Seiko Instruments Inc.

SMART LABEL PRINTER

SOFTWARE DEVELOPMENT KIT

7.1

DOCUMENTATION

SLP 620/650/650SE

©2008-2012 Seiko Instruments Inc.

Contributors

Written by Randy Turchik and Sanford Selznick.

Fine Print

This SII software is supplied to you by Seiko Instruments 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 Inc.
www.siibusinessproducts.com

Table of Contents

About this Document

This document describes the Seiko Instruments Inc. (SII) Smart Label Printer (SLP) Software Development Kit (SDK) for Windows XP or later. Described herein is a "C", "Visual Basic .Net" or "Visual Fox Pro" Application Programmer Interface (API) through which application developers can add SLP label printing functionality directly to their applications.

Revision History

- V7.1r0
  - Added samples for 203 DPI printers.
  - Updated installer.
  - Updated documentation.
- V7.0r0
  - Fixed issue with 64-bit barcode support.
  - Reworked installation documentation.
- v7.0
  - Added 64-bit support.
  - Added samples for C 64-bit, VB.Net 32/64-bit for .Net 2008.
  - Added QR barcode capability.
  - Improved samples for VB6.
- v6.2r6
  - Reworked samples with dedicated VB6 and VB.Net projects. The VB.Net sample targets .Net 2003 and can be upgraded to 2005, 2008 and 2010.
  - Added SlpComment to insert comments in to the debug log.
  - Added SlpGetErrorString to return a text description of the last error.
  - Added SlpGetPrinterDPI to return the DPI of the currently selected printer.
  - Added SlpGetBarCodeWidth() to learn the width of a specified barcode.
  - Various changes to debug logging, including:
    - Added debug logging for many Slp* functions that were not logged before.
    - The debug log header now contains version information, application information and OS information.
    - Added new detailed debug logging modes.
    - The debug log will be rolled when it grows larger than 100K bytes.
  - NULL may be passed to the first argument of SlpGetVersion to learn the length of the buffer necessary to hold the version information.
  - Fixed a problem with SlpGetPrinterName that would potentially return an incorrect or empty value.
  - Added new error codes for various barcode and drawing functions.
  - Several small bug fixes.
- v6.2r5
  - Added options to SlpDebugMode to output and log data to various targets.

    - Added SlpSetRotation for object rotation
    - Modified SlpGetVersion
    - Added SlpFindPrinters and SlpGetPrinterName to assist with enumerating available printers.
    - Added additional label type constants.
    - Added Visual Fox Pro example.
    - Added much more error checking around all function calls with appropriate logging via SlpDebugMode.
- v6.2r4
    - Updated installer to be Vista aware.
    - Added SLP command specification PDF to installer contents.
- v6.2r3
    - Added pcdlib32.dll.
- v6.2R2
    - Fixed silent install on some systems.
- v6.2
    - Added function SplGetTextHeight to get the text height of a font.
    - Added function SlpCopyLabelToClipboard. This is a handy way to test the look of labels without having to actually print. (It saves labels.)
    - SplDebugMode enables debug logging features.
    - Added a switch to flag VisualBasic 6 or VisualBasic.Net so the calls can be forced to use 32-bit integers.
    - Added SlpGetVersion to retrieve the specific version number of the DLL.
- v4 through v6.1
    - All printing from the SDK now prints through the 6.1 driver (available separately.) The SDK will print to SLP 410 through 450 printers that are available in Printers and Faxes.
    - SlpDrawBarcodeXY has been replaced with SlpDrawBarcode that now takes a rectangle and structure to describe the barcode. Many new barcode styles are now available (including 2D barcodes), described herein.
    - SlpDrawTextXY has been renamed to SlpDrawText.
    - SlpGetBarCodeWidth is no longer necessary and has been removed. SlpDrawBarcode automatically scales barcodes to the rectangle provided.
    - SlpDrawBitmap has been replaced with SlpDrawPicture which allows pictures to be imported directly from files. SlpDrawPicture uses the ImageMan graphics library (included with the SDK) to manipulate images.
    - SlpSetPrinter, SlpNewLabel and SlpPrintLabel have been replaced with two sets of functions:
        - SlpOpenPrinter and SlpClosePrinter
        - SlpStartLabel and SlpEndLabel
    - New “European Name Tag” and “Jewelry Label” sizes are now available.
    - SlpSetTextMode has been replaced with new handling of Unicode input described herein.
    - More error codes are returned from SlpGetErrorCode.

- The installer now has a “silent” mode that will install the SDK for use by end-users.

Compatibility

The SII SLP SDK framework is compatible with Windows XP (sp2) or later.

- Although the SDK may work well under earlier versions of Windows, if end-users are experiencing problems, they should be advised to update to the latest versions.
- The sample projects are intended for use with Visual Studio available with .Net 2008 and Visual Basic .Net.

Installation

Requirements

The SII SLP SDK requires that the Smart Label Printer Driver be installed. If the driver is not installed, there will be a warning during the installation process.

The Smart Label Printer Driver may be installed either with "plug-and-play" or with our combined application and driver installer that is available from <http://www.siibusinessproducts.com>.

Once the driver is installed, the SDK and examples will reference printers by their name in the system preferences printer list. (e.g.: "Printers and Faxes" for Windows XP.)

About 32- and 64-bit Operating Systems and Applications

Microsoft now offers 32-bit and 64-bit version of many of their operating systems. These operating systems are in turn compatible with 32-bit or 64-bit hardware.

Applications can be created as either 32-bit or 64-bit with modern development tools. A 32-bit application can run and launch just fine in a 64-bit environment, but a 64-bit application *cannot* launch in a 32-bit environment. A developer creating software can target their project for either 32- or 64-bit compatibility, and build on either a 32- or 64-bit OS if the proper compilers are installed. (To test a 64-bit application built on a 32-bit OS machine, remote debugging with a 64-bit OS will be required.) The SII SLP SDK provides capability to applications through Dynamic Linked Libraries (DLLs) referenced by name. The proper DLLs must be installed within proper system directories designed for either 32- or 64-bit support or in the same directory as the executable.

There is an extensive combinatorial repertoire of hardware, operating systems, development environments, execution environments and, ultimately, application compatibility, that developers should consider while developing with the SII SLP SDK.

Installing

The SII SLP SDK is fully compatible with 32- and 64-bit Windows Operating Systems. Developers starting development should install **all** components offered by the installer. These components include both 32- and 64-bit support, examples, additional installers, and this document.

Developers may install either the 32- or 64-bit support, depending on the settings in their project's preferences.

What's Installed

The SII SLP SDK is available as a Windows compatible installer, for no charge, from <http://www.siibusinessproducts.com/sdk/index.html>.

By default, the installer will create a folder to contain the SDK files. The folder path may vary slightly, depending on your version of Windows, but will generally be:

C:\Program Files (x86)\Seiko Instruments Inc\Smart Label Printer SDK

The following items will be installed in the SDK folder:

1. **SlpAPI.pdf** (This document.)
2. **SDK Core x32**: A directory containing the DLLs for 32-bit applications.
3. **SDK Core x64**: A directory containing the DLLs for 64-bit applications.
4. **SlpSdkRuntime32.msi**: An installer to install the 32-bit components (same as SDK Core x32) in to system directories.
5. **SlpSdkRuntime64.msi**: An installer to install the 64-bit components (same as SDK Core x64) in to system directories.
6. **SampleCode.zip**: A zip archive of sample applications in C, VB and C#. Right click to extract examples contained within to a new directory.

SampleCode.zip contains samples for VB.Net 32- and 64-bit, VB6 (32-bit only), Visual C, and Visual C Unicode. These examples will not run without installing the DLLs as described above. Deciding which to install depends not on the bit-depth of the operating system, but the bit-depth of the application developers want to build. This is determined by the project's preferences.

Instead of installing the DLLs in to system directories, developers may choose to copy the either of the SDK Core x* directories in to the same folder as the executable application you are creating. Ultimately this will be more compatible and easier to maintain.

Silent Installation

The application you are creating will eventually be installed on a users system and will require the DLLs described above to be installed.

You may distribute our DLLs with your own application installers. We provide both a 32- and a 64-bit versions of the DLLs for this purpose. To perform a silent installation of the DLLs, perform one of the following commands:

```
msiexec /I slpsdkruntime32.msi /QN
msiexec /I slpsdkruntime64.msi /QN
```

Distribution Checklist

The contents of the SDK Core x32 or x64 are the only items that need to be installed for applications to use the functions provided by the SII SLP SDK. Applications or Application Installers should install the DLLs before their application launches.

Furthermore, the SDK will not print properly if no printer instances for Smart Label Printers have been created in the User's Printers and Faxes Folder.

A restart is not required after installing the SDK. But a restart may be required after installing the driver.

SII SDK API Quick Start

The SlpApi7x32.dll/SlpApi7x64.dll uses modern techniques to communicate with Seiko Label Printers. Through a series of function calls, application programmers can do the following, and more:

1. Turn on debug logging (SlpDebugMode). Remove this for production.
2. Start a print job with multiple labels, or pages (SlpOpenPrinter, SlpStartLabel, SlpEndLabel, SlpClosePrinter).
3. Draw lines, rectangles and scaled pictures (from files) to labels (SlpDrawLine, SlpDrawRectangle, SlpDrawPicture).
4. Draw text with different font and style settings (SlpDrawText, SlpCreateFont, SlpDeleteFont).
5. Draw one-dimensional and two-dimensional barcodes (SlpSetBarCodeStyle, SlpDrawBarcode).

Printing One Label From C

A simple label printing session to print one label from C will look like this: (From HelloWorld examples.)

```
void main()
{
	SlpDebugMode(2);

	SlpOpenPrinter("Smart Label Printer 650", 1, FALSE);

	// layout and print one label
	{
		SlpStartLabel();

		HFONT font = SlpCreateFont("Courier", 12, 0);
		SlpDrawTextXY(30, 30, font, "Hello World!");

		SlpEndLabel(); // print the label!
	}

	SlpClosePrinter();
}
```

Printing One Label From Visual Basic 6

A simple label printing session to print one label from Visual Basic 6 will look like this: (From HelloWorld examples.)

```
Public Sub Command1_Click()

    Dim strPrinterName As String
    Dim hFont As Long
```

```
    SlpDebugMode (DEBUG_LOG)

    If SlpOpenPrinter("Smart Label Printer 650",
                      Size_Standard, False) <> 0 Then

        Call SlpStartLabel

        ' Layout and print one label
        hFont = SlpCreateFont("Courier", 12, 0)
        Call SlpDrawTextXY(30, 30, hFont, "Hello World!")

        Call SlpEndLabel ' Print the label!

        Call SlpClosePrinter

    End If

End Sub
```

Printing One Label From Visual Basic .Net

A simple label printing session to print one label from Visual Basic .Net will look like this: (From HelloWorld examples.)

```
Public Sub Command1_Click(ByVal eventSender As System.Object,
                          ByVal eventArgs As System.EventArgs)
                          Handles Command1.Click

    Dim strPrinterName As String
    Dim hFont As Integer

    SlpDebugMode((DEBUG_LOG))

    If SlpOpenPrinter("Smart Label Printer 650",
                      Size_Standard, False) <> 0 Then

        Call SlpStartLabel()

        hFont = SlpCreateFont("Courier", 12, 0)
        Call SlpDrawTextXY(30, 30, hFont, "Hello World!")

        Call SlpEndLabel() ' Print the label!

        Call SlpClosePrinter()

    End If
End Sub
```

Printing Multiple Labels

Print multiple labels like this:

```
void main()
{
      char s[128];

      SlpOpenPrinter("Smart Label Printer 650", 1, FALSE);

      // layout and print one label
      for (c = 0; c < 5; c++) {
            SlpStartLabel();

            sprintf(s, "Label %c", c);

            HFONT font = SlpCreateFont("Courier", 12, 0);
            SlpDrawTextXY(30, 30, font, s);

            SlpEndLabel(); // print the label!
      }

      SlpClosePrinter();
}
```

These functions, and more, are described in great detail in the section "SII SDK API Details" (Page 15).

More Samples

The Zip archives installed by the installer contain source files for both Visual Basic and C that print all types of labels from simple to complex.

SII SDK API Details

Contained herein is a detailed description of the various functions supported by the Smart Label Printer Printer (SLP) Application Programming Interface (API) library version 7.0 (SlpApi7x32.dll/SlpApi7x64.dll). For examples of how to use these functions, see the sample code for the appropriate development environment.

SlpOpenPrinter

Purpose
This function is used to specify the target printer. szPrinterName should be the name of the desired printer as defined in the Printer and Faxes folder. The nID parameter must be one of the following values:

nID	Description (See Label Catalog for Sizes)
1	Standard Address
2	Shipping
3	Diskette
4	Large Address
5	Multi-Purpose
7	VHS Spine
8	VHS Face
9	8mm Spine
10	File Folder
11	35mm Slide
12	Name Badge
13	Euro Folder (narrow)
14	Euro Folder (wide)
15	Iomega® Zip™ Disk
17	Jewelry Tag
18	Euro Name Tag
19	Hanging File Folder 1/3 Cut (SLP-3HFL)
20	Hanging File Folder 1/5 Cut (SLP-5HFL)
21	European Hanging File Folder (SLP-EHFL)
22	Fanfold Card Stock (SLP-FCS2)
23	European Name Tag Large (SLP-ENTL)
24	Retail Label (SLP-RTL)
25	Security Paper (SLP-DIA)
26	Topcoated Paper (SLP-P150)
27	Return Address (SLP-RTN)
28	Round (SLP-RND)
29	Multi-Part (SLP-MPL4)
30	Clear Return Address (SLP-RTNC)
31	Tamper Proof Label (SLP-TP)

32	Postage Label (SLP-STAMP1)
33	Large Postage Label (SLP-STAMP2)

Prototype
```
BOOL SlpOpenPrinter(LPSTR szPrinterName, int nID, BOOL fPortrait);
```

Parameters

szPrinterName	Specifies the target printer.
nID	Specifies the desired label type identifier (see below).
fPortrait	Set TRUE for portrait orientation, FALSE for landscape.

Return Value
If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpClosePrinter

Purpose
This function is used to close the printer and should be called before closing the application that opened the printer with SlpOpenPrinter().

Prototype
```
void SlpClosePrinter(void);
```

Parameters
None

Return Value

SlpStartLabel

None

Purpose
This function starts a new label by clearing the label buffer. It must be called before "drawing" a label.

Prototype
```
BOOL SlpStartLabel(void);
```

Parameters
None

Return Value
If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpEndLabel

Purpose
This function causes the current label to be printed. It should be called after all calls to the drawing functions (e.g.: DrawTextXY, DrawBarCodeXY, et al.) have been completed.

Prototype
```
BOOL SlpEndLabel();
```

Parameters
None

Return Value
If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpFindPrinters

Purpose
This function finds and builds and internal list of printers on the system. Details on each printer can then be found with SlpGetPrinterName, described below.

Prototype
```
int SlpFindPrinters(BOOL bAllPrinters);
```

Parameters

bAllPrinters	If set to TRUE, the list of printers is the same as all printers displayed in the Printers and Faxes control panel dialog. If set to FALSE, the list will contain only Smart Label Printers. To list only Smart Label Printers, the SDK will only look for printer names that contain either "Smart Label Printer" or "SLP" in their name. These are the default names after printers are installed. Determining if a printer is indeed a Smart Label Printer requires opening a connection to the printer. This is a very slow process on Windows, especially for Network printers.

Return Value
The number of printers found.

SlpGetPrinterName

Purpose
This function retrieves the name of one printer from the list of printers found by SlpFindPrinters. The SlpFindPrinters function should be called immediately before using this function. Otherwise the list of printers may be "stale" or empty.

Prototype
```
int SlpGetPrinterName(int nIndex, LPSTR szPrinterName, int nMaxLength);
```

Parameters

nIndex	A number from 0 to *n*-1, where *n* is the number of printers returned by the SlpFindPrinters function.
szPrinterName	A pointer to a string buffer to receive the name of the printer.
nMaxLength	The size of the buffer, in characters.

Return Value
Returns the number of characters copied into the szPrinterName buffer.

SlpGetPrinterDPI

Purpose
This function returns the DPI (dots per inch) resolution of the currently selected printer. If no printer is selected, zero is returned.

Prototype
```
int SlpGetPrinterDPI();
```

Parameters
None.

Return Value
The DPI of the currently selected printer. (e.g.: 300 for an SLP 450. 203 for an SLP 410.)

Notes
Developers should scale the coordinates of the elements of their printers within their own code. The SDK uses absolute label coordinates, not scalable vector coordinates. This allows developers to optimally scale their images and barcode objects.

SlpGetLabelHeight

Purpose
Retrieve the height of the printable area of the *current label*, in pixels.

Prototype
```
int SlpGetLabelHeight(void);
```

Parameters
None

Return Value
Returns the height of the printable area of the current label, in pixels.

SlpGetLabelWidth

Purpose
Retrieve the width of the printable area of the *current label*, in pixels.

Prototype
```
int SlpGetLabelWidth(void);
```

Parameters
None

Return Value
Returns the width of the printable area of the *current label*, in pixels.

SlpCreateFont

Purpose
The nAttribute parameter may be one of the values from the table below. C/C++ programs may use one or more of the attribute constants (NORMAL, BOLD, ITALIC, and UNDERLINE) defined in the SlpSdk7x.h header file.

This function creates a font with the specified typeface and attributes. The returned handle is used to specify the font when using the text drawing functions. When the font is no longer needed, the SlpDeleteFont function should be called to release memory used by the font.

nAttribute	Normal	Bold	Italic	Underline
0	X			
1		X		
2			X	
3		X	X	
4				X
5		X		X
6			X	X
7		X	X	X

Prototype
```
HFONT SlpCreateFont(LPSTR lpName, int nPoints, int nAttributes);
```

Parameters

lpName	Name of the typeface
nPoints	Height of font, in points

nAttributes Attributes for the font.

Return Value
If successful, the function returns a handle to the font, otherwise the return value is 0.

Visual Basic users: Because Visual Basic does not distinguish between signed and unsigned integers (used to store the HFONT), the return value from this function may be negative and is a valid value. When checking for an errant result from SlpCreateFont, be certain to use "myFont <> 0" instead of only "myFont > 0".

SlpDeleteFont

Purpose
To delete a font allocated with SlpCreateFont. This function deletes a font and frees memory allocated for it.

Prototype
```
int SlpDeleteFont(HFONT hFont);
```

Parameters
hFont Handle of font to be deleted.

Return Value
The function returns TRUE if successful or FALSE if hFont is invalid.

SlpGetTextWidth

Purpose
Returns the width (in pixels) of the specified text when drawn in the specified font.. This function computes the width of a character string using the specified font. The function will fail if the string contains any invalid characters.

Prototype
```
int SlpGetTextWidth(HFONT hFont, LPSTR lpText);
```

Parameters
hFont Specifies the font to be used for measuring the text.

lpText Points to the character string to be measured.

Return Value
Returns the width (in pixels) of the specified text when drawn in the specified font. The return value is 0 if lpText points to an empty string or if the function fails (the reason for failure may be obtained by calling the SlpGetErrorCode function).

SlpGetTextHeight

Purpose
Returns the height (in pixels) of the specified text when drawn in the specified font. This function computes the height of a character string using the specified font. The function will fail if the string contains any invalid characters.

Prototype
```
int SlpGetTextHeight(HFONT hFont, LPSTR lpText);
```

Parameters

hFont	Specifies the font to be used for measuring the text.
lpText	Points to the character string to be measured.

Return Value
Returns the height (in pixels) of the specified text when drawn in the specified font. The return value is 0 if lpText points to an empty string or if the function fails (the reason for failure may be obtained by calling the SlpGetErrorCode function).

SlpDrawTextXY

Purpose
This function draws a character string at the specified cursor position, using the specified font. The function will fail if the string contains any invalid characters or if the string will not fit at the specified cursor position.

The hFont parameter should be a handle obtained from a call to the SlpCreateFont function, or any valid Windows font handle owned by the current process. For example, the following code:

```
DrawTextXY(20, 10, hMyFont, "hello");
```

...would draw the word, "hello" at the position starting 20 pixels from the left, and 10 pixels from the top, of the printable area of the current label. The text is drawn using the font designated by hMyFont.

If x is negative, the actual x-coordinate will be automatically calculated to center the text horizontally on the label. For example:

```
DrawTextXY(-1, 10, hMyFont, "hello");
```

...would center the word, "hello" between the left and right margins.

Prototype
```
BOOL SlpDrawTextXY(int x, int y, HFONT hFont, LPSTR lpText);
```

Parameters

x	Specifies the physical x-coordinate of the starting location where the string will be drawn.
y	Specifies the physical y-coordinate of the starting location where the string will be drawn.
hFont	Specifies the font to be used for drawing the text..
lpText	Points to the character string to be drawn.

Return Value
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpDrawRectangle

Purpose
This function draws a rectangle at the specified position. The function will fail if the rectangle will not fit (either too wide or too high) at the specified position or if thickness is out of range.

For example, the following code:

```
Rectangle(20, 20, 100, 50, 4);
```

...will draw a rectangle 100 pixels wide and 50 pixels high, using lines which are 4 pixels thick.

Prototype

```
BOOL SlpDrawRectangle(int x, int y, int width, int height, int
thickness);
```

Parameters

x	Specifies the physical x-coordinate where the left side of the rectangle will be drawn.
y	Specifies the physical y-coordinate where the top of the rectangle will be drawn.
width	Specifies the width of the rectangle, in pixels.
height	Specifies the height of the rectangle, in pixels.
thickness	Specifies the thickness of the lines, in pixels. Must be in the range 1 to 8.

Returns
If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpDrawLine

Purpose
This function draws a line from the specified start point up to, but not including, the end point. The function will fail if the line will not fit (either too long or too thick) at the specified position or if thickness is out of range.

For example, the following code:

```
SlpDrawLine(20, 20, 100, 100, 4);
```

...will draw a diagonal line 80 pixels long and 4 pixels thick.

Prototype
BOOL SlpDrawLine(int xStart, int yStart, int xEnd, int yEnd, int thickness);

Parameters

xStart	Specifies the physical x-coordinate of the starting point of the line.
yStart	Specifies the physical y-coordinate of the starting point of the line.
xEnd	Specifies the physical x-coordinate of the ending point of the line.
yEnd	Specifies the physical y-coordinate of the ending point of the line.
thickness	Specifies the thickness of the line, in pixels. Must be in the range 1 to 8.

Return Value
If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpDrawPicture

This function draws the picture at the specified position on the current label. The left and top members of rect always define the position of the upper-left corner of the picture. The picture is normally stretched, as necessary, to make it fit into the specified rectangle, except as noted below.

If bottom is set to zero, the bottom position of the picture is automatically calculated so that the height of the picture is scaled proportionately to the width.

If right is set to zero, the right position of the picture is automatically calculated so that the width of the picture is scaled proportionately to the height.

If both bottom and right are zero, the picture is drawn actual size (pixel-to-pixel). If the picture is too wide and/or too tall to fit on the label, the extra part of the image is clipped.

Prototype

```
BOOL SlpDrawPicture( int nLeft, int nTop,
int nRight, int nBottom, LPSTR szPath);
```

Parameters

nLeft	Left edge of the picture.
nTop	Top edge of the picture.
nRight	Right edge of the picture.
nBottom	Bottom edge of the picture.
szPath	Specifies the image file path. The file must be one of the following types: JPEG image, PNG image, or Windows Bitmap (BMP).

Return Value

If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpSetBarCodeStyle

Purpose

To define the characteristics of a barcode drawn with SlpDrawBarcode. The SlpSetBarcodeStyle function defines the attributes of a bar code defined as follows:

Prototype

```
BOOL SlpSetBarCodeStyle(int nSymbology, int nRatio, int nMode,
  int nSecurity, BOOL bReadableText, int nFontHeight,
  int nFontAttributes, LPSTR sbFaceName);
```

Parameters

nSymbology — Specifies the symbology to be used for drawing bar codes. Must be set to one of the values in the following table:

Code	Symbology	Bar Ratio
1	Code 39	Variable
2	Code 2 of 5	Variable
3	Codabar	Variable
4	Code 128	Variable
5	UPC-A	Fixed
6	UPC-E	Fixed
7	EAN-13	Fixed
8	POSTNET	Fixed

9	RM4SCC	Fixed
20	Maxicode	Fixed
21	PDF417	Fixed
22	DataMatrix	Fixed
23	QR Code	Fixed

nRatio — Specifies the ratio of wide to narrow bars, for symbologies that allow a variable bar ratio. This value is ignored when defining a bar code with a fixed bar width ratio and may be set to zero.

The actual ratio used will be one-tenth of the value specified. For example, a value of 25 specifies a ratio of 2.5 : 1.

The value must be in the range of 20 – 30 (inclusive). If this value is set to zero, the default ratio (2.5 : 1) will be used.

nMode — This value specifies the mode for 2D bar codes. The value is dependant on the bar code symbology and must be set as follows:

Symbology	Mode
Maxicode	4 = Any data up to 84 data characters automatically split between primary and secondary messages. 5 = Any data up to 68 data characters automatically split between primary and secondary messages. Enhanced error correction used
PDF417	0 = EXC Alpha 1 = EXC Lower 2 = EXC Mixed 3 = EXC Punctuation 4 = Binary/ASCII Plus
DataMatrix	0 = Square 1 = Rectangular
QR Code	0 = Automatic mode (Always set to zero)

nSecurity — This value is valid only for PDF417 and DataMatrix symbologies (it is ignored by others), and must be set as follows:

Symbology	Security level
PDF417	One of PDF417's most valuable features is its ability to allow correction of errors. The number of damaged codewords (Nmax) which may be recovered depends on the security level, as follows: Level — Nmax 0 — 0 1 — 2 2 — 6 3 — 14

	4 30 5 62 6 126 7 154 8 510
DataMatrix	DataMatrix symbols can include a user-selected amount of error-correction data. For Square symbols the Security Level setting may be in the range 0 – 24, while for Rectangular symbols the allowed range is 0 – 6.

nReadableText	Set to TRUE to display human-readable text below the barcode, otherwise FALSE. This value is ignored for postal bar codes (e.g., POSTNET, RM4SCC) and 2D bar codes (Maxicode, PDF417, DataMatrix). When set to TRUE, font must point to a valid LOGFONT structure.
nFontHeight	Defines the height of the font (in points) to use for human-readable text. This value must be valid when sbReadableText is set to TRUE – otherwise, it may be set to 0.
nFontAttributes	Defines the attributes (e.g., bold, italic, etc.) of the font to use for human-readable text. This value must be valid when sbReadableText is set to TRUE – otherwise, it may be set to 0. See the Remarks section of SlpCreateFont for more information about attributes.
sbFaceName	Defines the typeface name (e.g., Arial) of the font to use for human-readable text. This value is ignored if sbReadableText is set to FALSE.

Return Value
If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpDrawBarCode

Purpose
Draw a bar code symbol that represents lpText within the specified rectangle using the specified bar code symbology (See SlpBarCodeStyle). The function will fail if the string contains any invalid characters or if the bar code symbol will not fit within the specified rectangle.

Prototype
```
BOOL SlpDrawBarCode (int nLeft, int nTop,
  int nRight, int nBottom, LPSTR lpText);
```

Parameters

nLeft	Left edge of the barcode.
nTop	Top edge of the barcode.
nRight	Right edge of the barcode.
nBottom	Bottom edge of the barcode.
lpText	Points to a character string that represents the bar code to be drawn.

Return Value
If the function succeeds, the return value is TRUE.
If the function fails, the return value is FALSE and the reason for failure may be obtained by calling the SlpGetErrorCode function.

SlpGetBarCodeWidth

Purpose
Retrieve the width in pixels necessary to draw a specified bar code. This function is identical to SlpDrawBarCode, except (1) the barcode is not drawn and (2) the width of the rendered barcode is returned. SlpBarCodeStyle must be called prior to calling this function or an error will occur. *The returned width, combined with the input height, may be used to compute a rectangle to center the barcode on a label.*

Prototype
```
int SlpDrawBarCode (int nLeft, int nTop,
  int nRight, int nBottom, LPSTR lpText);
```

Parameters

nLeft	Left edge of the barcode.
nTop	Top edge of the barcode.
nRight	Right edge of the barcode.
nBottom	Bottom edge of the barcode.
lpText	Points to a character string that represents the bar code to be drawn.

Return Value
If the function succeeds, the return value is the number of horizontal pixels that will be used to render the specified barcode. This value may be less than nRight minus nLeft.
If the function fails, the return value is a negative error code as documented in SlpGetErrorCode.

SlpSetRotation

Purpose
Sets the angle for subsequestly drawn objects. This affects objects drawn with the SlpDrawTextXY(), SlpDrawBarCode(), and SlpDrawPicture() functions. the rotation angle will remain in effect until changed. Therefore, be sure to set the rotation back to 0 after drawing rotated objects.

Prototype
```
int SlpSetRotation(int nAngle);
```

Parameters
nAngle Angle index for rotation. One of the following values:

nAngle	Rotation
0	0 degrees
1	90 degrees
2	180 degrees
3	270 degrees

Return Value
Returns the nAngle value the rotation was set to. If nAngle is out of range, rotation will be set to an nAngle of 0, and 0 will be returned.

SlpGetErrorCode

Purpose
Retrieve the global error code from SDK functions that set it. See SlpGetError string for a textual description of an error.

Prototype
```
int SlpGetErrorCode(void);
```

Parameters
None

Return Value
Returns an error code for the last API call. If no error occurred, the return value is zero. If the return value is zero, but the API function failed, an unexpected error occurred. Normally, the SlpGetErrorCode function does not need to be called unless an API function fails. See the table below for a list of possible error codes.

Value	Reason for failure
0	No failure.
-1	No device context has been created (probably because the SlpSetLabelType function was not called, or it failed).
-2	An unexpected error occurred in the API library.
-3	An invalid label type was passed to the SlpSetLabelType function.
-4	An invalid font handle was passed to an API function.
-5	An invalid thickness parameter was passed to SlpDrawRectangle. The

	thickness must be less than 9, but not greater than either the height or width of the rectangle.
-6	An invalid value was passed to SlpDrawBarCode. See the Remarks section under SlpDrawBarCode for correct values.
-7	An invalid printer type was passed to SlpSetPrinter. See the table under SlpSetPrinter for a list of supported printer types.
-8	An invalid port was passed to SlpSetPrinter. See the table under SlpSetPrinter for a list of supported ports.
-9	The current label type is not valid for the selected operation. Usually, this results from specifying a label type that is not supported by the selected printer (e.g., the SLP 410 model cannot use "wide" labels).
-10	An error occurred while processing the selected bar code symbology.
-11	An invalid bitmap handle was passed to SlpDrawBitmap, or an invalid file path was passed to SlpDrawPicture.
-12	The bar code library could not be opened.
-13	Internal error. Please contact Seiko Instruments with a debug log!
-14	The selected printer could not be opened.
-15	The image file is corrupted or of an unknown format.
-16	An error occurred while processing the image.
-17	The size of the passed buffer is not adequate for the requested content.
-18	Unexpected error. Please contact Seiko Instruments with a debug log!

SlpGetErrorString

Purpose
Retrieve a text string describing the last error that occurred in the SDK. If the last function call to the SDK was successful, an empty string ("") is returned.

Prototype
```
int SlpGetErrorString(LPSTR szMessage, int nMaxLength);
```

Parameters

szMessage	A character buffer to receive the error string. If szMessage is NULL, a length will be returned only.
nMaxLength	The number of characters, including null terminator, that may be copied to szMessage.

Return Value
The number of characters in the error string, including the null terminator.

SlpGetVersion

Purpose
Obtain a version string describing the DLL.

Prototype
```
int SlpGetVersion(LPSTR szVersion);
```

Parameters

szVersion	A character buffer to receive the version string, including null terminator. If szVersion is NULL, a length will be returned only.

Return Value
The number of characters in the version string, including the null terminator.

SlpCopyLabelToClipboard

Purpose
Copies the current in-memory image of the label to the clipboard for debugging purposes. This can be used at any time (e.g., after each element is added to the label).

Prototype
```
void SlpCopyLabelToClipboard(void);
```

Parameters
None

Return Value
A BMP in the system clipboard of the current label.

SlpDebugMode

Purpose
Enables a special debugging mode. In this mode, all API functions calls (and parameters) may be logged to the system debug monitor (using the Windows OutputDebugString() function). Also, when a label is printed, an image of the final label may copied to the clipboard, instead of actually printing a physical label. This should save a lot of labels during development and testing.

The tool 'DebugView', available from microsoft.com, can display the log messages. The program is sometimes called 'dbgview'.

The debug log file size is now checked after program startup. If the file size is greater than 100K bytes, the following operations will be performed:

1. If a file named SlpSdk.log.bak exists, it will be deleted.
2. The current SlpSdk.log file will be renamed to SlpSdk.log.bak.
3. A new SlpSdk.log file will be created.
4. The SlpSdk.log file will continue to grow until the program terminates.

Prototype
```
void SlpDebugMode(DWORD dwMode);
```

Parameters

dwMode	Numeric specifying the debug mode. These values may be added together to turn modes on or off.

DEBUG_OFF (0)	specifies no debug output.
DEBUG_ON (1)	For code designed using SlpDebugMode prior to SDK 6.2r5, this value defaults to (DEBUG_LOG + DEBUG_VIRTUAL_PRINT + DEBUG_ENTER_EXIT.
DEBUG_LOG (2)	For writing debug messages to the debug log file "SlpSdk.log". The debug log file will be located in the current user's Temp folder.
DEBUG_VIRTUAL_PRINT (4)	Output label image to clipboard, instead of printer.
DEBUG_OUT_DBG_MSG_BOX (8)	Display debug messages in a Windows message box with an "OK" button
DEBUG_OUT_DBG_STR(16)	Write debug messages to the system debugger.
DEBUG_ENTER_EXIT (32)	Log entry *and* exit from API functions.

Return Value
None.

SlpComment

Purpose
Allows user generated messages to be insert in to the debug log if debug mode is enabled.

Prototype
```
void SlpComment(LPSTR szMessage);
```

Return Value
None.

Appendix 1

About Unicode

The SLP Application Programming Interface (API) is compatible with both ANSI and Unicode text. The API functions described in this document assume that text strings are ANSI. However, there are Unicode counterparts provided for all functions that take text parameters. These functions have a "W" suffix appended to the function name (e.g., SlpPrintTextW is the Unicode equivalent of SlpPrintText) and are functionally equivalent to the ANSI versions. For example:

ANSI Version	Unicode Version
SlpOpenPrinter	SlpOpenPrinterW
SlpCreateFont	SlpCreateFontW
SlpDrawTextXY	SlpDrawTextXYW
SlpGetTextWidth	SlpGetTextWidthW
SlpSetBarCodeStyle	SlpSetBarCodeStyleW
SlpDrawBarCodeXY	SlpDrawBarCodeXYW
SlpGetBarCodeWidth	SlpGetBarCodeWidthW
SlpDrawPicture	SlpDrawPictureW

The "C" header file, SlpSdk7x.h, includes preprocessor directives that map the Unicode names to the corresponding ANSI prototypes when the symbol UNICODE is defined. Therefore, when compiling Unicode C/C++ source code for Windows XP or later, the correct functions are automatically used – no changes to the source code are required when SlpSdk7x.h is included.

Appendix 2

Graphics Mode

If your label contains fine-grained graphics or barcodes, you should consider placing the printer in to fine-mode. This can be done within Printers and Faxes, under the Advanced settings for the printer. Labels with fine-graphics or barcodes print much more accuracy.

Appendix 3

C programs and DLLs

All programs, be they written with C, C++, Visual Basic, Visual Fox Pro or any other language will interface with the SDK through its Dynamically Linked Library (DLL). All of the functions described in this manual are contained within the DLL.

The DLL was created with _stdcall interfaces generated by Microsoft Developer Studio. Although the DLL provides a "standard" interface, not all environments agree on what a standard interface is. For example, Visual Basic programs can call to functions in the DLL without any name modifications at all. If your particular environment, for whatever reason, cannot find and call the functions by the names described throughout this manual, try the Entry Points below as names instead.

Function Name	Entry Point
SlpClosePrinter	_SlpClosePrinter@0
SlpCreateFont	_SlpCreateFont@12
SlpDeleteFont	_SlpDeleteFont@
SlpDrawBarCode	_SlpDrawBarCode@20
SlpDrawLine	_SlpDrawLine@20
SlpDrawPicture	_SlpDrawPicture@20
SlpDrawRectangle	_SlpDrawRectangle@20
SlpDrawTextXY	_SlpDrawTextXY@16
SlpEndLabel	_SlpEndLabel@0
SlpFindPrinters	_SlpFindPrinters@4
SlpGetErrorCode	_SlpGetErrorCode@0
SlpGetLabelHeight	_SlpGetLabelHeight@0
SlpGetLabelWidth	_SlpGetLabelWidth@0
SlpGetPrinterName	_SlpGetPrinterName@0
SlpGetTextWidth	_SlpGetTextWidth@8
SlpGetVersion	_SlpGetVersion@4
SlpOpenPrinter	_SlpOpenPrinter@12
SlpSetBarCodeStyle	_SlpSetBarCodeStyle@32
SlpSetRotation	_SlpSetRotation@4
SlpStartLabel	_SlpStartLabel@0

Appendix 4

Changing the Printer's Serial Baud Rate

Please see the document named "SLPAdminNotes.pdf" that is installed with the full version of the Smart Label Printer end-user software. Contained within this document are instructions to use our software to change your printer's serial baud rate.

Index