



## Topaz Systems, Inc. SigPlusDemoLCD1x5 Guide

Topaz Systems, Inc.  
650 Cochran Street, Unit 6  
Simi Valley, CA, 93065  
©1995-2006, all rights reserved, US patent 6,307,955, and pending

www.topazsystems.com  
tech support: 805 520-8286  
support@topazsystems.com

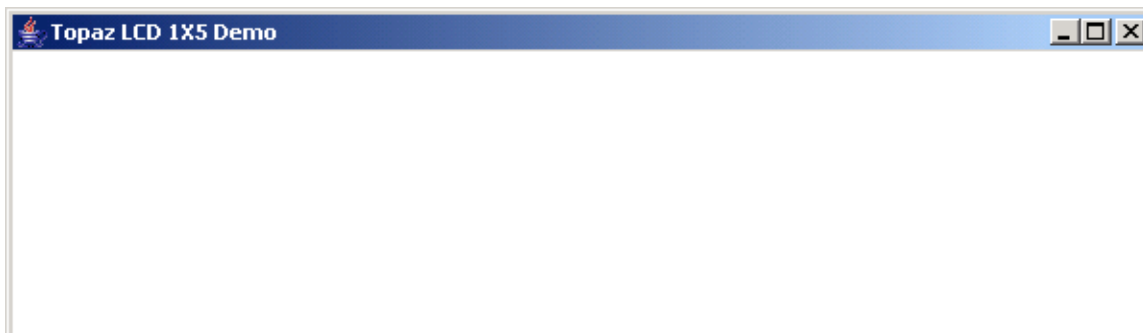
Welcome to the SigPlusDemoLCD1x5 Guide. This will walk you through our demo “SigPlusDemoLCD1x5.java”, available from our website ([www.topazsystems.com](http://www.topazsystems.com)).

To download go to:

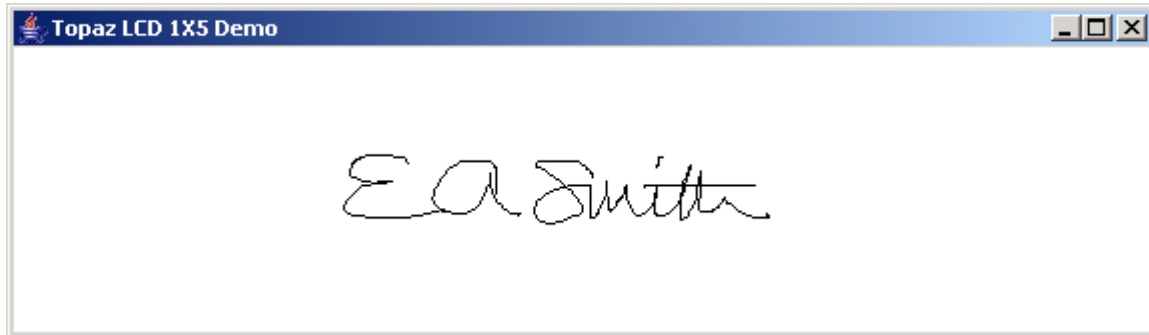
[http://www.topazsystems.com/software/download/java/sigplusjava2\\_45.zip](http://www.topazsystems.com/software/download/java/sigplusjava2_45.zip)

This demo shows the developer how to create an application that communicates information to an LCD, and how to add and manage hotspots on the LCD for user pen tapping. This demo is for use with an LCD 1x5 tablet. Begin by opening SigPlusDemoLCD1x5.java in a Java runtime environment.

The window below will appear on your monitor. Below you can also see what the interface on the pad during this stage in the demo.



Sign on the dotted line on your tablet and you will see the user signature on the screen (as shown below).



Press "Ok" after you are done signing. Your tablet will now display the options "Back" and "Exit." Click "Exit" to exit the application, or "Back" to go back to the previous screen.



Because there are no user connection or tablet type options for the user to access when creating this kind of application, it is very important to set all tablet properties correctly in your code..

## The Code

We will now look over the Java code used to control this application. The code below creates a form for the SigPlusDemoLCD1x5.

```
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import javax.comm.*;
import java.io.*;
import com.topaz.sigplus.*;

public class SigPlusDemoLCD1x5 extends Frame implements Runnable
{
    SigPlus      sigObj = null;
    static final int  numImages = 4;
    Image[]      rawImages;

    static final String  fileBase = "C:\\SigPlusJava2_45\\lcd1x5demo\\";
    Thread         eventThread;
    int            page;
    boolean        start;
    String         signature;

    public static void main(String Args[])
    {
        SigPlusDemoLCD1x5 demo = new SigPlusDemoLCD1x5();
    }

    public SigPlusDemoLCD1x5()
    {
        int  i;
        String  fileName;

        try
        {
            String drivename = "com.sun.comm.Win32Driver";
            try
            {
                CommDriver driver = (CommDriver)
Class.forName(drivename).newInstance();
                driver.initialize();
            }
            catch (Throwable th)
            {
                /* Discard it */
            }
        }
    }
}
```

The code below initializes the com API. This is necessary even with HSB pads because Java expects to see the com initialized. The com driver class is only necessary while in Windows. It then uses the ClassLoader class to create a new instance of SigPlus.

```
ClassLoader cl = (com.topaz.sigplus.SigPlus.class).getClassLoader();
sigObj = (SigPlus)Beans.instantiate(cl, "com.topaz.sigplus.SigPlus");

setLayout( new GridLayout(1, 1));
add(sigObj);
pack();
```

```

        setTitle("Topaz LCD 1X5 Demo");

addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent we)
    {
        System.exit(0);
    }

    public void windowClosed(WindowEvent we)
    {
        System.exit(0);
    }
});

```

There are three SigPlus event handlers in SigPlus that are currently unimplemented. These are not available for the developer to use.

```

sigObj.addSigPlusListener(new SigPlusListener()
{
    public void handleTabletTimerEvent(SigPlusEvent0 evt)
    {
    }
    public void handleNewTabletData(SigPlusEvent0 evt)
    {
    }
    public void handleKeyPadData(SigPlusEvent0 evt)
    {
    }
});

```

The code below sets the tablet model, Comport, and adds a time and date stamp. SigPlus and the LCD are also set so that inking wont happen. Then several hotspots are added.

```

setSize(600, 175);
show();

sigObj.setTabletModel("SignatureGemLCD1X5");
sigObj.setTabletComPort("HID1");
sigObj.setLDCaptureMode(2);
sigObj.setTabletState(1);

sigObj.setDisplayRotation(0);
sigObj.setDisplayJustifyMode(1);
sigObj.setDisplayJustifyX(50);
sigObj.setDisplayJustifyY(50);
//sigObj.setTimeStamp("August 16, 2002 14:12:00"); //Sets time stamp string
//sigObj.setDisplayTimeStamp(true); //Displays time stamp
//sigObj.setAnnotation("Topaz Java"); //Sets annotation string
//sigObj.setDisplayAnnotation(true); //Displays annotation

sigObj.lcdSetWindow(0, 22, 240, 40);
sigObj.keyPadSetSigWindow(1, 0, 22, 240, 40);

sigObj.keyPadAddHotSpot(0, 1, 195, 6, 19, 19);
sigObj.keyPadAddHotSpot(1, 1, 10, 4, 40, 17);
sigObj.keyPadAddHotSpot(2, 1, 28, 36, 32, 12);
sigObj.keyPadAddHotSpot(3, 1, 168, 37, 30, 12);

```

Below the list of image files to put on the LCD is created.

```
MediaTracker mt = new MediaTracker(this);
rawImages = new Image[ numImages ];
String[] imageTitles =
    {
        "ThankYou",
        "Sign",
        "Ok",
        "Clear"
    };

for(i = 0; i < numImages; i++)
{
    fileName = fileBase + imageTitles[i] + ".jpg";
    rawImages[ i ] = Toolkit.getDefaultToolkit().getImage(fileName);
    mt.addImage(rawImages[ i ], i + 1);
}
try
{
    mt.waitForAll();
}
catch(Exception e)
{
    System.out.println("Error opening bitmap files");
}
```

First we clear the LCD and write out 1 image to the background memory of the pad (rawImage[6]), and then write 3 images to the foreground of the LCD. Images written to the foreground will immediately show up on the LCD. Images written to the background will show up only when LCD refresh is called with an initial argument of 2.

```
sigObj.lcdRefresh(0, 0, 0, 240, 128);

sigObj.lcdWriteImage (1, 2, 0, 0, 240, 53, rawImages[0]);
sigObj.lcdWriteImage (0, 2, 0, 20, 240, 44, rawImages[1]);
sigObj.lcdWriteImage (0, 2, 207, 4, 21, 11, rawImages[2]);
sigObj.lcdWriteImage (0, 2, 15, 4, 37, 11, rawImages[3]);
    sigObj.lcdSetWindow(0, 22, 240, 40);
    sigObj.keyPadSetSigWindow(1, 0, 22, 240, 40);
page=1;

eventThread = new Thread(this);
eventThread.start();
}
catch (Exception e)
{
    return;
}
}
public void run()
{
    try
    {
        while (true)
        {
            Thread.sleep(300);
```

Here is the Pulling Loop where hotspots are managed. HotSpot(0) opens the next page.

HotSpot(1) clears the signature data. HotSpot(2) is on the next page and goes back to the previous page, while HotSpot(3) exits and resets the pad to default settings. Shown below.

```
if(sigObj.keyPadQueryHotSpot(0) != 0 && page==1)
{
    sigObj.keyPadClearSigWindow(1);
    sigObj.lcdRefresh(1, 210, 3, 14, 14);
    signature=sigObj.getSigString(); //pass signature to "signature" variable
    sigObj.clearTablet();
    sigObj.keyPadClearSigWindow(1);
    page=2;
    sigObj.lcdSetWindow(0, 0, 0, 0);
    sigObj.keyPadSetSigWindow(1, 0, 0, 0, 0);
    sigObj.lcdRefresh(2, 0, 0, 240, 64);
}
else if(sigObj.keyPadQueryHotSpot(1) != 0 && page==1)
{
    sigObj.keyPadClearSigWindow(1);
    sigObj.clearTablet();
    sigObj.lcdRefresh(1, 10, 0, 45, 17);
    sigObj.lcdRefresh(0, 0, 18, 240, 50);
    sigObj.lcdWriteImage(0, 2, 0, 20, 240, 44, rawImages[1]);
    sigObj.lcdRefresh(1, 10, 0, 45, 17);
    sigObj.keyPadClearSigWindow(1);
}
else if(sigObj.keyPadQueryHotSpot(2) != 0 && page==2)
{
    sigObj.keyPadClearSigWindow(1);
    sigObj.clearTablet();
    sigObj.lcdRefresh(1, 29, 38, 44, 19);
    sigObj.keyPadClearSigWindow(1);
    sigObj.lcdRefresh(0, 0, 0, 240, 64);
    page = 1;
    sigObj.keyPadSetSigWindow(1, 0, 22, 240, 40);
    sigObj.lcdSetWindow(0, 22, 240, 40);
    sigObj.lcdWriteImage (0, 2, 0, 20, 240, 44, rawImages[1]);
    sigObj.lcdWriteImage (0, 2, 207, 4, 21, 11, rawImages[2]);
    sigObj.lcdWriteImage (0, 2, 15, 4, 37, 11, rawImages[3]);
    sigObj.clearTablet();
}
else if(sigObj.keyPadQueryHotSpot(3) != 0 && page==2)
{
    sigObj.keyPadClearSigWindow(1);
    sigObj.clearTablet();
    sigObj.lcdRefresh(1, 182, 37, 30, 19);
    sigObj.keyPadClearSigWindow(1);
    sigObj.setTabletState(1);
    sigObj.setLDCaptureMode(1);
    sigObj.lcdRefresh(0, 0, 0, 240, 64);
    sigObj.setTabletState(0);
    System.exit(0);
}

sigObj.keyPadClearSigWindow(1);
}
}
catch (InterruptedException e)
{
}
}
}
```

## **Dependencies and Files Necessary to Run This Demo**

*Jar files must be included in the CLASSPATH*

1. SigPlus2\_xx.jar (at the creation of this document the current version was 2\_45)
2. Comm.jar (This file will vary depending on OS, you will need the correct java com API for your OS.)

### *Additional Files*

1. (Win)- Win32com.dll must reside in the System32 folder
2. (HSB in Win) – SigUsb.dll must reside in the System32 folder

For more information regarding products and support, please visit the Topaz web site at <http://www.topazsystems.com>