIISLP_GetStatus.....	14
SIISLP_Print	15
SIISLP_PrintAsync	16
SIISLP_PrintFinishedCallbackPtr.....	17
SIISLP_CancelAllAsync	17
SIISLP_TopOfForm	17
SIISLP_GetMaxDotsPerLine.....	18
SIISLP_GetDotsPerInch.....	18
SIISLP_SetDensity.....	18
SIISLP_SetGraphicsMode.....	19
SIISLP_GetFirmwareVersion.....	19
SIISLP_GetLabelTypes.....	20
SIIUtilities.mm.....	20
NSImageToBitMap	20
GWorldToBitMap	21
BitMapCreate	21
BitMapDispose.....	22
BitMapRotate90	22
Index.....	23

About this Document

This document describes the Seiko Instruments USA Inc. (SII) Smart Label Printer (SLP) Software Development Kit (SDK) for Mac OS X. Described herein is a C Application Programmer Interface (API) through which application developers can add SLP label printing functionality directly to their applications.

Revision History

- Beta 2
 - Changed SIISLP_GetMaxDotsPerLine to return its value as an SInt32 rather than a UInt32.
 - Updated documentation for SIISLP_PrintFinishedCallbackPtr.
- v1.0
 - Now detecting USB topology change in 10.2.
- v1.1
 - Updated all label sizes in plist.
 - Universal binary is now compatible with both PowerPC and Intel.
 - Added functionality to stop the Mac OS X Classic Environment from interfering with the SIISLP_Open command.
 - Added SIISLP_SetGraphicsMode function.
- v1.2
 - All printing is now faster. The print pathway is highly compressed and optimized to leverage each printer model.

Compatibility

The SII SLP SDK framework is compatible with Mac OS X versions 10.2.8 and later. This SDK has been tested through 10.4.

- Although the SII SLP SDK *framework* may work well under earlier versions of 10.2, 10.3, and 10.4, if end-users are experiencing problems, they should be advised to update to the latest versions of 10.2, 10.3 or 10.4. These updates are available for free from the Software Update panel from within System Preferences.
- CFM applications may load the framework bundle dynamically using Mac OS X mechanisms available in CarbonLib.
- The framework has been tested with Xcode 2.2 from Apple Computer, Inc. and CodeWarrior 9.3 from Metrowerks, Inc. Programmers should refer to their environment's documentation for instructions on linking their Carbon or Cocoa application with the SIISeikoLabelPrinter.framework located in /Library/Frameworks.

- Label printing for Mac OS 9 native or from within the Classic Environment is not supported at this time.
- The SII SLP SDK sample project is intended for use with Xcode 2.2. This limits building the sample application to Mac OS X 10.3 and 10.4. The sample application, once built, will work under both 10.2 and later.

Installation

The SII SLP SDK is available for free from <http://www.siibusinessproducts.com/sdk/mac/index.html> as a Mac OS X compatible Disk Image. Contained on the disk image is an installer that installs the following items at the indicated paths:

`/Library/Frameworks/SIISmartLabelPrinter.framework`

This file is the framework that applications will link with to communicate with Seiko Label Printers.

`/Developer/Examples/SIISmartLabelPrinterSDK`

This folder contains an application sample that prints NSImages and GWorlds, this document, a ReadMe file, and an SIIUtilities.mm file for creation, destruction and conversion to BitMaps.

Contained herein are the following files:

1. SII SLP SDK Documentation. (This document.)
2. A Sample directory that contains an Xcode project for an application that prints labels from Cocoa NSImages and Carbon GWorlds. (The example also contains useful utilities.)
3. A ReadMe document.

`/Library/System/Extensions/SIISmartLabelPrinter.kext`

With this Kext present, and loaded by the system at startup, the Mac OS X Classic Environment will never occupy USB channels necessary for native Mac OS X software to open a connection to the printer. See the Distribution Checklist and SIISLP_Open for more details.

Distribution Checklist

The `SIISmartLabelPrinter.framework` is the only item that needs to be installed for applications to print directly to Smart Label Printers. Applications or Application Installers should install the framework before their application launches.

The `SIISmartLabelPrinter.kext` can be installed to prevent interference from the Mac OS X Classic Environment. Although the `SIISmartLabelPrinter.framework` can prevent such interference without the `kext` installed, installing `kext` is more efficient. A system restart is required to load the `kext`. See `SIISLP_Open` for more details.

Installer authors should be careful to not overwrite newer versions of the framework when installing.

SIISDK API Quick Start

The SIISmartLabelPrinter.framework uses state of the art techniques to allow applications to communicate directly with Seiko Label Printers. Through a series of function calls, application programmers can do the following, and more:

1. Obtain a list of available printers complete with user-readable names (SIISLP_FindNext, SIISLP_GetTextDescription).
2. Save a given printer for use after a system restart (SIISLP_GetReferenceFromID, SIISLP_GetIDFromReference).
3. Print a label synchronously (SIISLP_Print.)
4. Print a label asynchronously (SIISLP_PrintAsync.)
5. Query and Set information about a given printer (SIISLP_GetStatus, SIISLP_TopOfForm, SIISLP_GetMaxDotsPerLine, SIISLP_SetDensity, SIISLP_GetFirmwareVersion)
6. Cancel all asynchronous/queued printing operations (SIISLP_CancelAllAsync).

All Seiko Label Printers are 1-bit black and white printers. To perform printing, pass a printer reference and a BitMap structure (containing a graphics buffer, rowBytes and rectangle) to the SIISLP_Print or SIISLP_PrintAsync functions. To create BitMaps, programmers may use the functions we've included as part of our Sample code within the file SIIUtilities.mm:

1. To allocate and create a BitMap from an NSImage (NSImageToBitMap).
2. To allocate and create a BitMap from a GWorld (GWorldToBitMap).
3. To rotate* a BitMap 90 degrees to the right (BitMapRotate90).
4. Sample code is also provided for creating and disposing BitMap structures. (BitMapCreate, BitMapDispose).

*By rotating a bitmap just before printing, developers can lay out their GWorlds or NSImages horizontally using standard techniques.

A simple label printing session will look like this:

Initialization and Finalization:

```
int main()
{
    SIISLP_Initialize(NULL, NULL);

    // Do stuff

    SIISLP_Finalize();
    return 1;
}
```

And printing:

```
void printLabel(NSImage *image, SIISLP_LabelPrinterRef printer)
{
    SIISLP_Open(printer);

    BitMap bitmap = {0};
    NSImageToBitMap(image, bitmap);
    BitMapRotate90(bitmap, bitmap);

    // SIISLP_Print will not return until the label
    // has finished printing.
    SIISLP_Print(printer, bitmap);

    SIISLP_Close(printer);
}
```

These functions, and more, are described in great detail in the section “SII SDK API Details” (Page 10).

SIISDK API Details

Contained herein is a detailed description of the SIISDK API. The API is intended for use in C, Objective-C, C++ or Objective C++ applications for Mac OS X.

SIISLP_Initialize

Purpose

Initialize the SIISmartLabelPrinter.framework and prepare it for use. This must be called before any other SDK function is called. SIISLP_Initialize must not be called more than once per process.

Prototype

```
OSStatus SIISLP_Initialize(SIISLP_ChangeCallbackPtr changeCallback, long refCon);
```

Parameters

changeCallback	A function pointer to a callback routine. This argument may be NULL. The callback routine is called when the user changes the USB topology by physically adding or removing a printer. See below for a detailed description of SIISLP_ChangeCallbackPtr
refCon	A 32-bit constant that will be passed, as is, to the changeCallback function. This value may be NULL.

Returns

noErr upon successful completion of the routine.

Sample Usage

```
void mySIISLP_Callback(  
SIISLP_LabelPrinterRef labelPrinter,  
SIISLP_ChangeType changeType,  
long refCon)  
{  
    switch (changeType) {  
        case kSIISLPPrinterAdded:  
            break;  
        case kSIISLPPrinterRemoved:  
            break;  
    }  
}  
  
OSStatus status = SIISLP_Initialize(mySIISLP_Callback, NULL);
```

SIISLP_ChangeCallbackPtr

Purpose

An optional function of this type will be called when the user adds or removes a USB device from their computer. A function of this type is passed to SIISLP_Initialize. **This callback may not be called for changes made to non-powered USB ports on 10.2 or 10.3.**

Prototype

```
typedef void (*SIISLP_ChangeCallbackPtr)(SIISLP_LabelPrinterRef  
labelPrinter, SIISLP_ChangeType changeType, long refCon);
```

Parameters

labelPrinter	A reference to the label printer that has been added or removed. If the printer was removed, labelPrinter is a dead reference and cannot be used as parameters to any other framework functions.
changeType	One of three values <ol style="list-style-type: none">1. kSIISLPPrinterAdded: A printer has been added.2. kSIISLPPrinterRemoved: A printer has been removed.
refCon	The refCon value passed to SIISLP_Initialize.

SIISLP_Finalize

Purpose

Inform the SIISmartLabelPrinter.framework to free any of its allocated resources. No functions within the framework should be called after SIISLP_Finalize is called. SIISLP_Finalize must not be called more than once per process.

Prototype

```
OSStatus SIISLP_Finalize();
```

Parameters

None

Result

noErr upon successful completion of the routine.

SIISLP_FindNext

Purpose

Pass a pointer to a NULL reference to get the first printer. Pass the address of a valid reference to get the reference to the next printer.

Prototype

```
OSStatus SIISLP_FindNext(SIISLP_LabelPrinterRef *nextLabelPrinter);
```

Parameters

nextLabelPrinter The address of a `SIISLP_LabelPrinterRef`. Set the value of the `SIISLP_LabelPrinterRef` to `NULL` to have it replaced with a reference to the first available printer. Call with a valid printer to get the next attached printer.

Result

Returns `kUSBNoDeviceErr` if there are no more printers available. Returns `noErr` upon successful completion of the routine.

Sample

```
SIISLP_LabelPrinterRef nextLabelPrinter = NULL;

do {
    err = SIISLP_FindNext(&nextLabelPrinter);
    if (err) {
        break;
    }

    char printerName[256] = "";
    err = SIISLP_GetTextDescription(nextLabelPrinter,
        printerName, sizeof(printerName));
    if (err) {
        break;
    }

    SIISLP_LabelPrinterID printerID = ""; // just a C-string
    err = SIISLP_GetIDFromReference(nextLabelPrinter,
        printerID);
    if (err) {
        break;
    }

    // Do something with printerName and printerID here
} while (not err);
```

SIISLP_GetIDFromReference

Purpose

Obtain a non-volatile ID for a specific printer from its reference. Non-volatile IDs can be saved to disk and used later to find the same printer, including after a reboot. IDs may not work to find printers if changes are made to the USB topology between invocations of the `SIISmartLabelPrinter.framework`.

Prototype

```
OSStatus     SIISLP_GetIDFromReference(SIISLP_LabelPrinterRef
labelPrinter, SIISLP_LabelPrinterID uniqueIdentifier);
```

Parameters

labelPrinter A valid reference to a label printer.

uniqueIdentifier Upon successful completion of the routine, contains the non-volatile ID of the printer. SIISLP_LabelPrinterID is an ASCII character buffer.

Result

noErr upon successful completion of the routine.

SIISLP_GetReferenceFromID

Purpose

Obtain a reference to a specific printer from a non-volatile ID. It may not be possible to always find the same printer if the USB topology has changed since the call to SIISLP_GetIDFromReference.

Prototype

```
OSStatus      SIISLP_GetReferenceFromID(SIISLP_LabelPrinterID
uniqueIdentifier, SIISLP_LabelPrinterRef *labelPrinter);
```

Parameters

uniqueIdentifier An SIISLP_LabelPrinterID returned from SIISLP_GetIDFromReference.

labelPrinter Contains a reference to the printer upon successful completion of the routine,

Result

noErr upon successful completion of the routine.

SIISLP_GetTextDescription

Purpose

Get a human-readable, ASCII, textual description of the printer. This function should be used to generate text for on-screen lists or popup menus of available printers. Printer names are smaller than 128 bytes.

Prototype

```
OSStatus      SIISLP_GetTextDescription(SIISLP_LabelPrinterRef
labelPrinter, char *buffer, size_t bufSize);
```

Parameters

labelPrinter A valid label printer reference.

buffer A pointer to a character buffer to receive the human-readable textual description of the printer.

bufSize The number of bytes available in buffer.

Result

noErr upon successful completion. An error will be returned in bufSize is too small to hold the description.

SIISLP_Open

Purpose

Opens a connection to the specified printer.

Prototype

```
OSStatus SIISLP_Open(SIISLP_LabelPrinterRef labelPrinter);
```

Parameters

labelPrinter A valid label printer reference returned by SIISLP_FindNext or SIISLP_GetReferenceFromID.

Result

noErr upon successful completion of the routine.

Notes

If the Mac OS X Classic Environment is running, it may interfere with SIISLP_Open succeeding. (Classic temporarily opens connections to all USB devices in short intervals.) SIISLP_Open will try to open a connection with the specified printer for up to 5 seconds. If the SIISmartLabelPrinter.kext is installed, the Mac OS X Classic Environment will never interfere with SIISLP_Open.

SIISLP_Close

Purpose

Closes the connection to the specified printer.

Prototype

```
OSStatus SIISLP_Close(SIISLP_LabelPrinterRef labelPrinter);
```

Parameters

labelPrinter A valid label printer reference that was previously opened with SIISLP_Open.

Result

noErr upon successful completion.

SIISLP_GetStatus

Purpose

Get the status of a label printer. The printer must be open before attempting to get its status.

Prototype

```
OSStatus      SIISLP_GetStatus(SIISLP_LabelPrinterRef labelPrinter,  
SIISLP_Status *status);
```

Parameters

labelPrinter A valid reference to an open label printer.

status A pointer to a variable of type SIISLP_Status. Upon return this variable will contain the status of the label printer.

Result

noErr upon successful completion of the routine.

Possible values returned in status are:

- kSIISLPIdle – Printer is idle,
- kSIISLPBusy – Printer is busy,
- kSIISLPerrNoLabels – Printer is out of labels.
- kSIISLPerrPaperJam – Printer is jammed.
- kSIISLPerrPlatenOpen – Printer platen is open. (That’s the roller.)
- kSIISLPerrHardwareOther – Unrecognized hardware error.
- kSIISLPerrCommunications – Can’t communicate with printer.
- kSIISLPerrUnknown – Unknown error.

SIISLP_Print

Purpose

Print the specified bitMap to the labelPrinter synchronously (this routine will not return until the label has been printed completely). The printer must be open before attempting to print. Note that printing cannot be canceled. See SIISLP_PrintAsync for asynchronous, cancellable, printing options.

Prototype

```
OSStatus      SIISLP_Print(SIISLP_LabelPrinterRef labelPrinter, BitMap  
*bitMap);
```

Parameters

labelPrinter A valid reference to an open label printer.

bitMap A pointer to a valid bitMap structure that contains the BitMap to print.

Result

noErr upon successful completion of the routine.

SIISLP_PrintAsync

Purpose

Print the specified bitMap to the labelPrinter asynchronously. The printer must be open before attempting to print. **The bitMap may NOT be freed by the caller until the SIISLP_ChangeCallback routine has been called with either kSIISLPPrintFinished or kSIISLPPrintFailed.**

Multiple labels may be queued to this routine. It is acceptable to queue labels from within the finishedCallback; before calling SIISLP_PrintAsync, the status of the printer should be kSIISLP_Busy or kSIISLP_Idle. Recall that “Busy” refers to the state of the printer, not the framework.

Neither SIISLP_Close or SIISLP_Finalize should be called from inside the finishedCallback.

Prototype

```
OSStatus SIISLP_PrintAsync(SIISLP_LabelPrinterRef labelPrinter,
BitMap *bitMap, SIISLP_PrintFinishedCallbackPtr finishedCallback, long
refCon);
```

Parameters

labelPrinter	A valid reference to an open label printer.
bitMap	A pointer to a valid bitMap structure that contains the BitMap to print.
finishedCallback	A function pointer to a callback routine. This argument may be NULL. The callback routine is called when the printer has finished printing a label or an error has occurred that prevents the label printing from completing. See below for a detailed description of SIISLP_PrintFinishedCallbackPtr.
refCon	A 32-bit constant that will be passed, as is, to the finishedCallback function. This value may be NULL.

Result

noErr upon successful completion of the routine.

See Also

SIISLP_CancelAllAsync

Sample Usage

```
void mySIISLP_FinishedCallback(
SIISLP_LabelPrinterRef labelPrinter,
OSStatus result,
long refCon)
{
    // Handle printing finished here.
    // Idea: Pass a pointer to printed bitmap in refcon
    // so the label can be freed when printing is finished.
```

```

}

SIISLP_LabelPrinterRef printerRef = MyPrinter();
BitMap *bitMap = MyBitMap();
OSStatus status = SIISLP_PrintAsync(printerRef, bitMap,
mySIISLP_FinishedCallback, NULL);

```

SIISLP_PrintFinishedCallbackPtr

Purpose

An optional function of this type will be called when the printer finishes printing a label that was initiated with the SIISLP_PrintAsync command.

Prototype

```
typedef void (*SIISLP_PrintFinishedCallbackPtr) (SIISLP_LabelPrinterRef
labelPrinter, OSStatus result, long refCon);
```

Parameters

labelPrinter	A reference to a label printer that has finished its printing.
result	The result of the print operation.
refCon	The refCon value passed to SIISLP_PrintAsync.

SIISLP_CancelAllAsync

Purpose

Cancels any and all outstanding asynchronous print requests. The SIISLP_PrintFinishedCallbackPtr function pointer **will still be called** for each submitted Print request, however the callbacks may not be called in the same order that the jobs were submitted.

Prototype

```
OSStatus SIISLP_CancelAllAsync(SIISLP_LabelPrinterRef labelPrinter);
```

Parameters

labelPrinter	A valid reference to an open label printer.
--------------	---

Result

noErr upon successful completion of the routine.

SIISLP_TopOfForm

Purpose

Advance the printer to the top of the next label. If the label is already top, don't advance. The printer must be open before calling this function.

Prototype

```
OSStatus SIISLP_TopOfForm(SIISLP_LabelPrinterRef labelPrinter);
```

Parameters

labelPrinter A valid reference to an open label printer.

Result

noErr upon successful completion of the routine.

SIISLP_GetMaxDotsPerLine

Purpose

Returns the maximum dots that can be printed on a single line

Prototype

```
OSStatus        SIISLP_GetMaxDotsPerLine(SIISLP_LabelPrinterRef  
labelPrinter, UInt32 *maxDotsPerLine);
```

Parameters

labelPrinter A valid reference to an open label printer.

maxDotsPerLine The address of an unsigned-32-bit integer to receive the
maxDotsPerLine value,

Result

noErr upon successful completion of the routine.

SIISLP_GetDotsPerInch

Purpose

Returns the dots per inch for a printer.

Prototype

```
OSStatus        SIISLP_GetDotsPerInch(SIISLP_LabelPrinterRef labelPrinter,  
UInt16 *dpi);
```

Parameters

labelPrinter A valid reference to an open label printer.

dpi The address of an unsigned-16-bit integer to receive the dots per
inch of the printer.

Result

noErr upon successful completion of the routine.

SIISLP_SetDensity

Purpose

Sets the desired densityPercentage between 70 and 130.

Prototype

```
OSStatus      SIISLP_SetDensity(SIISLP_LabelPrinterRef labelPrinter,  
SInt16 densityPercentage);
```

Parameters

labelPrinter A valid reference to an open label printer.

densityPercentage The address of a signed-16-bit integer between 70 and 130 to specify the density percentage value.

Result

noErr upon successful completion of the routine.

SIISLP_SetGraphicsMode

Purpose

Enables or Disables graphics mode in the printer on or off. In GraphicsMode the printer prints more slowly so the print head is less likely to become unaligned. GraphicsMode is disabled by default.

Prototype

```
OSStatus      SIISLP_SetDensity(SIISLP_LabelPrinterRef labelPrinter, int  
graphicsMode);
```

Parameters

labelPrinter A valid reference to an open label printer.

graphicsMode 1 to enable GraphicsMode. 0 to disable GraphicsMode.

Result

noErr upon successful completion of the routine.

SIISLP_GetFirmwareVersion

Purpose

Get an ASCII description of the printer firmware version.

Prototype

```
OSStatus      SIISLP_GetFirmwareVersion(SIISLP_LabelPrinterRef  
labelPrinter, char *buffer, size_t bufSize);
```

Parameters

labelPrinter A valid reference to an open label printer.

buffer The address of a character buffer into which the firmware version will be copied.

bufSize The size, in bytes, of buffer.

Result

noErr upon successful completion of the routine.

SIISLP_GetLabelTypes

Purpose

To obtain, from the framework, a list of label sizes and their descriptions.

Prototype

```
OSStatus SIISLP_GetLabelTypes(CFPropertyListRef *pLabelData);
```

Parameters

pLabelData A pointer to a CFPropertyListRef that will, upon return, contain data describing available Seiko Instruments USA Inc. label sizes. Use CFShow() to observe the contents of the pLabelData in Console.app. (Note: This data format may change in the final version of the SDK.)

Result

noErr upon successful completion of the routine.

SIIUtilities.mm

This section describes briefly the utilities provided in the SIIUtilities.mm file. Programmers may find these utilities useful when adding label printing functionality to their applications.

NSImageToBitMap

Purpose

To convert any arbitrary NSImage to a BitMap structure. NSImageToBitMap allocates a BitMap structure that must be freed with BitMapDispose.

Prototype

```
void NSImageToBitMap(NSImage *image, BitMap &outBitMap);
```

Parameters

image The address of a Cocoa NSImage object.

outBitMap Receives a freshly allocated 1-bit, black and white BitMap structure that may be used with SIISLP_Print or SIISLP_PrintAsync.

Result

true upon successful completion of the routine.

GWorldToBitMap

Purpose

To convert a GWorld to a BitMap structure. GWorldToBitMap allocates a BitMap structure that must be freed with BitMapDispose. This function uses CopyBits dithering to convert the input GWorld to a 1 bit BitMap.

Prototype

```
void GWorldToBitMap(GWorldPtr &inGW, BitMap &outBitMap, Rect &bounds);
```

Parameters

inGW	A pointer to a Carbon GWorld structure.
outBitMap	Receives a freshly allocated 1-bit, black and white BitMap structure that may be used with SIISLP_Print or SIISLP_PrintAsync.
bounds	The bounding rectangle of the GWorld.

Result

true upon successful completion of the routine.

BitMapCreate

Purpose

To initialize and allocate a BitMap structure from a 1-bit big-endian buffer, the number of bytes per row, and the rectangle of the image. BitMapCreate allocates a BitMap structure that must be freed with BitMapDispose. If inBaseAddress is NULL, the Pixel Map is allocated but not initialized allowing programmers to save a CopyBits operation after this function call.

Prototype

```
bool BitMapCreate(BitMap &outBitMap, char *inBaseAddress, SInt32 inRowBytes, Rect &inBounds);
```

Parameters

outBitMap	Receives a freshly allocated 1-bit, black and white BitMap structure that may be used with SIISLP_Print or SIISLP_PrintAsync.
inBaseAddress	The base address of an image buffer containing the first byte with a bit to be printed.
inRowBytes	The number of bytes per row or “line” of data. The framework to find lines of data uses this value.
inBounds	The address of a Carbon Rect structure containing the rectangle to be printed.

Result

true upon successful completion of the routine.

BitMapDispose

Purpose

Dispose the contents of a BitMap structure initialized by other functions found in SIIUtilities.cp.

Prototype

```
void BitMapDispose(BitMap &ioBitMap);
```

Parameters

ioBitMap A BitMap previously allocated by other functions in SIIUtilities.cp. This function may crash or cause unexpected errors if it is passed a BitMap structure that was not allocated by one of these routines.

BitMapRotate90

Purpose

Rotate a source BitMap 90 degrees to the right into a destination BitMap. BitMapRotate90 allocates a BitMap structure that must be freed with BitMapDispose. Rotation is lossless. Both the source and destination bitmaps need to be allocated by the functions within the SIIUtilities.cp file. The source and destination BitMaps can be the same structure.

Prototype

```
bool BitMapRotate90(BitMap &srcBitMap, BitMap &destBitMap);
```

Parameters

srcBitMap The BitMap to use as the source bitmap to rotate.

destBitMap The BitMap structure to receive the rotated copy of srcBitMap. Receives a freshly allocated 1-bit, black and white BitMap structure that may be used with SIISLP_Print or SIISLP_PrintAsync.

Result

true upon successful completion of the routine.

Index

- BitMap 8, 9, 15, 16, 17, 20, 21, 22
- BitMapCreate 8, 21
- BitMapDispose..... 8, 20, 21, 22
- BitMapRotate90 8, 9, 22
- Carbon 4, 6, 21
- Cocoa 4, 6, 20
- Compatibility 4
- GWorld 8, 21
- GWorldToBitMap 8, 21
- Installer 6, 7
- kSIISLPBusy..... 15
- kSIISLPerrCommunications..... 15
- kSIISLPerrHardwareOther 15
- kSIISLPerrNoLabels 15
- kSIISLPerrPaperJam 15
- kSIISLPerrPlatenOpen 15
- kSIISLPerrUnknown 15
- kSIISLPIdle 15
- NSImage 8, 9, 20
- NSImageToBitMap 8, 9, 20
- Quick Start 8
- SIISLP_CancelAllAsync 8, 16, 17
- SIISLP_ChangeCallbackPtr..... 10, 11
- SIISLP_FiindNext..... 11
- SIISLP_Finalize 9, 11
- SIISLP_GetDotsPerInch..... 18
- SIISLP_GetFirmwareVersion..... 8, 19
- SIISLP_GetIDFromReference.. 8, 12, 13
- SIISLP_GetLabelTypes..... 20
- SIISLP_GetMaxDotsPerLine 4, 8, 18
- SIISLP_GetReferenceFromID.. 8, 13, 14
- SIISLP_GetStatus 8, 14, 15
- SIISLP_GetTextDescription..... 8, 12, 13
- SIISLP_Initialize..... 9, 10, 11, 17
- SIISLP_Open 6, 7, 9, 14
- SIISLP_Print.. 4, 8, 9, 15, 16, 17, 20, 21, 22
- SIISLP_PrintAsync 8, 15, 16, 17, 20, 21, 22
- SIISLP_PrintFinishedCallbackPtr.. 4, 16, 17
- SIISLP_SetDensity..... 8, 18, 19
- SIISLP_SetGraphicsMode..... 4, 19
- SIISLP_TopOfForm..... 8, 17
- SIIUtilities..... 6, 8, 20, 22