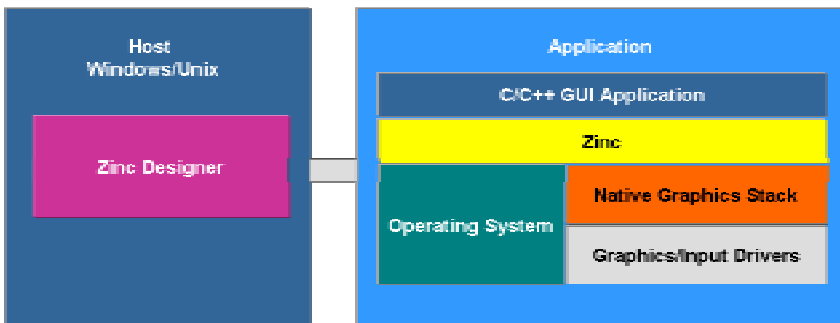


Zinc™ - platform independent GUI library

Zinc™

Once the domain of PC users only, graphical user interfaces (GUIs) are increasingly demanded by embedded device users as well. However, the unique and varied requirements of embedded devices have often made available GUI solutions unsuitable. These devices are characterized by their diversity: they are usually developed with specialized hardware under cost-constraints and used for a dedicated purpose. Therefore, a GUI solution for embedded devices has to address the special requirements unavailable with traditional, desktop GUI solutions.

Zinc™ from PSA fulfills the graphical requirements of embedded devices making it possible for a developer to build a rich, full-scale embeddable GUI using real-time operating systems with low system overhead and rapid time to market.



Scalability and configurability

Almost every embedded design is unique, so an off-the-shelf software solution needs to be flexible in order to be adapted to the unique visual branding for the device, and must include just those components required for a specific application. Zinc provides an application developer with a variety of methods for providing a customized “look” and “feel” for their specialized device. Zinc also provides this scalability through its object-oriented design and its scalable architecture.

Resource efficient

Desktop GUI solutions are developed for systems with almost infinite memory that run on high-end processors. An embedded GUI solution must fit into systems with small memory footprints that run on processors with widely different performance criteria. Zinc was designed to run in low memory environments and its minimal footprint is roughly 350 KB.

Product Features

- Scalable starting from 350KB
- Easy porting to custom hardware
- High-performance graphical output
- Highly customizable “look” and “feel”
- Intuitive, complete C++ API
- Sophisticated event routing and model/view architectures
- Powerful visual design tool/GUI builder
- Full internationalization support
- Multithreading support

Internationalization

- Embedded devices must be able to use the same fundamental design in different countries, and the challenge is to have a GUI component with the same flexibility. Zinc provides integrated support for localized applications so that all standard UI texts are automatically translated and all date and number formats are changed. Zinc can be configured to use UNICODE 16-bit wide characters, which allows a seamless integration of non-western fonts, such as Asian fonts.

User interface objects

- Window, dialog window, scrolled window, MDI window, child window, message window
- Horizontal and vertical splitters, group, scroll bar
- Toolbar, pull-down menu, pop-up menu
- Button, radio buttons, check box
- Vertical list, horizontal list, combo box, spin control
- Table, tree list, notebook
- Bitmap, image, icon, progress bar
- String, formatted string, text
- Integer, real, big number
- Date, time
- Status bar, prompt, generic
- File, print, help, and error dialogs

Custom Zinc Development Services

Need specific customization of Zinc or project development assistance? PSA offers domestic and international development services. Contact PSA for more information.

Purchasing Information

Email: sales@psa-software.com
 Tel: 1.810.724.5200
 Fax: 1.810.724.5500
 Web: www.psa-software.com

Zinc™ - platform independent GUI library

Differentiable – Customized look and feel

Because most embedded devices serve a dedicated purpose, they also require customized user interfaces not a static, predefined look and feel that does not address unique GUI requirements. The appearance of Zinc objects can be customized, as can the manner in which they handle events, in an easy and intuitive way. The Zinc designer, a GUI builder for rapid prototyping, allows the integration of customized objects. These can then be used in the Zinc designer to quickly prototype the new look and feel of the GUI in a graphical way. For example JPEG images can be used to customize the appearance of a button.



Easy to use

Embedded application development is driven by strict deadlines and GUI solutions must address the same time-to-market issues as any other software component. Zinc was designed using object-oriented methodologies with derivation of classes that make it easy to understand. The Zinc designer helps with building applications graphical on the development host even before target hardware is available.

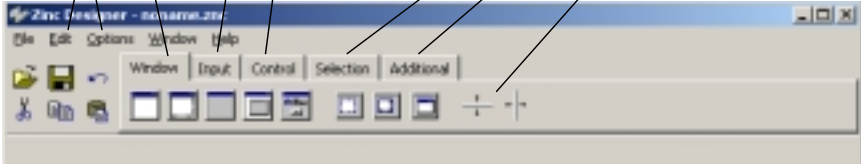
Optimized for multithreaded environment

Embedded systems often have no MMU and therefore do not support separate memory spaces and complete process models. Instead they provide only a multithreaded environment. Zinc is optimized for a multithreaded environment.

A complete GUI solution

Zinc provides a complete object-oriented, C++ application program interface (API) for the creation of GUIs and event-driven applications. Zinc is composed of GUI libraries, a visual design tool, hypertext-based on-line documentation, and numerous examples and tutorials. Zinc can easily be scaled and configured to meet the exact GUI requirements of a given application. Designed for memory constrained environments, Zinc can fit a complete configuration consisting of VxWorks and Zinc libraries in less than 1MB of memory. Rounding out the Zinc feature set are portable file system support, help and error systems, and sophisticated model/view and event-routing architectures.

Zinc Designer



- Create a variety of main and child window types: scrolled, dialog, message, and MDI parent windows
- Generate code in a click
- Quickly modify geometry management and tab order. Add in Help Context, Language, and Locales
- Add control objects: button, radio button, check box, group object, pull-down menu, toolbar, and notebook
- Easily input objects: string, formatted string, number, date, and time
- Create selection objects: vertical and horizontal lists, combo boxes, spin controls, tree, and table objects
- Add additional objects: prompt, bitmap, image, icon, status bar, progress bar, scroll bar, and chart
- Create horizontal and vertical splitters