How-To Guide

SigPlus Demo LCD 4x3 Java

Table of Contents

Overview

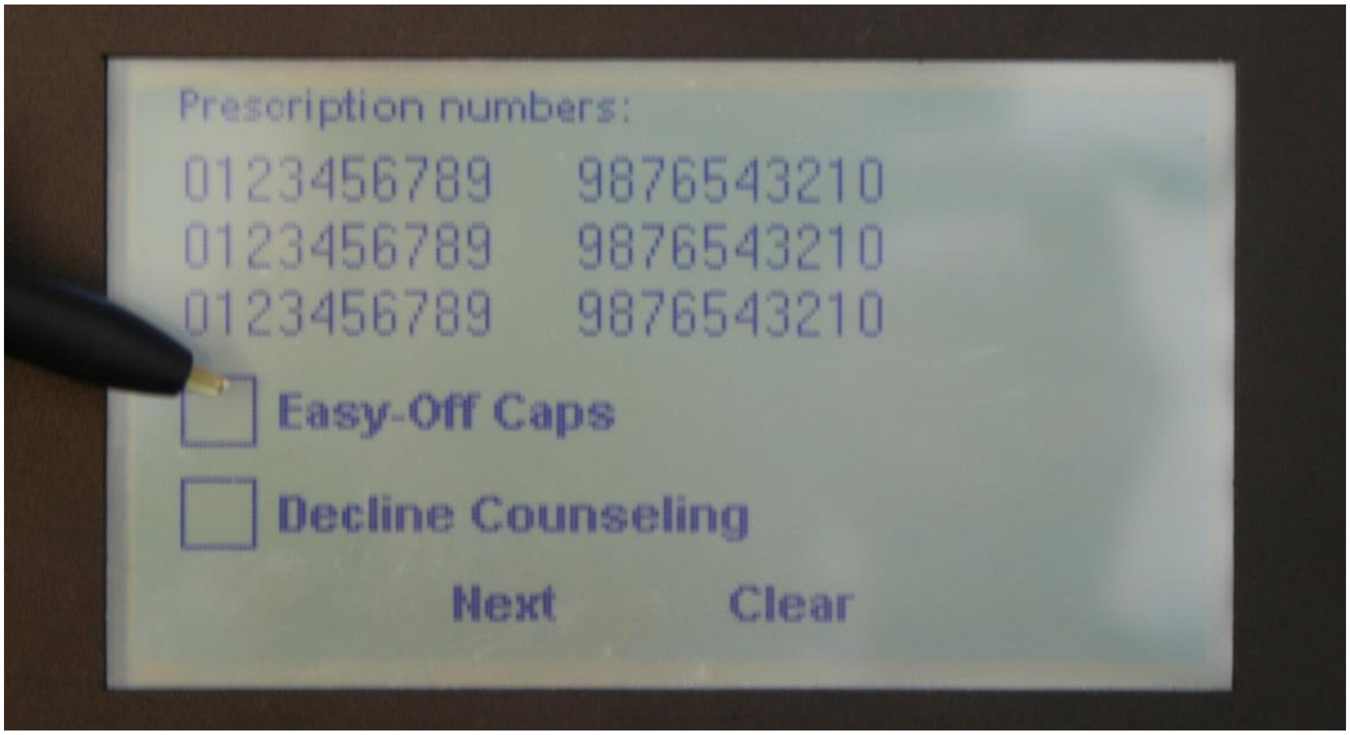
Welcome to the SigPlus Demo LCD 4x3 Guide. This will walk you through our demo “SigPlusDemoLCD4x3.java”, available from the Topaz website.

Download at:
www.topazsystems.com/sigplusprojava.html

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.

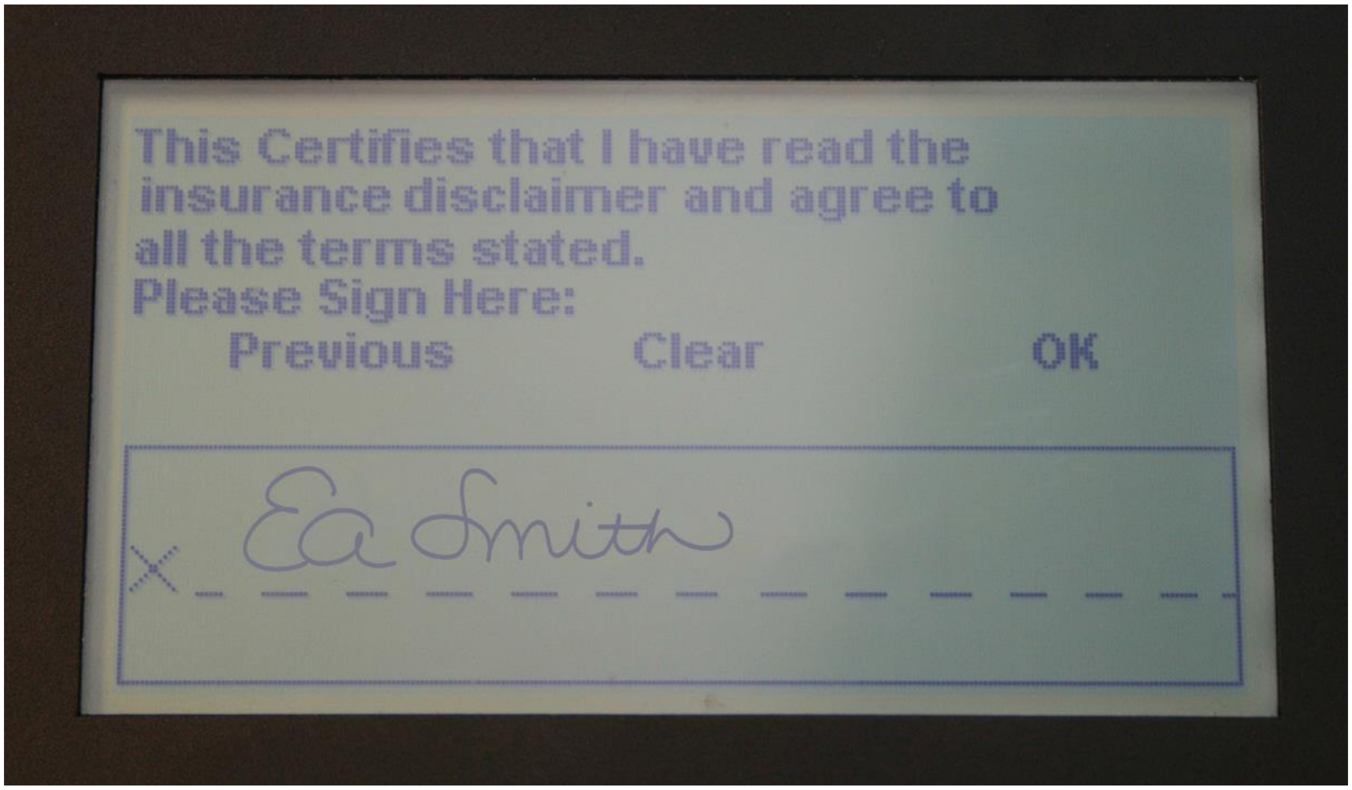
Below you can see both what the interface looks like on the computer screen and what is shown on the pad during the demo. This demo is for use with an LCD 4x3 tablet.

The “Topaz LCD 4x3 Demo” window will appear on your monitor. Below you can also see what the interface on the pad during this stage in the demo looks like


Prescription numbers:
0123456789 9876543210
0123456789 9876543210
0123456789 9876543210
Easy-Off Caps
Decline Counseling
Next Clear

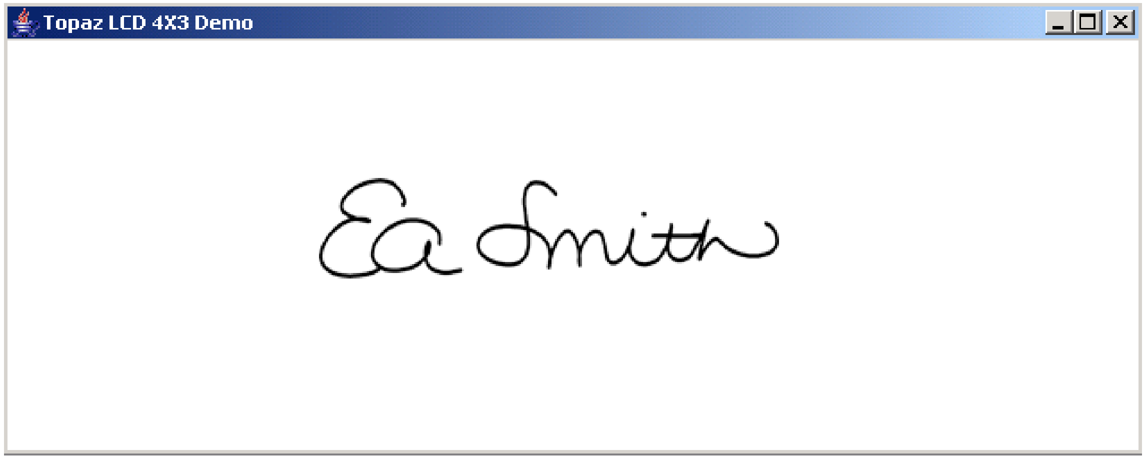
From here, you can choose to turn “Easy-Off Caps” on, or you can “Decline Counseling.” Make your selections, and tap “Next”. If you make a mistake, you can press “Clear” to clear whatever check marks you have made.

The next screen prompts the user to sign. Sign on the dotted line.



This Certifies that I have read the insurance disclaimer and agree to all the terms stated.
Please Sign Here:

The screen below will show in the "Topaz LCD 4x3 Demo" window on your monitor.



Press "OK" after you are done signing. Your tablet will now display a thank you message, and the application is closed.

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 “SigPlusDemoLCD4x3”.

```
import java.awt.*;
import java.awt.event.*;
import java.beans.*;
import javax.comm.*;
import java.io.*;
import com.topaz.sigplus.*;
public class SigPlusDemoLCD extends Frame implements Runnable
        {
        SigPlus             sigObj = null;
        static final int    numImages = 8;
        Image[]             rawImages;

        static final String fileBase = "C:\\SigPlusJava2_45\\lcd4x3demo\\";

        int AppScreen;
        Thread eventThread;
        int currentImage;
        int page;
        String signature;

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

        public SigPlusDemoLCD()
                {
                int i;
                String fileName;

                try
                {

                        String drivername = "com.sun.comm.Win32Driver";
                                try
                                {
                                CommDriver driver = (CommDriver)
Class.forName(drivername).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 4x3 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, COM port, and adds a time and date stamp. SigPlus and the LCD are also set so that inking won’t happen. Then several hotspots are added.

```
setSize(640, 256);
show();

sigObj.setTabletModel("SignatureGemLCD4x3New");
        sigObj.setTabletComPort("HID1");
sigObj.setLCDCaptureMode(2);
        sigObj.setTabletState(1);
sigObj.setDisplayRotation(0);
sigObj.setDisplayJustifyMode(1);
sigObj.setDisplayJustifyX(50);
sigObj.setDisplayJustifyY(50);
//sigObj.setTimeStamp("August 16, 2002 13:40:00"); //Timestamp string
//sigObj.setDisplayTimeStamp(true); //Sets up to display the timestamp
//sigObj.setAnnotation("Topaz Java"); //Annotation string
//sigObj.setDisplayAnnotation(true); //Sets up to display annotation

sigObj.keyPadSetSigWindow(1, 0, 0, 0, 0);
sigObj.lcdSetWindow(0,0,0,0);

sigObj.keyPadAddHotSpot(0, 1, 11, 55, 15, 25);
        sigObj.keyPadAddHotSpot(1, 1, 11, 75, 15, 25);
        sigObj.keyPadAddHotSpot(2, 1, 70, 95, 30, 30);
        sigObj.keyPadAddHotSpot(3, 1, 132, 95, 40, 30);
        sigObj.keyPadAddHotSpot(4, 1, 18, 53, 59, 15);
        sigObj.keyPadAddHotSpot(5, 1, 106, 52, 40, 16);
        sigObj.keyPadAddHotSpot(6, 1, 186, 53, 35, 15);
```

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

```
MediaTracker mt = new MediaTracker(this);
rawImages = new Image[ numImages ];
String[] imageTitles =
        {
                "check",
                "checkboxes",
                "next_clear",
                "please",
                "prescription",
                "screen1",
                "Screen2",
                "thanks"
};

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, 128, rawImages[6]);
      sigObj.lcdWriteImage (0, 2, 10, 0, 100, 10, rawImages[4]);
      sigObj.lcdWriteImage (0, 2, 10, 64, 130, 40, rawImages[1]);
      sigObj.lcdWriteImage (0, 2, 70, 110, 100, 10, rawImages[2]);  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. In HotSpot(0), we check off the 1st option. HotSpot(1) checks off the 2nd option. HotSpot(2) moves to the next page, while HotSpot(3) clears the checked boxes.

```
if(sigObj.keyPadQueryHotSpot(0) !=0 && page==1)
      {
      sigObj.keyPadClearSigWindow(1);
      sigObj.clearTablet();
      sigObj.keyPadClearSigWindow(1);
      sigObj.lcdWriteImage(0, 3, 12, 65, 14, 13, rawImages[0]);
      }
else if(sigObj.keyPadQueryHotSpot(1) !=0 && page==1)
      {
      sigObj.keyPadClearSigWindow(1);
      sigObj.clearTablet();
      sigObj.keyPadClearSigWindow(1);
          sigObj.lcdWriteImage(0, 3, 12, 88, 14, 13, rawImages[0]);
      }
else if(sigObj.keyPadQueryHotSpot(2) !=0 && page==1)
      {
      sigObj.keyPadClearSigWindow(1);
      sigObj.clearTablet();
      sigObj.keyPadClearSigWindow(1);
              sigObj.lcdRefresh(1, 70, 110, 30, 15);
```

```
                    sigObj.lcdRefresh(2, 0, 0, 240, 128);
                    sigObj.lcdSetWindow(0, 74, 240, 118);
            sigObj.keyPadSetSigWindow(1, 0, 74, 240, 118);
                          page = 2;
                 }
        else if(sigObj.keyPadQueryHotSpot(3) !=0 && page==1)
                 {
            sigObj.keyPadClearSigWindow(1);
            sigObj.clearTablet();
            sigObj.keyPadClearSigWindow(1);
                    sigObj.lcdRefresh(1, 132, 110, 40, 15);
                    sigObj.lcdWriteImage(0, 2, 10, 64, 130, 40, rawImages[1]);
            Thread.sleep(300);
            sigObj.keyPadClearSigWindow(1);  sigObj.lcdRefresh(1, 132,
                    110, 40, 15);
            }
```

HotSpot(4) is used to go back to the previous page. HotSpot(5) is used to clear the signature data from the LCD. HotSpot(6) is used to accept the signature after checking to ensure it exists.

```
else if(sigObj.keyPadQueryHotSpot(4) !=0 && page==2)
        {
            sigObj.keyPadClearSigWindow(1);  sigObj.clearTablet();
            sigObj.keyPadClearSigWindow( 1 );
                    sigObj.lcdRefresh(1, 18, 53, 58, 15);
                    sigObj.lcdRefresh(0, 0, 0, 240, 128);
            sigObj.lcdWriteImage (0, 2, 10, 0, 100, 10, rawImages[4]);
           sigObj.lcdWriteImage (0, 2, 10, 64, 130, 40, rawImages[1]);
           sigObj.lcdWriteImage (0, 2, 70, 110, 100, 10, rawImages[2]);
                         sigObj.lcdSetWindow(0, 0, 0, 0);
            sigObj.keyPadSetSigWindow(1, 0, 0, 0, 0 );  page=1;
          }
        else if(sigObj.keyPadQueryHotSpot(5) !=0 && page==2)
          {
            sigObj.keyPadClearSigWindow(1);  sigObj.clearTablet();
            sigObj.keyPadClearSigWindow(1);
                    sigObj.lcdRefresh(1, 106, 52, 40, 16);
            Thread.sleep(300);  sigObj.keyPadClearSigWindow(1);
                    sigObj.lcdRefresh(2, 0, 0, 240, 128);
                    sigObj.keyPadSetSigWindow(1, 0, 74, 240, 118);
                 }
        else if(sigObj.keyPadQueryHotSpot(6) !=0 && page==2)
                 {
            sigObj.keyPadClearSigWindow(1);  sigObj.lcdRefresh(1,
            186, 53, 35, 15);

            if( sigObj.numberOfTabletPoints() > 0)
              {
                sigObj.lcdRefresh(0, 0, 0, 240, 128);
                signature = sigObj.getSigString(); //signature variable gets sig data
                sigObj.clearTablet();
                sigObj.keyPadClearSigWindow(1);
                             sigObj.lcdWriteImage(0, 2, 3, 46, 233, 24, rawImages[7]);
```

```
                                sigObj.setTabletState(1);
                                        sigObj.lcdRefresh(0, 0, 0, 240, 128);
                                sigObj.setLCDCaptureMode(1);
                                sigObj.setTabletState(0);
                                System.exit(0);
                              }
                                      else
                              {
                                        sigObj.lcdRefresh(0, 0, 0, 240, 128);
                                sigObj.clearTablet();  sigObj.keyPadClearSigWindow(1);
                                        sigObj.lcdWriteImage( 0, 2, 4, 46, 234, 20, rawImages[3]);
                                        sigObj.lcdRefresh(0, 0, 0, 240, 128);
                                        sigObj.lcdRefresh(2, 0, 0, 240, 128);
                                     sigObj.lcdSetWindow(0, 74, 240, 118);
                                sigObj.keyPadSetSigWindow(1, 0, 74, 240, 118);
                                         page = 2;
                                            //Start over again
                                           }
                             sigObj.keyPadClearSigWindow(1);
                             sigObj.clearTablet();
                          }
                         }
                           }
                  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