PEG+™

Product Brief

Swell Software, Inc.

(810) 982-5955
Fax: (810) 982-5949

www.swellsoftware.com

Graphics Software For Embedded Systems

Summary

The PEG+ (Portable Embedded GUI) library is a professional quality graphical user interface library created to meet the needs of embedded systems developers. Unlike the alternatives, PEG+ is small, fast, and easily ported to virtually any hardware configuration capable of supporting graphical output. PEG+ also delivers exceptional visual appeal and an intuitive and powerful API for application-level user interface development.

PEG+ simply provides the most complete GUI solution available to real-time embedded systems developers. All of the utilities, documentation, support, and development tools you will require for creating a graphical user interface on an embedded system are included with the PEG+ library development package.

PEG+ is not an operating system, but instead is designed from the ground up to work seamlessly in a real-time multitasking environment. PEG+ allows you to choose the real-time operating system which best meets your needs and provides the tools, API, and class library that will allow you to quickly create an outstanding graphical presentation.

PEG+ is licensed on a per-developed-product basis, eliminating royalty fees. PEG+ is delivered with full source code, many example application programs, a complete set of supporting utility programs, hardware interface objects for several common video configurations and input devices, and very thorough documentation to simplify the integration process.

Features

Programming Model

The PEG+ library is written entirely in EC++ and implements an event-driven programming paradigm at the application level. The event-driven paradigm has proven to be a superior method for creating user interface software, providing structure and order to the otherwise difficult task of responding to external system events arriving asynchronously from many sources. This programming model has the added benefit of being easily integrated into real-time multitasking environments.

Reduced Size

PEG+ achieves a small footprint through several means, including heavy reliance on EC++ inheritance to encourage code re-use. Each control type is built incrementally upon its predecessor, allowing you to select and use only the objects which meet your requirements without forcing you to carry along any unneeded baggage. In addition, PEG+ is written with the embedded market firmly in mind, meaning that the value of every feature is weighed against the code size and performance requirements of that feature.

In addition the PEG+ library is configured at compile time to include only the features your application requires. A large set of configuration flags allow you to completely define the set of input devices, drawing primitives, and many higher level features that your target will require. This configuration capability allows you to remove at compile time any library features that your application does not require, in essence allowing you to customize the library to exactly meet your needs.

Maximum Performance

PEG+ achieves maximum performance by minimizing the system overhead required to maintain a graphical presentation. This includes support for advanced window and viewport clipping techniques that prevent unnecessary screen drawing. In addition, PEG+ always interacts directly with video and input hardware to achieve the greatest possible throughput.

Highly Portable

PEG+ achieves portability primarily by abstracting all hardware dependent functionality. The vast majority of the PEG+ library is completely hardware independent, relying on well-defined hardware interface objects to provide a consistent, simple, and reliable set of I/O methods. PEG+ applications are running today on all of the most common embedded processors including x86, ARM, PowerPC, MIPS, StrongARM, ColdFire, DragonBall, OMAP, BlackFin, ST3500, and CPU32 based microprocessors.

Portability is further achieved by making no assumptions about the software environment. OS-specific dependencies are encapsulated and well documented, allowing users to follow the examples provided to create new interface classes for any environment. PEG+ further avoids the use of intrinsic data types and floating point math to eliminate problems associated with CPU word length differences and the common lack of floating point support

Finally, portability is increased through general conformity to the EC++ standard. PEG+ does not require support for C++ exception handling or run-time type identification and can be compiled in full EC++ compatibility mode. PEG+ is fully verified with a large set of popular compilers for embedded systems (below).

Flexible and Powerful API

The PEG+ library provides an intuitive and robust object hierarchy. Objects may be used directly as provided, or enhanced through user derivation. PEG+ imposes no artificial limits on how objects are used, meaning that the user is free to design screens with an indefinite nesting level of controls within windows within other windows. You will find that you can do things very quickly and easily with PEG+ -- things that would be very difficult and time-consuming using the mainstream desktop GUI programming environments.

The default appearance of PEG+ objects is almost identical to the common desktop graphical environments. This appearance can, of course, be enhanced or simplified to fit the requirements of the application. In fact users often prefer to create a user interface with a very custom appearance and no resemblance to the desktop graphical environments. This is also readily accommodated within the design of PEG+.

The PEG+ API is defined entirely by the public functions provided by the library classes. This API provides robust and intuitive methods for performing even very complex graphical operations. We encourage you to obtain an evaluation copy of the library and experiment with PEG+ to insure that the library meets your project's requirements.

Real-Time Awareness:

PEG+ is fully integrated with RTOS messaging, memory management, and synchronization services. This yields the lowest possible overhead and the only true real-time multi-tasking GUI environment available. PEG+ input devices are interrupt driven, and again use RTOS services to communicate user input information to the graphical user interface.

Development Environments

Microsoft Windows Development Environment

Very often during the early stages of an embedded development effort, it is difficult or impossible to do software testing and debugging on the intended target platform. For this reason PEG+ provides a set of hardware and OS encapsulation classes which allow your PEG+ user interface to run as a standard 32-bit Windows application. ***You can create, test, and debug your entire user interface while using the very mature Windows application development tools***. Once your target platform is available, you simply have to replace the PEG+ interface encapsulation classes with versions provided for your target.

MS-DOS Development Environment

Building on the above concept, PEG+ also comes complete with interface classes for running PEG+ as a DOS real-mode, 16-bit protected mode, or 32-bit flat mode application program. This is often an advantage for users who intend to use some form of x86 CPU on the final target system. Porting to the final target is again simply a matter of rebuilding the PEG+ library with I/O interface classes designed for your target. Your application level GUI software is guaranteed to run without modification on the target platform.

X11 Development Environment

Similar in functionality to the Microsoft Windows Development Environment, the X11 Development Environment allows you to code, test and debug your application while running your application on the X11 Windowing System. The added advantage of this environment is that you may be running X11 on top of the same OS that acts as your embedded OS. This allows you to have a clear representation of your final application running on it's target OS while still having

access to all of your desktop tools. Again, porting to your final target is simply a matter of recompiling the PEG+ library with I/O interface classes designed for your target.

Multi-Tasking

PEG may be configured to support multiple GUI tasks. These tasks can be of differing priorities and can each **directly** create, display, and control any number of GUI windows or child controls. **This advanced capability is unique to the design of PEG**. In this configuration, PEG protects internal data structures from corruption through judicial use of semaphores, which again are provided by the underlying OS. This tasking model has several advantages, the most significant being that the application level programming is greatly simplified. At any time any task in the system can directly display a window or any other type of GUI object and update the information displayed within that object.

Finally, PEG can also be run standalone without a multitasking kernel. This model is most often used in smaller, less complex applications.

Compiler Support

The PEG+ library has been fully verified with each of the following compilers:

Compiler	Processor
• Analog Devices Visual DSP	Blackfin DSP
• ARM Developer Suite Ver. 1.2	All ARM Cores
• ARM RealView (RVCT) Ver. 2.1	All ARM Cores
• Borland C++ Ver. 4.52, 5.0	x86 real mode x86 protected-mode
• CAD-UL C++	x86 32-bit
• GNU C++	All Unix/Linux environments
• Green Hills Multi	All ARM Cores, StrongARM PowerPC, SH3, MIPS, 68K PXA, x86, MCORE, Freescale i.MX
• Hitachi	All SH Series
• IAR Workbench	All ARM Cores
• MetaWare High C/C++	All ARM Cores,x86 32-bit All ARC Cores
• metroWerks CodeWarrior	All ARM Cores, 68K, MCORE Coldfire, DragonBall, PowerPC Freescale i.MX
• Microsoft MSVC++ Ver.1.52, 5.0, and 6.0, .Net	x86 real-mode x86 protected-mode.
• Paradigm	All x86 Targets

• Sybase (Watcom) C++	x86 real-mode x86 32-bit protected mode
• ST Microelectronics	All ST series
• Tasking for C167	
• Texas Instruments Code Composer	All TI Cores

Input Devices

Pointing Device

PEG+ includes complete support for mouse, joystick, and/or touch screen input. This includes drawing various pointer bitmaps or using hardware cursor capabilities. Complete mouse input drivers and touch screen input drivers are provided for each of the reference platforms. All pointer input functionality can also be removed from the library simply by turning off a library configuration flag prior to compiling the library.

When operating with a touch screen, PEG+ can be configured to eliminate drawing of the mouse pointer and can also be configured not to highlight the object that has input focus. These options further reduce code size and drawing overhead when running with touch screen input. When running with a touch screen PEG+ also eliminates the need for "pointer move" messages and thus all operations and controls work normally with only touch and release input messages. This feature simplifies touch screen input driver development.

Keyboard/Keypad

Full keyboard support is optionally included in the PEG+ library. This includes all handling required for keyboard navigation through menus, windows, and dialogs. Keyboard support may range from a full QWERTY keyboard to a small set of user-defined membrane keys. Full navigation and operation can be accomplished with as few as three unique input keys.

Soft Keys

Using soft keys, i.e. membrane keys placed at the perimeter of the display screen, is also supported. When using soft keys, the system programmer configures PEG+ to operate as with a touch screen, and sends to PEG+ touch-screen click messages corresponding to the screen position adjacent to each membrane key.

Language Support

PEG+ provides industry leading support for multi-lingual applications. PEG+ fully supports two-byte characters and Unicode string encoding. Our ***CompositeFont*** technology provides an industry-leading solution for incorporating even very large character sets in memory-limited embedded systems. Virtually any language can be supported using any combination of character sets including Latin, Cyrillic, Han, Katakana, Hiragana, etc., in a single PEG+ application. This capablity gives the system developer unequaled range and flexibility in designing products for foreign markets.

The PEG+ library provides a full complement of compiler-independent 'C' string library functions, eliminating the need for any special non-ANSI compiler support for 2-byte characters.

PEG+ also provides complete string table editing and maintenance capabilities. System developers use this feature to enter and edit all strings for all supported languages. The PEG+ ***String Table Editor*** allows you to edit a string in any language using only a mouse or Latin (ASCII) keyboard. **SJIS** and **Unicode** data entry formats are also supported. The String Table Editor exports your system strings in a unique source file while at the same time mapping each string to the character set or character sets you have specified for you target system.

Control Types

PEG+ supports a rich and growing compliment of GUI object types including:

- **PegAnimationWindow**
- **PegBitmap**
- **PegBitmapButton**
- **PegBitmapConvert**
- **PegBitmapLight**
- **PegButton**
- **PegCapture**
- **PegChart**
- **PegCheckBox**
- **PegCircularBitmapDial**
- **PegCircularDial**
- **PegColor**
- **PegColorLight**
- **PegComboBox**
- **PegDecoratedButton**
- **PegDecoratedWindow**
- **PegDial**
- **PegDialog**
- **PegEditBox**
- **PegEditField**
- **PegFileDialog**
- **PegFiniteDial**
- **PegFiniteBitmapDial**
- **PegFont**
- **PegGroup**
- **PegGifConvert**
- **PegHorizontalScrollBar**
- **PegHorizontalList**
- **PegIcon**
- **PegImageConvert**
- **PegJpgConvert**
- **PegLight**
- **PegLinearScale**
- **PegLinearBitmapScale**
- **PegLineChart**
- **PegList**
- **PegMenu**
- **PegMenuBar**
- **PegMenuButton**
- **PegMenuDescription**
- **PegMessageQueue**
- **PegMessageWindow**
- **PegMLMessageWindow**
- **PegMLTextButton**
- **PegMultiLineChart**
- **PegNotebook**
- **PegPresentationManager**
- **PegPngConvert**
- **PegPoint**
- **PegProgressBar**
- **PegProgressWindow**
- **PegPrompt**
- **PegQuant**
- **PegRadioButton**
- **PegRect**
- **PegScale**
- **PegScreen**
- **PegScroll**
- **PegScrollInfo**
- **PegSlider**
- **PegSpinButton**
- **PegSpreadSheet**
- **PegStatusBar**
- **PegStripChart**
- **PegTable**
- **PegTextBox**
- **PegTextButton**
- **PegTextThing**
- **PegThing**
- **PegTimer**
- **PegTitle**
- **PegToolBar**
- **PegToolBarPanel**
- **PegTreeNode**
- **PegTreeView**
- **PegVertList**
- **PegVerticalPrompt**
- **PegVerticalScrollBar**
- **PegWindow**
- **PegZip/PegUnzip**
- **Peg2DPolygon**

Video Output

Overview
PEG is designed work with and take full advantage a broad range of video output devices and display screens. PEG can be configured for monochrome, 4 grays, 16 grays, 16 colors, 256 colors (in palette or packed formats), 65K colors (in 5:5:5 or 5:6:5 formats), true 24-bit RGB, and 32-bit RGB-Alpha color output encoding. Further, the output color depth may be defined at compile time (producing the smallest code) or at run time (allowing the video output device to be determined during system initialization).

A full range of VGA and LCD display devices are supported, including LCD devices of unique x-y resolutions or orientations. The design of the PEG+ display drivers enables common resolutions such as 640x480 VGA or 320x240 LCD screens to be handled in an identical manner as very unique or very small x-y pixel resolutions.

All PEG+ screen interface operations are performed by a library class named PegScreen. This class defines the drawing primitives and other operations that are available in every PEG+ system, regardless of display type or video controller in use. Specific derived versions of PegScreen are then provided for each color depth, resolution, and video controller.

Hardware Acceleration
PEG+ takes full advantage of video controllers which support hardware acceleration capabilities such as hardware cursor or hardware bit-blit. These capabilities are always provided in the display driver software, via software emulation when the target controller does not provide a specific feature directly in hardware. When hardware acceleration is available, a small set of functions in the display driver are reduced to take advantage of the video hardware acceleration.

Double-Buffering
Double-buffered video output is optionally supported in every PEG+ configuration. This configuration allows all intermediate drawing operations to be performed to an off-screen or local memory buffer. At the conclusion of a drawing operation, the invalidated region of the local memory buffer is transferred to the visible video memory, using hardware bit-blitting if provided. This mode of operation provides flicker-free animation and scrolling. Double-buffered output, while always supported, is not required. PEG+ can also be configured to do all drawing directly to visible video memory.

Fonts
PEG+ supports an unlimited number and style of fonts. Fonts can be basic binary, outlined, or anti-aliased format. Users can produce any number of custom fonts using the PEG Development Toolkit.

Graphics

PEG+ provides advanced facilities for display of bmp, png, jpg, and gif formatted images. The capabilities are completely ROMable and can be used on any embedded system.

Screen Driver Templates

A full set of screen driver templates are provided for 1-bpp (monochrome) through 24-bpp (TrueColor RGB) video output. These template drivers are designed to work with any CPU architecture that supports direct, linear access to the video memory buffer.

The screen driver templates are capable of supporting any screen resolution from 1 x 1 to 65535 x 65535 pixels.

Accelerated Screen Drivers

Many embedded controllers such as the Elan, ARM, and PowerPC CPUs provide integrated video control functionality with limited acceleration features. These controllers work best using one of the template drivers listed above.

On the other hand, several popular external video controllers are also applied to embedded applications where higher performance is required. PegScreen driver classes tuned to take advantage of the hardware acceleration features of many of these external video controllers are also available. Customized driver classes are currently available for the following video controllers:

- **Advanced Micro Devices** – Geode processor, Elan
- **ATI –** Rage Mobility, Mobility Radeon
- **All ARM Cores-** including ARM7/ARM9, Samsung, Thumb Mode
- **Cirrus Logic –** GD5430, 71110, 7212, 7312
- **Epson –** S1D13300, S1D13503, S1D13504, S1D13505, S1D13506, S1D13704, S1D13705, S1D13706, S1D13806, S1D13A04/S1D13A05, SPC8106 VGA LCD/CRT Controller
- **Freescale**- i.MX1 and i.MXL, i.MX21, PowerPC 823/860
- **Fujitsu**-Orchid, Scarlet
- **3Dlabs-** Permedia II
- **Intel –** PXA 250/255
- **Linux** - Linux Framebuffer Device
- **Linux, Solaris, NetBSD, Lynx OS -**X11 Windows
- **Chips & Technology -** CT545 Alpine, CT65550, CT69000/69030
- **MediaQ -** MQ200 & MQ400
- **Philips-** Trimedia
- **Sharp LH Series Controllers -** LH79531, LH77790, LH79520 256 color, LH79524, LH75401, LH7A400
- **Silicon Motion –** Lynx 3DM, LynxEM+, SM501
- **ST Microelectronics** – STV3500
- **Topro** – TP6508 controller
- **Texas Instruments** – OMAP, DM270, DM320 DSP color (for RSA environment)
- **X86 Standard VGA**
- **X86 VESA Extended Modes**

Custom PegScreen drivers for additional video controllers are available on request.

Development Environment Screen Drivers
As stated above, it is often useful to run PEG+ in a PC development environment regardless of your final target architecture. This allows software and hardware development to proceed in parallel, shortening time to market for a new design. The PEG+ development package always ships with the following PegScreen driver classes to facilitate this type of concurrent development:

- Generic VGA- Runs on any PC-compatible in any processor mode.
- Win32- Runs on any PC-compatible running MS Windows 95 or later.
- X11 – Runs on any PC-compatible running X11 R6 including XFree86 3.3.x

Drawing Primitives

While PEG+ applications rely primarily on the provided PEG+ class library objects for window and control drawing, it is often useful to perform custom drawing at the application level. For this reason all PEG+ drawing primitives can be invoked at any time by the user's application level software. The PEG+ drawing primitives include:

Function	Comments
BeginDraw	Begins a sequence of drawing operations.
Bitmap	Draws a bitmap at the desired location.
BitmapFill	Tiles a bitmap to fill a given area.
Capture	Captures an area of the screen.
Circle	Any color outline or fill, optional fill, any outline width.
CreateBitmap	Used to draw offscreen or for animations.
DeleteFont	Delete font created with MakeFont.
DrawText	Any color, position, font, transparent background or fill.
Ellipse	Any color border or fill, any outline width.
EndDraw	Ends a sequence of drawing operations.
GetPointerType	Returns mouse pointer type.
GetXRes	Display x resolution, in pixels.
GetYRes	Display y resolution, in pixels.
HidePointer	Removes the mouse pointer from the screen.
Invalidate	Only invalid screen regions may be drawn to.
Line	Any width, color, orientation.
MakeFont	Create bitmapped font from vector font.
PatternLine	Any width, color, orientation, pattern.
PlotPoint	Any color.
Polygon	Outline and/or fill, pattern fill, any outline width.
Rectangle	Outline and/or fill, pattern fill, any outline width.
RectangleXOR	Performs logical XOR operation with pixels in region.
RectMove	Used for scrolling and animation.
ResetPalette	Restores default palette.
Restore	Restore previously captured screen area.
RestorePointer	Restores hidden mouse pointer.
SetPointer	Sets mouse pointer position
SetPointerType	Sets mouse pointer shape
SetPalette	Allow loading custom palette.
TextHeight	Returns height of string in current font, in pixels.
TextWidth	Returns width of string in current font, in pixels.

All drawing primitives enforce object clipping and viewport validation, thus preventing run-time address errors due to invalid parameters being passed to drawing functions.

PEG Developers Toolkit

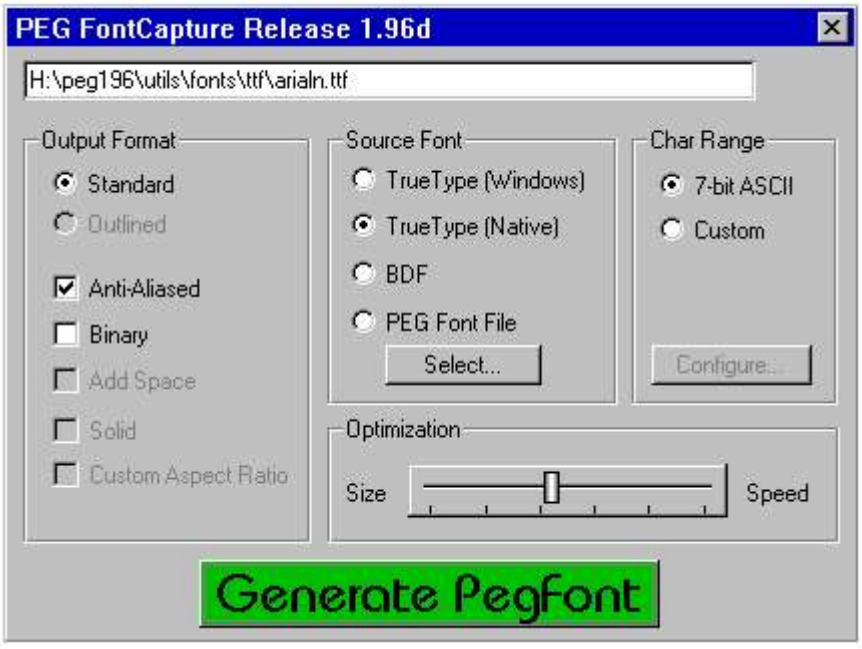
PEG+ is delivered with a full set of utility programs useful for embedded developers. These utility programs allow you to generate and use your own fonts, convert and ROM several forms of bitmap images, and design and automatically generate the source code for your PEG+ windows and dialogs.

FontCapture
PEG supports many font formats. These include a custom 1-bit-per-pixel variable-width bitmapped font format, a scalable vector font format, outlined fonts, and anti-aliased fonts.

FontCapture is a development tool that allows users to convert TrueType and BDF font files into the native font format required internally by PEG. The output of FontCapture is a source file containing "C" style data arrays describing the captured font. These source files are then compiled and linked in with your application, allowing you to associate any number of custom fonts with any PEG control type that displays string data.

FontCapture includes a complete character editor allowing the user to customize individual characters of the generated font. Unicode character mapping and multi-page fonts containing thousands of characters are supported. FontCapture is available for Windows, Linux/X11, and Solaris hosts.

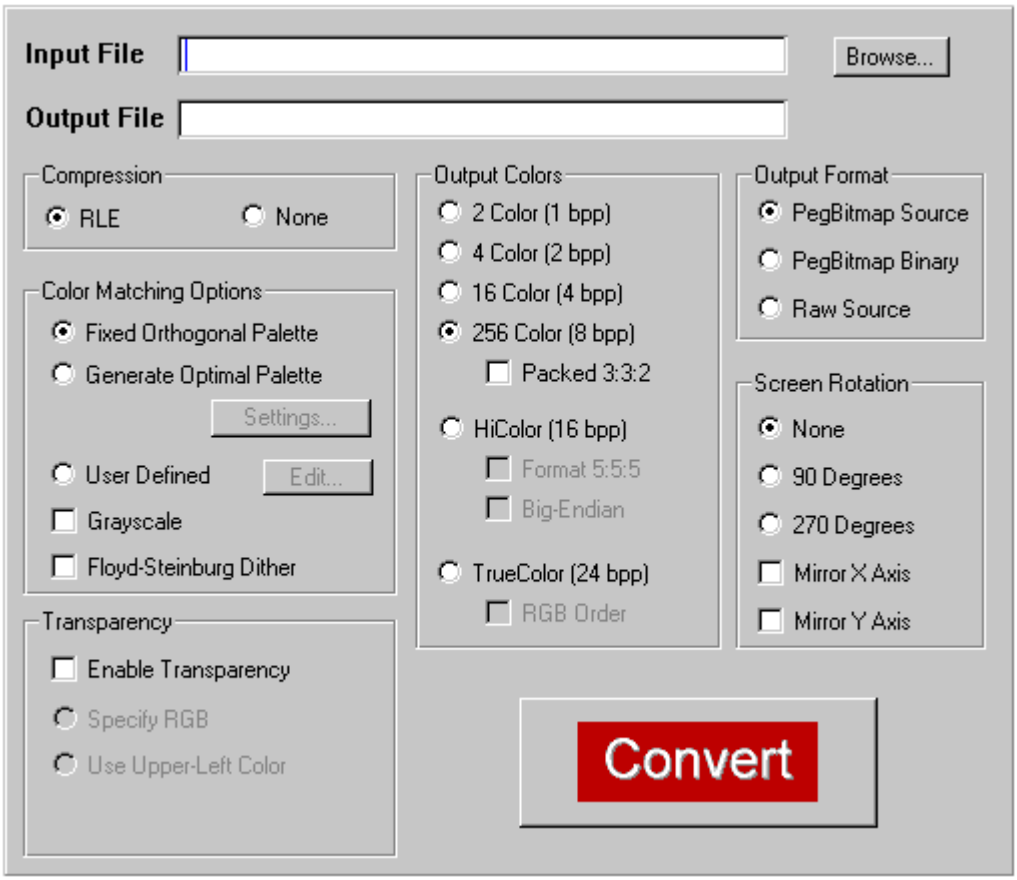
The image below is a screen shot of the FontCapture utility program.



ImageConvert

Image Convert is a development tool that converts BMP, GIF, JPEG, and PNG images into a compressed format supported by the PEG+ bitmap functions. The output of ImageConvert is a 'C' source file that can then be compiled and linked in with your application. ImageConvert will optionally compress the output bitmaps on an individual basis, allowing you to achieve the best mix of speed vs size.

Image Convert also creates optimized palettes for applications running in 4 or 8 bpp mode. The user can input any combination of supported image files, and ***ImageConvert*** will perform advanced quantization and color-reduction producing an optimal palette or palettes corresponding to the input images. The output images are then automatically re-encoded using the newly generated palette information. Multiple palettes can be used in one application. ImageConvert also supports dithering of pixel color data when reducing the color depth of an input image to match the output display capabilities. The image below is a screen shot of the ImageConvert utility program.



PEG Window Builder™

PEG WindowBuilder is a powerful visual design tool created for use with the PEG+ library. PEG WindowBuilder allows the developer to quickly create and use custom windows and dialogs. The output of PEG WindowBuilder is the EC++ source code required to create the desired window or dialog. PEG WindowBuilder also generates the prototype message handling code required to process signals received from the window or dialog child controls.

PEG WindowBuilder is actually a PEG+ application program running in the Win32 development environment. This guarantees that the operation of PEG WindowBuilder is fully and completely WYSIWYG, and your target will appear exactly as it does in the PEG WindowBuilder environment.

The developer using PEG WindowBuilder can import any number of custom fonts and apply those fonts to PEG+ controls using drag-and-drop techniques. The PEG+ controls immediately update to use the custom font, and the source code produced also configures each control to use the font selected.

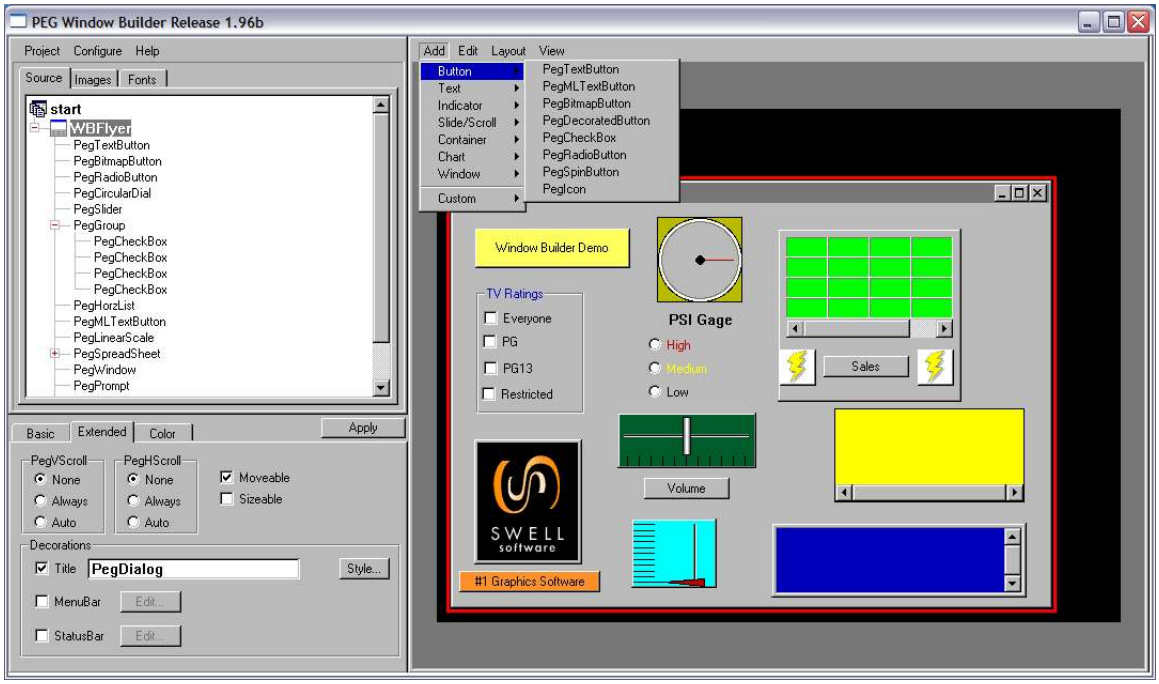
Likewise any number of graphics files may also be imported into the PEG WindowBuilder project environment. Imported graphics may be any combination of .bmp, .jpg, or .gif files. PEG WindowBuilder will re-encode these files, performing dithering and color reduction if required, to insure that the graphics are ready to use on the target platform. Imported graphics are written out to 'C' style data arrays, ready to be compiled and linked into the target system software.

Once a graphic has been imported into the PEG WindowBuilder environment, it can be applied to any PEG+ object which supports bitmap decorations. The source code generated by WindowBuilder will perform all the necessary steps to create the controls and assign the correct bitmaps to each control. While in the WindowBuilder environment, the graphics appear on the selected controls exactly as they will appear on the final target.

We strongly believe that PEG Window Builder™ is simply the most powerful GUI application development tool available to embedded developers at any price!

As with all PEG+ supporting tools, PEG WindowBuilder is essential to making PEG+ a complete embedded GUI solution and is therefore included with the PEG+ library at no extra charge.

The following is a screen shot of the PEG WindowBuilder™ application program:



Requirements

PEG+ is designed to work with any C++ compiler/debugger combination and any embedded CPU. Since PEG+ is provided in source form, you can usually just compile the PEG+ source files, generate the PEG+ library, and link the library into your target software. Support is provided for any user who experiences difficulty using PEG+ with a specific toolset.

For multi-tasking environments, the target real-time operating system must support means for inter-task communication via messages. In addition, PEG+ provides high-level timer services which are only operational if the underlying OS supports a periodic timer interrupt service. When PEG+ is configured to support multiple GUI tasks running at different priorities, the OS must support means for semaphore protection of critical code sections. These requirements are trivial in quality commercial real-time operating systems.