



SPIRAL SDK

Copyright ©2011 Thomas W Holtquist

www.skewworks.com

Version 1.0.8.0

Table of Contents

[Change Log](#)

[Introduction](#)

[1.1 System Features](#)

[1.2 Warnings](#)

[Skewworks.Spiral](#)

[2.1 Graphics.Initialize](#)

[2.2 Graphics.Brightness](#)

[2.2 Graphics.Calibrate](#)

[2.3 Graphics.Clear](#)

[2.4 Graphcis.DrawEllipse](#)

[2.5 Graphics.DrawImage](#)

[2.6 Graphics.DrawLine](#)

[2.7 Graphics.DrawRectangle](#)

[2.8 Graphics.FillRectangle](#)

[2.9 Graphics.GetPixel](#)

[2.10 Graphics.ModalStart](#)

[2.11 Graphics.ModalStop](#)

[2.12 Graphics.SaveToBMP](#)

[2.13 Graphics.SetCalibration](#)

[2.14 Graphics.SetPixel](#)

[2.15 Landscape and Portrait Mode](#)

Font Class

[3.1 Font.ComputeExtentEX](#)

[3.2 Font.ComputeTextInRect](#)

[3.3 Font.DrawText](#)

Skewworks.Spiral.Image

[4.1.1 Image.StreamImage](#)

[4.2.1 TinyBitmap](#)

[4.2.2 TinyBitmap.Clear](#)

[4.2.3 TinyBitmap.DrawEllipse](#)

[4.2.4 TinyBitmap.DrawLine](#)

[4.2.5 TinyBitmap.DrawRectangle](#)

[4.2.6 TinyBitmap.FillRectangle](#)

[4.2.7 TinyBitmap.Flush](#)

[4.2.8 TinyBitmap.GetBytes](#)

[4.2.9 TinyBitmap.GetPixel](#)

[4.2.10 TinyBitmap.GetPixelRaw](#)

[4.2.11 TinyBitmap.SaveToBMP](#)

[4.2.12 TinyBitmap.SetPixel](#)

[4.3.1 PictureBox Control](#)

Skewworks.Spiral.Input

[5.1 Prompt.Show](#)

[5.2 VirtualKeyboard](#)

[5.3 Textbox Control](#)

[5.4 Combobox Control](#)

[Skewworks.Spiral.Strings](#)

[6.1 Strings Class](#)

[6.2 StringSorter Class](#)

[Examples](#)

[7.1 Spiral Application Example](#)

[7.2 Creating a Font](#)

Change Log

February 15, 2012

Fixed loading of SIFs

Made GHI DLLs non-version specific

November 8, 2011

Fixed button state issue

Added Strings namespace

Added Combobox control

September 30, 2011

Fixed landscape mode calibration

Added Graphics.Brightness

Added 1ms sleep to end of Image.StreamImage

Image.StreamImage now properly closes FileStream

August 8, 2011

Emulation library has been removed

Fixed and updated Listbox

Increased render speed on RadioButtons

Added TinyBitmap class

Added PictureBox control

Added Textbox control (Skewworks.Spiral.Input)

Added VirtualKeyboard (Skewworks.Spiral.Input)

Added Graphics.SaveToBMP

June 29, 2011

IMPORTANT: Graphics class has been updated to require a call to Initialize() before using the library. This allows you to select pins for working with boards other than Panda II.

Calibrate and SetCalibration has been added to the Graphics class.

The Emulation class has been updated to match changes to Graphics.

1.0 Introduction

Spiral is a set of graphic interfaces designed to make development of .NET Micro Framework GUI applications more accessible on small devices. This release supports the [FEZ Panda II](#) or [FEZ Domino](#) with a [FEZ Touch](#) (or compatible LCD) module attached.

1.1 SYSTEM FEATURES

- Render large and full screen images straight from a uSD card
- Alpha-blending support
- Basic drawing commands (Line, Rectangle, Ellipse)
- Dynamic font loading
- Rendering of opaque and transparent backed text
- Modal prompt windows
- Built-in windowed controls
- Ability to create custom controls
- Touch-screen support
- TinyBitmaps

1.2 WARNINGS

Neither Skewworks nor Thomas W Holtquist is responsible for any harm to you or your devices that may occur from the use of this software. Spiral is provided **as is** and you use it at your own risk.

2.0 Skewworks.Spiral

Graphics is a static class inside the Skewworks.Spiral namespace and assembly. This class handles initializing the LCD and Touch Screen automatically and contains all the standard graphics calls.

2.1 GRAPHICS.INITIALIZE

This method must be called before using Spiral. It can either be called with no parameters (to start Spiral with the FEZ Touch on the Panda II) or with the required pins to use any compatible LCD on any compatible GHI board.

Graphics.Initialize now supports an optional Boolean parameter “Landscape” so you can start Spiral in Landscape mode instead of starting in Portrait and then switching to Landscape.

2.2 GRAPHICS.BRIGHTNESS

Call this method to control the brightness of the LCD backlight with an integer value. Accepted values are from 0 – 100; values above or below this range will automatically be adjusted to fall within the accepted range.

2.3 GRAPHICS.CALIBRATE

Calling this method sets Spiral into calibration mode and returns a set of points used to restore calibration the next time Spiral is started. It is recommended that you save the points using EWR (Extended Weak References) and restore them each time Spiral starts.

2.4 GRAPHICS.CLEAR

Clears the screen either full black or with an optionally supplied value.

2.5 GRAPHICS.DRAWELLIPSE

This method draws an ellipse either with or without a border. You can also supply an Opacity value of 0.0 (transparent) to 1.0 (opaque).

2.6 GRAPHICS.DRAWIMAGE

Allows you to draw an image directly to the screen. This method requires you have a byte array of 565 pixel values (2 bytes per pixel) as well as a supplied height and width. Supplying values that do not correspond with the actual dimensions of the image will produce unexpected results.

This method does **not** support TinyBitmaps, to draw TinyBitmaps to the screen you must either add them to a PictureBox or call the TinyBitmap.Flush() method.

2.7 GRAPHICS.DRAWLINE

Creates a line on the screen using the coordinates supplied. Alpha-blending is not currently supported on this method.

2.8 GRAPHICS.DRAWRECTANGLE

This method draws a rectangle either with or without a border. You can also supply an Opacity value of 0.0 (transparent) to 1.0 (opaque).

2.9 GRAPHICS.FILLRECTANGLE

When you need to draw a solid rectangle without a border consider using this method; it will execute slightly faster.

2.10 GETPIXEL

Gets the RGB value of a given pixel location.

2.11 GRAPHICS.MODALSTART

ModalStart is an advanced method used by features like Skewworks.Spiral.Input. This begins a secondary touch gathering thread and sends all responses directly to the form supplied. If you intend to use this method it should be called in conjunction with a *ManualResetEvent* to block other threads. Only one modal thread is supported at a time.

2.12 GRAPHICS.MODALSTOP

This method is called to end modal operations.

2.13 GRAPHICS.SAVETO BMP

Spiral is capable of taking a snapshot of the entire screen and saving it to a standard BMP file using this method. You are responsible for starting whatever storage medium you plan to use (such as with PersistentStorage).

WARNING: This method requires scanning every pixel on the screen and so takes in excess of 4 minutes to complete; it is only recommended for use to take screenshots of commercial products.

2.14 GRAPHICS.SETCALIBRATION

This method allows you to supply an array of 3 points to set the Touch Screen calibration; it should be used with the array supplied by Graphics.Calibrate().

2.15 GRAPHICS.SETPIXEL

Sets a specific pixel to the RGB value supplied.

2.16 LANDSCAPE AND PORTRAIT MODES

By default Spiral runs in *Portrait* mode, however you can switch at any point to *Landscape* mode by setting Graphics.LandscapeMode = true. Switching modes will automatically redraw your active form unless you set Graphics.AutoUpdate = false. Additionally when you switch between modes your forms will automatically update their X, Y, Width and Height values to reflect the screen orientation the next time they are rendered.

3.0 Font Class

The Font class resides inside the main Skewworks.Spiral assembly and allows you to dynamically load fonts from a file or resource. Spiral fonts must be 1bpp fixed and width. It is recommended you use the Spiral Font Creator.

3.1 FONT.COMPUTEEXTENTEX

This method returns the width and height of a given string, including new line characters.

3.2 FONT.COMPUTETEXTINRECT

Returns the render width and height of a given string inside of a given maximum width.

3.3 FONT.DRAWTEXT

This method draws a string at the given location. You may optionally also supply a height and width to render in.

4.0 Skewworks.Spiral.Image

Skewworks.Spiral.Image has been upgrade to include streaming from files, a PictureBox control, and the TinyBitmap class.

4.1.1 IMAGE.STREAMIMAGE

This static method allows you to stream Sprial formatted images (*.sif) created with the Spiral Image Converter straight to the LCD. This is useful for large images that cannot fit inside RAM.

4.2.1 TINYBITMAP

TinyBitmap allows you to create Bitmap like images inside Spiral. These images can be placed inside PictureBoxes, drawn to the screen and even saved to a standard bitmap file.

4.2.2 TINYBITMAP.CLEAR

Clears the image either full black or with an optionally supplied value.

4.2.3 TINYBITMAP.DRAWELLIPSE

This method draws an ellipse either with or without a border.

4.2.4 TINYBITMAP.DRAWLINE

Creates a line on the image using the coordinates supplied. Alpha-blending is not currently supported on this method.

4.2.5 TINYBITMAP.DRAWRECTANGLE

This method draws a rectangle either with or without a border.

4.2.6 TINYBITMAP.FILLRECTANGLE

When you need to draw a solid rectangle without a border consider using this method; it will execute slightly faster.

4.2.7 TINYBITMAP.FLUSH

Draws the image directly to the screen at the coordinates provided.

4.2.8 TINYBITMAP.GETBYTES

This method returns all the bytes associated with the image (Bitmap header not included) formatted as 2 bytes per pixel.

4.2.9 TINYBITMAP.GETPIXEL

Gets an RGB for the specified pixel.

4.2.10 TINYBITMAP.GETPIXELRAW

Gets the raw ushort value for the specified pixel.

4.2.11 TINYBITMAP.SAVETO BMP

Saves the image as a standard BMP file. You are responsible for starting whatever storage medium you plan to use (such as with PersistentStorage).

NOTE: This method does not require scanning the LCD and so executes significantly faster than Graphics.SaveToBMP. A 50x50 pixel image saves in ~0.7 seconds.

4.2.12 TINYBITMAP.SETPIXEL

Sets the specified pixel within the image to the color value supplied.

4.3.1 PICTUREBOX CONTROL

Since the PictureBox control depends on the TinyBitmap class to function it is located within Skewworks.Spiral.Image rather than inside Skewworks.Spiral with all other controls (except textbox which is inside Skewworks.Spiral.Input).

5.0 Skewworks.Spiral.Input

The Skewworks.Spiral.Input namespace and DLL contain all the classes and controls associated with gathering text or modal input from the user.

5.1 PROMPT.SHOW

This static method displays a modal dialog to the user requesting that they respond by pressing a button or buttons on the form. This is also useful for displaying information or warnings.

5.2 VIRTUALKEYBOARD

The VirtualKeyboard is a class that allows you to display a virtual keyboard to the user, modally, and returns the text entered. You can supply a default text and a password character if desired.

5.3 TEXTBOX CONTROL

Since the Textbox relies on the VirtualKeyboard it is found inside of Skewworks.Spiral.Input instead of the top level Skewworks.Spiral DLL like the rest of the controls (except PictureBox which is located in Skewworks.Spiral.Image).

5.4 COMBOBOX CONTROL

The Combobox control allows you to present a list of options in a more space friendly way than the Listbox. As this control requires modal access it is found in the Input namespace.

Additionally, unlike other controls, the Combobox does not require Height to be set at creation and has height requirements. The height of any given Combobox cannot be less than 4 pixels bigger than its font or 12 pixels (whichever is greater).

6.0 Skewworks.Spiral.Strings

The Skewworks.Spiral.Strings namespace and DLL contain all the classes associated with working with strings.

6.1 STRINGS CLASS

This static class contains several helpful string methods including a replace method.

6.2 STRINGSORTER CLASS

This class allows you to provide a list of strings to be sorted in Ascending, Descending and Case Insensitive modes; which is useful for sorting options for Listboxes and the like.

7.0 Examples

7.1 SPIRAL APPLICATION EXAMPLE

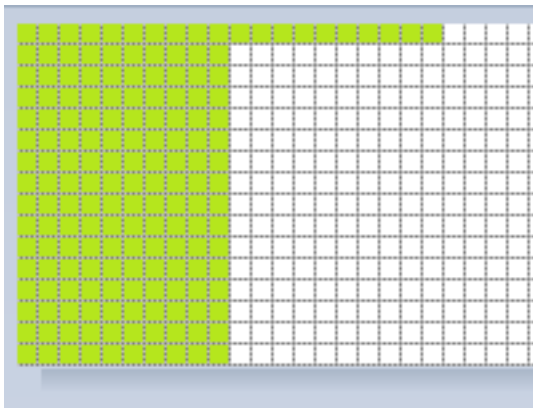
You can find an example project here: [InstallDir]\Skewworks\Spiral\Demo Project\SpiralDemo.sln

7.2 CREATING A FONT

Since all fonts must be fixed width the first thing to do is decide the dimensions you want your font to be.

Next create a bitmap that size in your favorite graphics editor.

An easy way to figure out your spacing is to start by drawing a green line along the very top of your image that is **twice** the width of your font. Next fill in a green rectangle that is exactly the size of your font on the left, like below.



Now copy and paste this over and over until it fills your entire image. You'll be able to easily visualize where one character ends and the next begins.

Once you're done simply draw the characters one at a time. You'll want to run CharMap.exe to help you draw the images in order. Start from the "!" character (which comes first) and go until you reach the "~" character.

Now simply remove all the green pixels and you're ready to create your font.