

smxPEG

Portable Embedded GUI

Summary

PEG is a Portable Embedded GUI library and development tool set designed specifically for embedded systems. PEG's small footprint, high performance and event-driven programming model make it perfect for today's multi-threaded embedded applications using LCD and video displays and controllers.

Besides an API and Class Library, PEG includes a complete set of development tools that are listed below. These tools are designed to help you reduce development time by making it easier to create professional-quality, multi-lingual, full-featured embedded GUIs.

PEG Development Tool Set

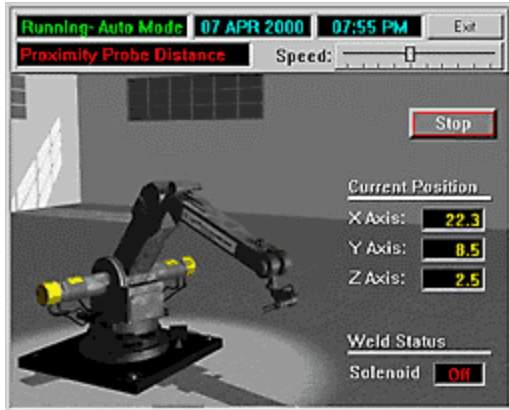
WindowBuilder	<ul style="list-style-type: none">• Create custom windows and dialogs• 100% WYSIWYG• Generate "C++" source code• Produce multi-lingual applications
FontCapture	<ul style="list-style-type: none">• Make your own fonts• Capture TrueType and BDF fonts• UNICODE support• Generate "C++" source code
ImageConvert	<ul style="list-style-type: none">• Convert .bmp, .gif and .jpg images into compressed format• ROM bitmap images• Generate "C++" source code

Features

- Designed specifically for embedded systems
- Royalty free
- Includes library, API and complete set of development tools
- Small footprint – 100K code, 4K stack, 8K RAM for full-featured GUI
- Completely ROMable
- Full C++ source code included
- Fast – interacts directly with video and input hardware
- Supports popular target processors, video controllers, and I/O devices
- Easy to port to new hardware – well-defined hardware interface. Screen driver templates included.
- Event-driven programming model – familiar to Windows developers
- Reduce development time – Use PC to design GUI before target hardware is ready
- Multi-lingual support – including 2-byte character sets and UNICODE string encoding
- Outstanding technical support

Look and Feel

PEG's default appearance is almost identical to Windows. This makes it easy for Windows programmers to get up and running.

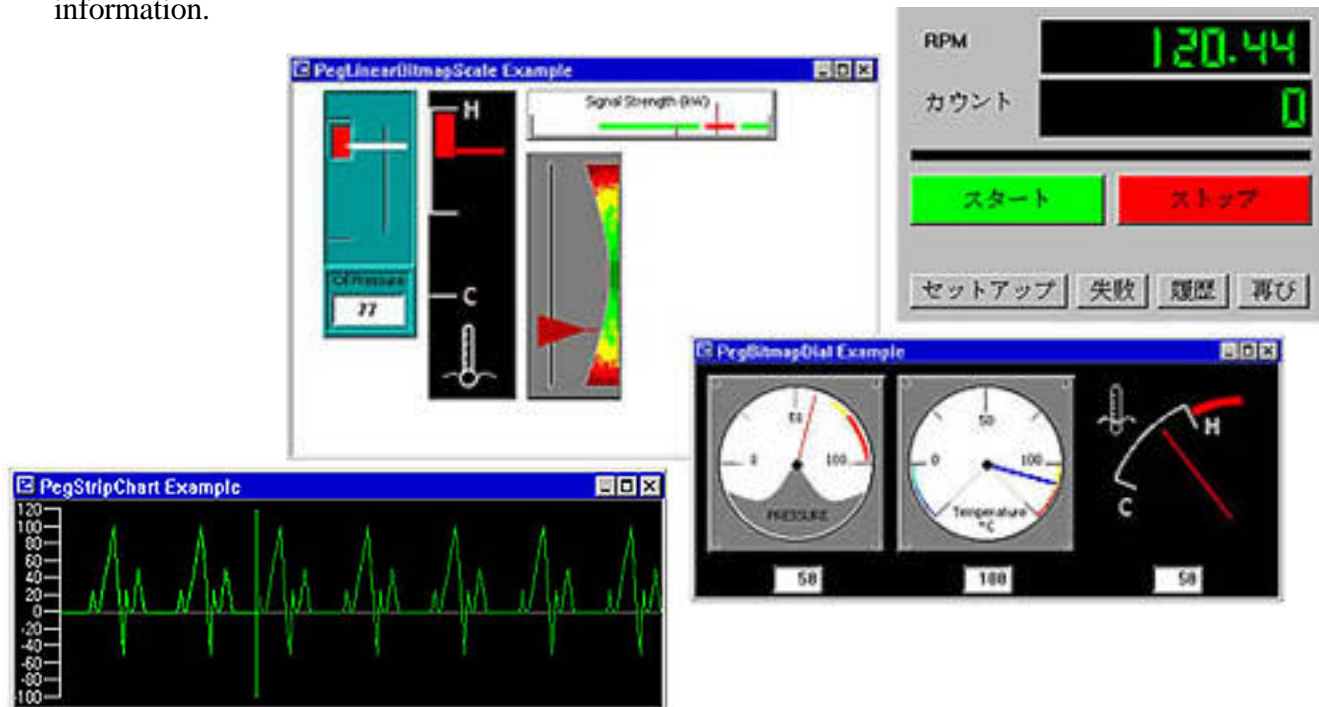


If you want to create a different look and feel, PEG can be enhanced or simplified to suit your needs. PEG makes it easy to design a unique user interface and to distinguish your product from the competition.

Sample Displays

Below are a few sample displays made with PEG.

Note the Japanese button labels on the RPM screen display below. Using PEG's industry-leading string table editor, it is easy to work with any language using only a mouse or Latin (ASCII) keyboard. Two-byte character sets, JIS and UNICODE encoding are supported. See multi-lingual support for more information.



To see PEG on your PC, download some of PEG's demos from our website.

Fast and Small

PEG is designed specifically for embedded-system GUI development. The value of every feature of PEG is carefully weighed against its code size and performance requirements.

Small Footprint

A minimum PEG footprint occupies only 50K of code space, 4K of stack, and 2K of dynamic memory. A typical full-featured PEG GUI requires only 100K of code space, 4K of stack, and 8K of dynamic memory.

PEG's small footprint is achieved by several means:

- Heavy reliance on C++ inheritance – encourages code re-use
- Each control type is built incrementally upon its predecessor – you select only the objects needed by your application at compile time
- Configurable library – configuration flags let you completely define the input devices, drawing primitives, and higher level features required by your target

Maximum Performance

PEG achieves maximum performance by minimizing the system overhead required to maintain a graphical presentation.

- Interacts directly with video and input hardware – guarantees greatest possible throughput
- Supports video controllers with hardware acceleration and double buffering – provides flicker-free animation and scrolling
- Uses advanced window and viewport clipping techniques – prevents unnecessary screen drawing

Multi-Lingual Support

PEG now provides *industry-leading* support for multi-lingual application development. This gives you unequalled range and flexibility in designing products for any of the world's markets.

- Support for two-byte character sets – includes "C" string library functions
- Use very large character sets in memory-limited systems
- Multi-lingual support – mix Latin, Cyrillic, Han, Katakana, Hiragana etc. in a single PEG application
- String table editor provided – easy to enter string with mouse or keyboard
- JIS and UNICODE formats supported

Use Large Character Sets in Memory-Limited Systems

Our new *CompositeFont* technology provides you with a solution for incorporating very large character sets into memory-limited embedded systems. In a single PEG application, virtually any language can be used in any combination of character sets. These include Latin, Cyrillic, Han, Katakana, Hiragana, etc.

Build String Table with Keyboard or Mouse

PEG's new *String Table Editor* lets you enter a string in any supported language using only a mouse or Latin (ASCII) keyboard. The editor maps each string to the character sets you have specified and generates a "C++" source file. JIS and UNICODE data entry formats are also supported.

Includes "C" String Library Functions for Two-Byte Characters

The PEG library provides a full complement of compiler independent "C" string library functions. This eliminates the need for special non-ANSI compiler support for two-byte characters.

Cut Development Time

Reduce Development Time

PEG's development tools run on Windows and X11 development platforms. Use your PC to save development time by avoiding unstable target hardware. Begin designing your PEG GUI even before the hardware is ready. When it is ready, just recompile with your target's I/O interface classes. WindowBuilder is great for testing design ideas and creating prototypes for management.

- Begin GUI design, testing, and debugging on your PC
- Start development before target hardware is ready – avoid unstable hardware
- Design GUI in a standard PC environment – use familiar tools
- Create quick GUI prototypes for management – get a raise

What You See Is What You Get

The GUI objects you create with WindowBuilder will appear exactly the same on your target as they do on your PC. WindowBuilder is actually a PEG application program running in a Microsoft Windows or X11 Windowing System development environment. This guarantees that WindowBuilder is completely WYSIWYG.

Familiar Event-Driven Programming Model

If you have experience writing applications for Windows or other GUI environments, you will find PEG's event-driven programming model is familiar ground. You can get your PEG GUI up and running in less time and with minimum hassle.

Model Get Some PEG Training

Take advantage of our training services and get your application running in record time. We offer customer training at our corporate headquarters or your location. To make arrangements for a training session contact sales@swellsoftware.com.

Highly Portable

We made the PEG library highly portable by abstracting all hardware-dependent functionality. The vast majority of the library is completely hardware independent, relying on well-defined hardware interface objects to provide a consistent, simple, and reliable set of I/O methods.

- Hardware independent
- Proven portability – PEG applications already run on major embedded CPUs and RTOSs
- Makes no assumptions about software environment – provided examples make it easy to create new interface classes
- Conforms to EC++ standard – verified with many embedded CPU compilers
- Screen driver templates provided – support external and CPU-embedded controllers

Proven Portability

PEG applications are running today on the major embedded processors. Example applications are provided that work with most of the common display and input devices.

Easily Integrated with RTOS

PEG makes no assumptions about the software environment. OS-specific dependencies are encapsulated and well documented. You can easily create new interface classes for any environment using the examples that are provided. PEG has been integrated with a variety of RTOSs. Many of our customers have also integrated PEG with their RTOS. Complete RTOS porting instructions are provided.

Conforms to EC++ Standard

PEG's library conforms to the EC++ standard. It does not require support for C++ exception handling or run-time type identification. PEG is fully verified with a large set of embedded CPU compilers.

Screen Driver Templates Make Porting Easy

For new I/O devices and video controllers, we provide screen driver templates to help you create your own drivers. Using an I/O device template, you only need to write three functions; `GetVideoAddress()`, `SetupPalette()`, and `ConfigureController()`. Video controller templates support both external and CPU-embedded controllers including those with hardware acceleration capabilities.

Development Tools

PEG is the only embedded GUI package on the market that offers a complete set of development tools that are designed to save you time and effort. `WindowBuilder`, `FontCapture`, and `ImageConvert`, written using PEG, will help you create eye-catching, intuitive, full-featured, and multi-lingual embedded GUIs using your Windows and Linux based development platform.

- Design PEG windows and dialogs using `WindowBuilder`
- Create multi-lingual applications with `WindowBuilder`
- Generate your own embedded fonts using `FontCapture`
- Convert bitmap images into ROMable format with `ImageConvert`
- Quickly build GUI prototypes using Windows or Linux based PC
- Automatically generate "C++" source code

Development Environments

In the early stages of the development cycle it can be difficult or impossible to do application development on the intended target platform. Either the hardware is unstable or unavailable.

Microsoft Windows

We have solved this problem by making it possible for you to run your PEG GUI as a standard Windows 32-bit or X11 application. Now you can create, test, and debug your entire user interface using familiar and mature Windows development tools on your PC. This lets you get a head start on your project and reduce development time.

MS-DOS

PEG also comes complete with interface classes for running PEG as a DOS real-mode, 16-bit protected-mode, or 32-bit flat-bit mode application program. This is very helpful if your target is a member of the x86 family.

X11 Windowing System

The LinuxPEG package comes with a version of WindowBuilder that runs on the X11 development environment. This allows you to code, test, and debug your application while running it on the X11 Windowing System. If you are running X11 on top of the same OS that will be used on your target, you can get a clear representation of your final application and have access to all of your desktop tools.

Porting to Target

Porting to your final target is simply a matter of recompiling the PEG library with I/O interface classes designed for your target. Your application level GUI software is guaranteed to run without modification on your target platform.

Multi-Tasking Models

PEG can be configured to support several tasking models to meet the needs of your embedded system. This flexibility makes PEG an ideal solution for a wide range of applications.

- PEG as a single low-priority task – real-time performance is not effected
- PEG supporting multiple GUI tasks
 - Internal data structures are protected
 - Application level programming is greatly simplified
 - Any GUI task can display a window or GUI object and update display data
- PEG running standalone – perfect for small and simple applications

Single Low-Priority Task Model

In this model, PEG is configured to execute as a single low-priority task in your overall multi-tasking system. The user-interface software interacts with I/O devices and other system tasks via messaging methods supported by the underlying OS kernel. Your user-interface software is insulated from the real-time tasks executing on the target CPU. This guarantees that PEG will not effect real-time performance of your target system.

Multiple GUI Task Model

PEG may also 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 judicious use of semaphores that are provided by the underlying OS kernel. Application programming is greatly simplified.

Standalone Model

PEG can also run standalone without a multitasking kernel. This model is perfect for smaller, less complex applications.

Flexible and Powerful API

The PEG library provides an intuitive and robust object hierarchy. You can use PEG objects as provided or enhance them to suit your needs. PEG imposes no artificial limits on how objects are used so you are free to design screens with an indefinite nesting level of controls within windows within other windows.

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.

- Enhance PEG objects to suit your needs.
- Indefinite nesting level of controls.
- API is defined entirely by public functions.
- Default appearance of PEG objects is similar to common desktop graphical environments – can be enhanced or simplified.
- Rich set of GUI object types.

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.

Target Processors

PEG is designed to be highly portable and work with any embedded processor. PEG applications are running on all of the major processor families including those listed below. For a discussion of recommended target environments see Requirements.

Supported Target Processors and Reference Platforms

Processor Vendor	Processor Name	Reference Platform Vendor
AMD	Elan x86	AMD ElanSC400
Cirrus Logic	ARM	Cirrus Logic EP7212
Infineon	C167	
Intel	StrongARM x86	Intel StrongARM Assabet ZF MicroSystems 486 netDisplay
MIPS		
Motorola	ColdFire DragonBall 68332 68340 MPC821 MPC823 MPC860	Arnewsh ColdFire 5206 Motorola MPC823 FADS Embedded Planet MPC823 Blue Planet
Samsung	ARM	Samsung ARM S3C44A0X
Sharp	ARM	Sharp LH77790 Sharp LH79531

Input Devices

PEG includes complete support for all standard input devices.

- Interacts directly with input hardware – maximizes throughput
- Complete control of functionality using library configuration flags – reduces code size
- Mouse input and touch screen drivers are provided for supported reference platforms
- Full set of GUI object types is included

Input Devices

Mouse Joystick	<ul style="list-style-type: none">• Input drivers provided for reference platforms• Drawing pointer bitmaps and hardware cursors are supported• Pointer input functionality can be removed from library – reduces code size
Touch screen	<ul style="list-style-type: none">• Input drivers provided for reference platforms• Mouse pointer drawing can be eliminated – reduces code size and drawing overhead• “Pointer move” messages not required – simplifies driver development• Highlighting of objects with "Input focus" can be turned off – reduces code size
Keyboard Keypad	<ul style="list-style-type: none">• Supports navigation via menus, windows, and dialogs• Everything from full QWERTY keyboard to user-defined membrane keys is supported• Full navigation can be accomplished using only 3 input keys• Keyboard functionality can be removed from library – reduces code size
Soft Keys	<ul style="list-style-type: none">• Supports "membrane keys" placed at perimeter of display screen

Video Output

Overview

PEG is designed to take advantage of a full range of video output devices and display screens. You can use video controllers that have hardware acceleration capabilities and double-buffered video output. A full set screen driver templates is included to help you get started.

PEG supports:

- Monochrome
- 4 and 16 grays
- 16, 256, and 65535 colors
- True 24-bit RGB color
- Video and LCD displays of any resolution or orientation

Hardware Acceleration

PEG takes full advantage of video controllers that support hardware acceleration capabilities such as hardware cursor or hardware bit-blit. Software emulation is provided when the target controller does not support a specific feature in hardware.

Flicker-free Animation and Scrolling

PEG can be configured to support double-buffered video output. All intermediate drawing operations are performed in an off-screen or local memory buffer. At the conclusion of the drawing operation, the invalidated region of the local buffer is transferred to visible video memory, using hardware bit-blitting if provided. This mode of operation gives your application flicker-free animation and scrolling. You can also configure PEG to draw directly to visible video memory.

RTOS Support

PEG is designed to work with any commercial-quality RTOS. Our customers have also integrated PEG with their in-house and other commercial RTOSs.

True real-time multi-tasking GUI environment

PEG the only true real-time multi-tasking GUI environment available on the market today. PEG is fully integrated with RTOS messaging, memory management, and synchronization services. This assures you the lowest possible overhead.

Compiler Support

PEG is designed to work with any C++ compiler/debugger combination. The PEG Library has been fully verified with each of the compilers listed below.

Supported Compilers

Compiler	Processor
ARM ADS Ver. 1.0	ARM
Borland C++ Ver. 4.52, 5.0	x86 real-mode x86 protected-mode
CAD-UL C++	x86 32-bit
Diab C/C++	ColdFire PowerPC
GNU C++	All Unix/Linux environments
Green Hills C++	ARM StrongARM PowerPC SH3
MetaWare High C/C++	ARM x86 32-bit
Metrowerks CodeWarrior	68K ColdFire MCORE PowerPC
Microsoft MSVC++ Ver. 1.52, 5.0, 6.0	x86 real-mode x86 protected-mode
Sybase (Watcom) C++	x86 real-mode x86 32-bit protected-mode
Tasking C++	C167

Control Types

PEG's rich and growing compliment of GUI object types are listed below:

- PegAnimationWindow
- PegBitmap
- PegBitmapButton
- PegBitmapConvert
- PegBitmapLight
- PegButton
- PegCheckBox
- PegColorBitmapLight
- PegColorLight
- PegComboBox
- PegDecoratedWindow
- PegDial
- PegDialog
- PegEditBox
- PegFiniteDial
- PegFiniteBitmapDial
- PegFont
- PegGroup
- PegGifConvert
- PegHorizontalScrollBar
- PegHorizontalList
- PegIcon
- PegImageConvert
- PegJpgConvert
- PegLight
- PegLinearScale
- PegLinearBitmapScale
- PegLineChart
- PegList
- PegMaximizeButton
- PegMenu
- PegMenuBar
- PegMenuButton
- PegMessageWindow
- PegMinimizeButton
- PegMLTButton
- PegMultiLineChart
- PegNotebook
- PegPointer
- PegProgressBar
- PegProgressWindow
- PegPrompt
- PegRadioButton
- PegRect
- PegScale
- PegSlider
- PegSpinButton
- PegSpreadSheet
- PegStatusBar
- PegString
- PegStripChart
- PegSystemButton
- PegTable
- PegTerminalWindow
- PegTextBox
- PegTextButton
- PegThing
- PegTitle
- PegToolbar
- PegToolbarpanel
- PegTreeNode
- PegTreeView
- PegVerticalList
- PegVerticalPrompt
- PegVerticalScrollBar
- PegWindow

Drawing Primitives

PEG applications rely primarily on PEG class library objects for window and control drawing. However it is often useful to perform custom drawing at the application level. All PEG drawing primitives can be invoked at any time by your application. They are listed below:

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, any outline width.
Rectangle	Outline and/or 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.

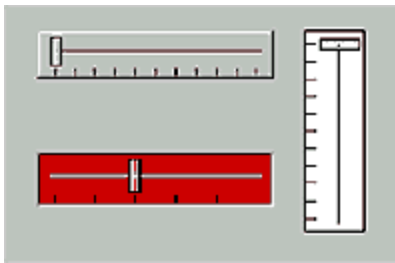
Code Example

PEG The following function creates the PegSlider shown below. The slider's attributes include:

- Vertical slider
- Minimum value of 0
- Maximum value of 200
- Tick mark interval of 20 (10 tick marks will be drawn)
- Initial slider value of 50

Code Example

```
void MyWindow::AddSlider(void)
{
    PegRect SliderRect;
    SliderRect.Set(20,20,60,120);
    PegSlider *pSlider=new PegSlider(SliderRect,0,200,ID_SLIDER,20);
    pSlider->SetCurrentValue(50);
    Add(pSlider);
}
```



PegSlider

Requirements

Tool Chain and Processor

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 available if you need help using PEG with a specific toolset.

Multi-Tasking Environment

The target's real-time operating system must support a means for inter-task communication via messages. A periodic timer interrupt service is also needed to support PEG's high-level timer services. When PEG is configured to support multiple GUI tasks running at different priorities, the OS must support a means for semaphore protection of critical code sections. These requirements are trivial commercial-quality RTOS packages.

Recommended Processor Type and Configuration

There are no internal restrictions on target CPU type or hardware configuration. For adequate performance we recommend a 16-bit or better CPU running at 16 MHz or higher. For applications needing very high resolution or color depth, the CPU performance will probably need to be increased to achieve a responsive graphical display.

Target Applications

PEG is being used on a variety of embedded applications ranging from medical instrumentation to wireless devices. Here is a list of some of the more interesting applications that our customers have told us about.

- Digital Cameras
- MP3 Players
- Industrial Controls
- Medical Devices
- GPS Instrumentation
- Handheld Computers
- Military / F.L.I.R. Applications
- Printers
- Internet Appliances
- Set-top Boxes
- Wireless Devices
- Web Browsers

Why is PEG the Right Choice for You?

- Royalty-free – priced right for high-volume applications
- Designed specifically for embedded systems – not a Windows or UNIX retrofit
- Includes a complete set of development tools
- Provides *industry leading* multi-lingual support
- *New CompositeFont* technology incorporates very large character sets into memory-limited applications
- *New String Table Editor* supports JIS and UNICODE
- Supports all types of display devices
- Customizable look and feel – make your application stand out from the competition
- Reduces your development time
- Supports popular RTOSs, target processors, video controllers, and input devices
- Used in all types of embedded applications
- Excellent technical support and customer training

Use only the best GUI library and development tools on your next project. Join the growing list of software engineers that are using PEG to create some of the most exciting embedded applications on the market.