



# Creating Next-Generation Display Applications for Real-Time Embedded Systems

By Yannick Lefebvre

## ABSTRACT

The VAPS XT software suite from Presagis is a powerful environment that can be used

- to design a wide number of display applications, including both aerospace and automotive screens,
- to automatically generate code, and
- to deploy executables to a wide variety of real-time hardware platforms.

By using a streamlined process to execute all of these tasks, the user can spend more time developing tailored content and less time working to get it running on the target hardware. This paper will describe the VAPS XT application creation workflow from beginning to end and will then focus on the process of creating a custom set of runtime libraries to deploy the code on a number of target systems.

## VAPS XT OVERVIEW

VAPS XT is an object-oriented model-based development tool that enables users to create the visual appearance, logic, and data connectivity of their displays in a user-friendly environment.

In the past decade, there has been a steady growth in the acceptance and use of model-based tools to create Human-Machine interfaces (HMIs). However, some projects still use hand-coding to produce the code driving their products and to present information to the end-user. In this era of increasing productivity expectations and shorter deadlines, these methods are not always able to produce the desired results. Hand-coding also produces results that are difficult to maintain over a vehicle's lifespan and hard to update regularly during the initial definition of a display.



Figure 1. Primary Flight Display created with VAPS XT

With its graphical editor and extensible architecture, VAPS XT allows some users to focus on the visual appearance of their displays while more advanced programmers can augment the tool as necessary by creating new visual components as well as interfaces to external data sources and hardware components. In addition, the work created can easily be re-used across multiple projects through simple drag-and-drop operations. When modifications to a display are required down the line, all of the design artifacts can be loaded back into the editor in order to easily create a new revision of the code.

With its automatic code generation capabilities and its ability to support a wide variety of software and hardware environments, VAPS XT accelerates the turnaround time from making modifications to a design to being able to see it running on the target hardware. It also simplifies both the process of supporting multiple hardware platforms simultaneously and the migration task of switching to a new hardware supplier in the middle of a project. Finally, it shifts the focus from being very close to the code and having to develop a complete runtime infrastructure to the content of the displays.

For customers who are already using VAPS to create displays, VAPS XT builds on the strengths of VAPS. VAPS XT addresses some of the limitations of its predecessor by offering more flexibility and a cleaner interface to speed up the development process. It also provides a migration path for importing legacy work into the new environment, either to provide a starting point for new projects or to be enhanced by the new building blocks and rendering techniques in VAPS XT. Customers who have been deploying their code to embedded targets using CCG Lite or QCG will feel right at home in the new software suite and will be able to continue enjoying the speed of compilation and deployment of executables to real-time systems.

## THE VAPS XT APPLICATION CREATION PROCESS

The first few steps of creating a VAPS XT application are performed in the VAPS XT Editor, a Microsoft Windows-based environment that offers a variety of drawing tools and facilities to define a display's logic and behavior. The remaining steps include focusing around the code that is generated by the tool and then compiling it against a set of runtime libraries to adapt it to a target environment.

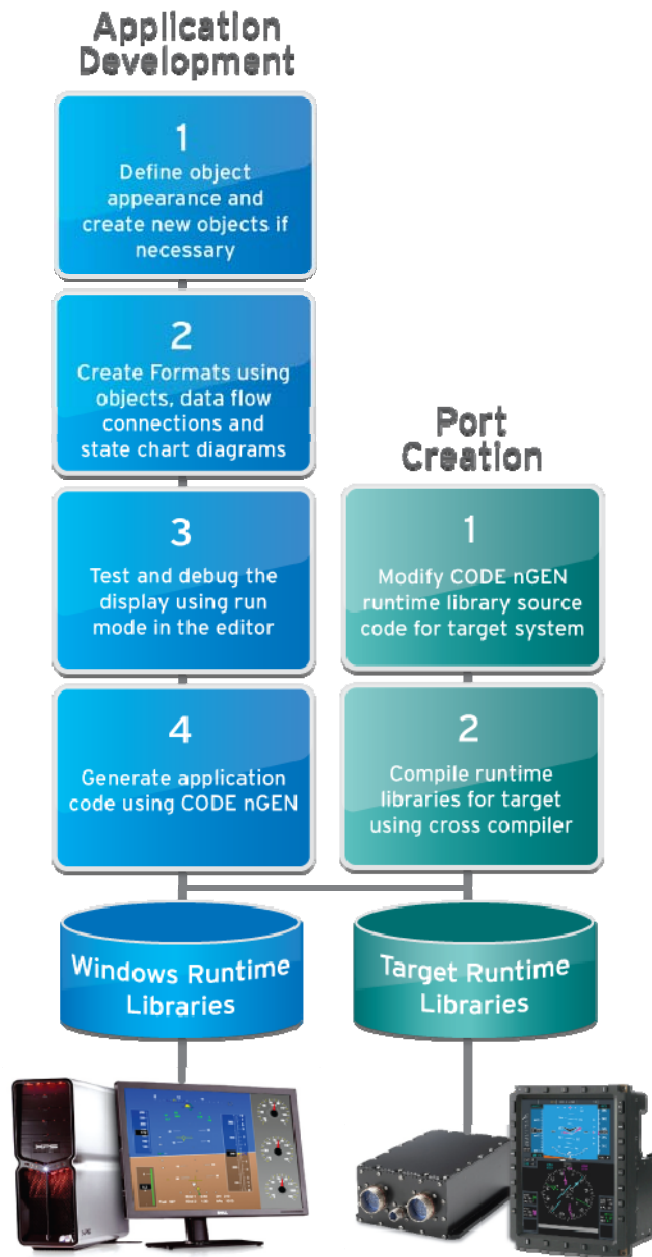


Figure 2. The VAPS XT Workflow



In this first step, the user will either (1) create new objects inside the development environment based on their display needs or (2) modify the look and feel of existing objects in order to achieve the desired visual appearance.

VAPS XT provides multiple objects, such as circle, ellipses, dials, text readouts, sliders, and windows. These building blocks range in levels of complexity from static graphical elements to dynamic objects that react to incoming data and interactive components that the user can activate. It is also possible to import high-resolution images in BMP or PNG format into the tool and to use these images in dynamic objects. While the first one is a Windows standard, PNG files offer a high compression ratio and per-pixel transparency definition. The various parts of a display (background, needle, moving parts) should be broken up into multiple pieces before using them in the editor.

Any object provided with VAPS XT can be used as is, or it can become the basis for the user's own custom component. When making modifications, the editor provides a large number of graphical building blocks and components to choose from. VAPS XT features gradient fills and such high-quality graphical elements as transparency right in the editor. It also contains a large object library of more than 300 photo-realistic objects that can be used as is or that can become the basis for new custom elements.

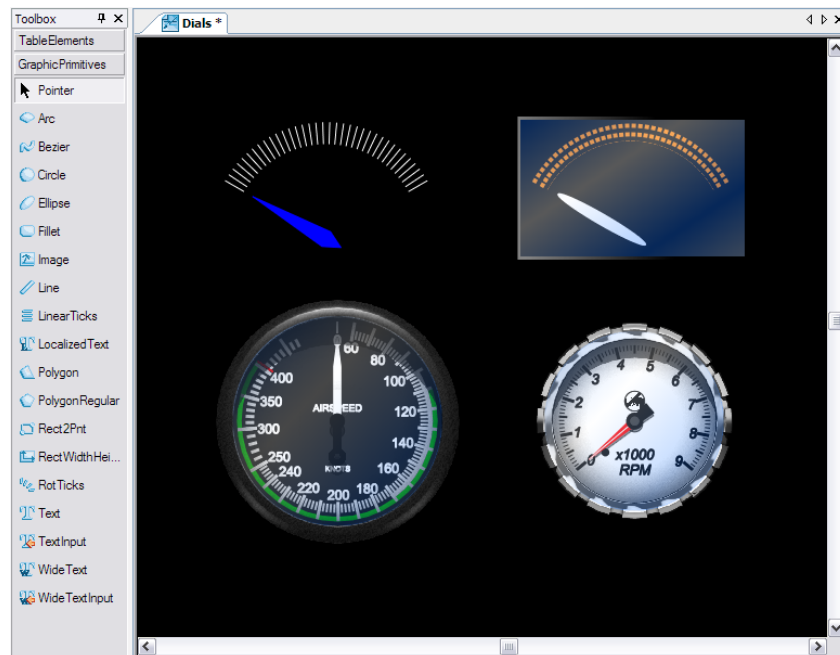
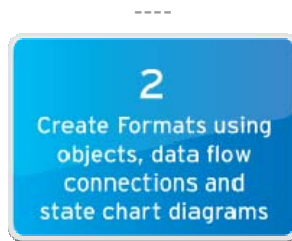


Figure 3. Basic and complex dials created with VAPS XT

In cases where a specific type of functionality is not represented within the built-in object library in VAPS XT, the open environment lets users create their own objects by combining or extending standard building blocks through the graphical editor or by writing C++ object-

oriented code. For example, a user could create either a custom object to receive data from an external data source or a new type of pick-list component.

When working in the editor, the functionality of the new element is created through drag-and-drop operations. For coded objects, VAPS XT automatically generates a code template that simplifies all of the laborious tasks of writing code to interface with the rest of the environment and lets the user focus on the core functionality of the object. The VAPS XT open architecture also allows users to create custom lists of properties, events, and operations for their objects. In all cases, new objects that are created have the same capabilities as the ones that are provided with the base package and can be re-used across the development of multiple applications.



### Graphics

Once all of the basic components have been put in place, the next step is to assemble these components to create a complete screen or a page within a larger system. This task is also performed in the VAPS XT editor through simple drag-and-drop operations or by entering exact coordinates where objects should be positioned to match a project's specifications. The XT design environment also contains a series of tools that facilitate the creation of displays, including align, grids, distribute, and duplicate.

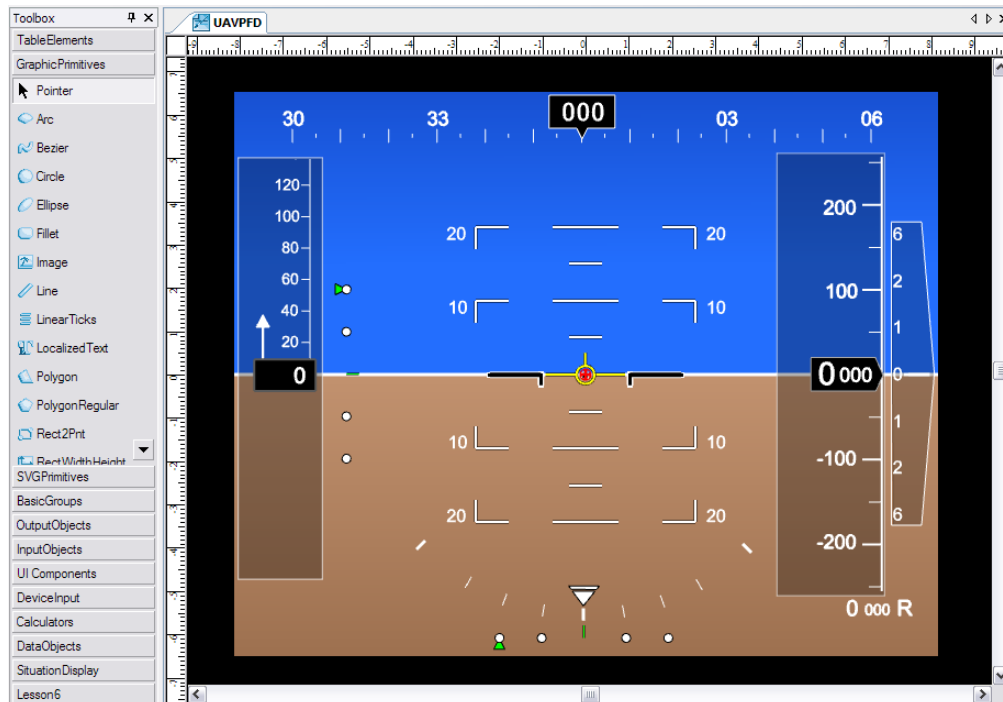


Figure 4. Reusable PFD component

## Animation and Logic

Animating displays and determining what they will display when can be accomplished by using two different techniques inside of VAPS XT: Data Flows and State Charts.

Data flows are connections between two object properties within a display. Both properties could belong to the same object or be part of different elements of the display. Some of the properties that are connected through a data flow can also be linked to external data sources, such as an aircraft simulation or a real hardware component. Once connected to a data source, display objects come to life at runtime when they react to incoming data while interactive components will send data out when the user activates them.

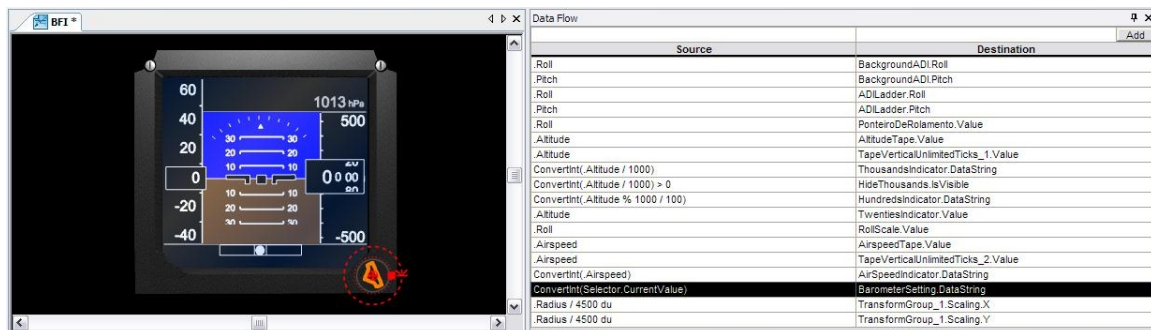


Figure 5. Data Flow connections define links between objects

When it comes to display logic, graphical state charts can be created directly within the VAPS XT development environment to handle user and external events. Based on the state-charting subset of UML, the integration of graphics and logic in the same environment makes it easy to select elements of the graphical design to create triggers and actions inside of the application logic.

With its object-oriented architecture, VAPS XT supports the creation of state chart logic both at the application level and also inside of custom objects and pages of a system. These state charts are then attached to each instance of custom objects and also keep track of their own local states. Each of these state charts can be created using either a flat design or hierarchical states for more complex logic and a cleaner overall organization.

Finally, having the graphics and logic in the same tool allows the user to associate graphics to the logic in order to control their visibility. These links appear directly in the state chart diagram, providing immediate visual feedback and simplifying logic verification.

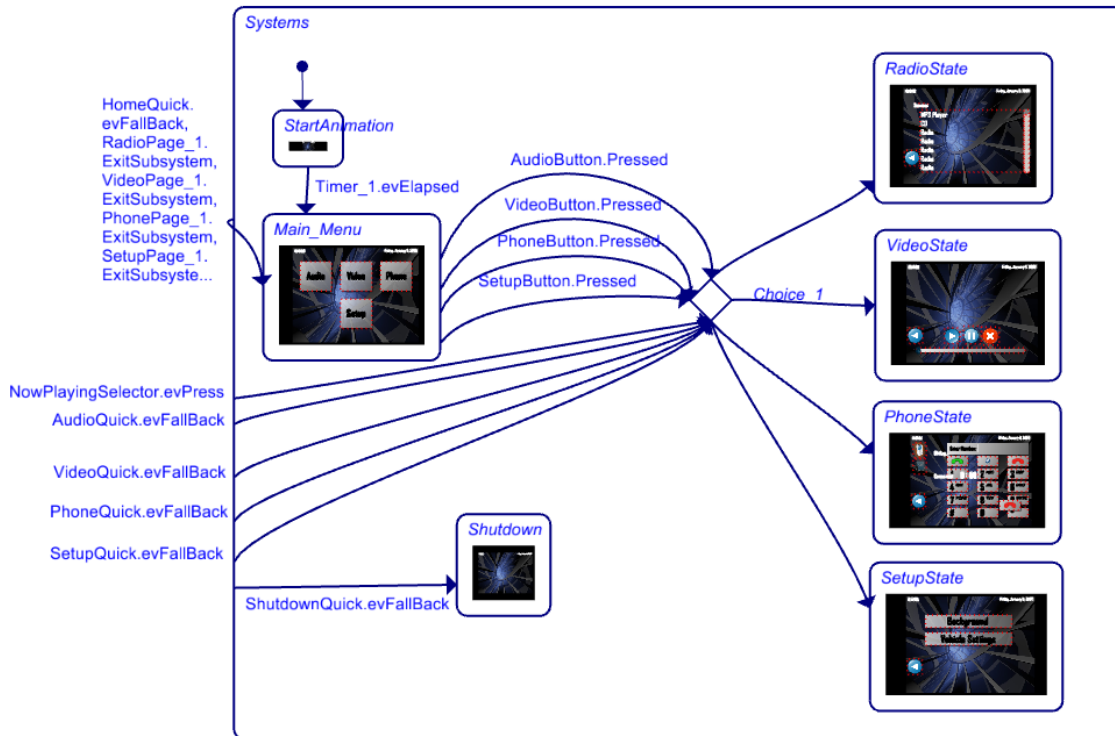


Figure 6. State Chart Diagram used to define display logic and event handling

---

## 3

Test and debug the display using run mode in the editor

VAPS XT has an integrated run mode where the application can be tested as a whole or in parts. Since it is directly part of the editor, the run mode can animate the display based on the current project data without needing to save files. Changes can then be saved once the desired functionality has been validated.

Another advantage of having an integrated test mode is that the properties of objects within the application can easily be inspected using a Watch Window while the state chart can provide debugging visual runtime trace information. Finally, the ability to record and playback user interactions with a display greatly helps in automatic test suites.





The true power of VAPS XT resides in its ability to create stand-alone executables from the display design once the user has finished creating the displays. This task is accomplished using Code nGEN, the automatic C++ code-generator in VAPS XT. Code nGEN analyzes the contents of the application, generates C++ code, and compiles the results directly from the VAPS XT interface into a stand-alone executable.

This program can be executed without needing to have VAPS XT installed and can be used in a variety of environments:

- PC-based Simulator
- Internal review or send to customer for review
- Application testing

Executables can also be launched with a variety of options to give them a specific position and size on the screen, to change the scaling factor of the graphical elements, or to remove the Windows border around them so that they take up the entire screen.



Figure 8. A running stand-alone executable

## THE PORTING PROCESS

The C++ code that is automatically generated by Code nGEN is platform-independent. It only contains the definition of the pages and objects that make up the display, as well as the logical relationships between these elements.

Calls to the graphic card and operating system to render the display are made a link with a set of runtime libraries has been established. VAPS XT comes out-of-the-box with libraries for the Windows operating system that render graphics in OpenGL. To get the code running on a different combination of hardware and operating system, customers can acquire the source code to the VAPS XT runtime libraries and customize them as necessary. This is typically referred to as the porting process.

The runtime libraries are divided into two distinct sections (see Figure 9). The first section is the runtime architecture of VAPS XT and contains internal mechanisms, including the scheduler, event handler, and update mechanisms. The lower section, the porting layer, contains all of the calls to the OS and hardware, including function calls to initialize the display, allocate memory, and initialize timers and callbacks (when available) as well as all graphic rendering commands.

By having access to the direct source code of the library, users can port VAPS XT applications to a large variety of platforms. While the default graphic rendering is done using OpenGL, the runtime libraries can be adapted to render other standard graphic libraries (such as DirectX or WinGDI on Windows) or any custom API. As for the operating system, VAPS XT can run on a variety of standard environments, including Wind River VxWorks, QNX Neutrino, and Green Hills Integrity. In addition, it can also run on a minimalist kernel that only offers basic services on devices that have restricted resources.

Once a set of libraries has been ported for a specific target, it can be used to quickly build any number of VAPS XT applications to that environment. It is also possible to create multiple sets of runtime libraries for different pieces of hardware, thus enabling an application created from a single design to be distributed to a multiple targets.

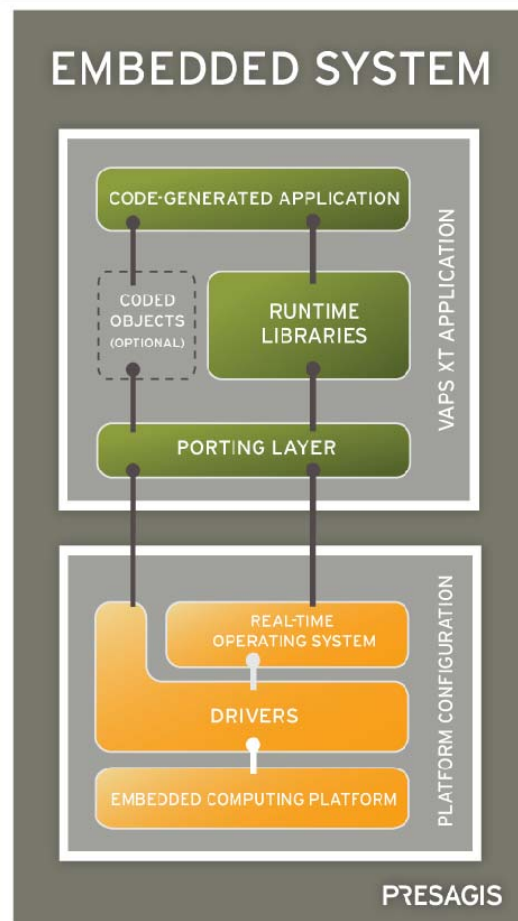


Figure 9. Embedded System Architecture

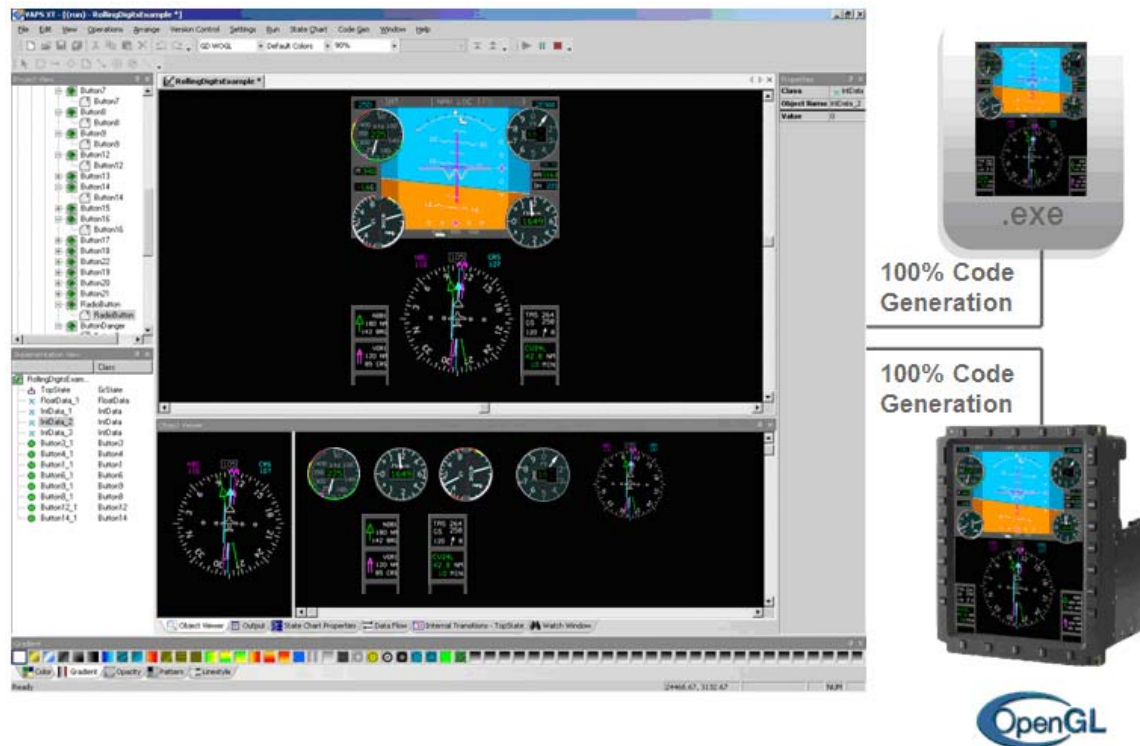


Figure 10. Automatic Code Generation Options

The porting process can be done after completing the creation of the main application or in parallel with the display development effort.

#### Supported Embedding Computing Platforms

- PowerPC® (Freescale, Motorola and IBM)
- Intel® IA32/IA64
- ARM® 11 CPUs
- And others...

#### Supported Real-time Operating Systems

- Linux
- Microsoft® Windows®XP and CE
- Sysgo® PikeOS®
- Wind River® VxWorks® and VxWorks AE653
- Green Hills® Integrity® and Integrity-178B
- LynuxWorks® LynxOS® and LynxOS-178
- And others...



All sections of code that are specific to the Windows environment or the OpenGL graphic library have been carefully identified and documented inside of the VAPS XT porting layer. With this information in hand, finding the right functions to perform the equivalent tasks for the target hardware and operating system can be done quickly and efficiently.

```

//-----+
// VXT_DRAW_VERTICES
//
// Draw a sequence of vertices in OpenGL. This
// convenient macro is re-used in different contexts
// to draw polylines, and both filled and unfilled
// polygons.
//-----+
#define VXT_DRAW_VERTICES(pts, nb_pts) \
{ \
    const sqxRCoord *_pts = (pts); \
    sqxInt _i; \
    for (_i = (nb_pts); _i > 0; _i--)\
    { \
        glVertex2fv((const GLfloat *)_pts); \
        _pts++; \
    } \
}

```

Figure 11. Example code

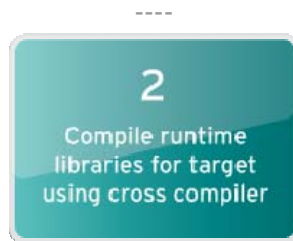
The porting process also includes the creation of a configuration file that tells VAPS XT which compiler and linker should be used to compile the runtime libraries and about any application created for that target system.

For example, Figure 12 shows configuration parameters for the linker to produce an executable for the Windows platform. A user only needs to specify the name of the linker executable, and VAPS XT will expect that executable to be in the computer's path.

```
#####  
# LINKER OPTIONS  
#####  
  
# The following line specifies the linker.  
LD=$(ECHO_ON_OFF)link.exe  
  
# Add your linker options below if necessary.  
# Include here any object files that must be linked with all applications.  
LDOPTIONS= -nologo -SUBSYSTEM:CONSOLE -MACHINE:X86 /NODEFAULTLIB:nafxcw.lib  
/NODEFAULTLIB:uafxcw.lib /delay:nobind /delayload:sqxComms.dll  
/delayload:sqxCommsHelper.dll Delayimp.lib  
  
# The linker option to specify the output directory.  
LD_DIR_OPTION= /OUT:  
  
# The linker option to specify the library search path.  
LIB_PATH_OPT= /LIBPATH:  
  
# The directory to place the application link output.  
LD_DEST_DIR=
```

Figure 12. Configuration Parameters

This configuration file is automatically added to the graphical user interface and, in order to build for the specific target, the file can then be selected in the same manner as one would select to build an executable for Windows.



Once all of the necessary changes have been made, the source code can be recompiled to produce a new set of runtime libraries. Many compilers for embedded systems provide a development environment for Windows. In these cases, the libraries can easily be compiled and used directly from the VAPS XT editor.

In cases where the compiler is on a different operating system, the source code for the runtime libraries will need to be transferred to that environment for compilation. As well, the source code produced when designing displays will also need to be sent to the platform to create the final executable.

### Re-Use and Deployment

As mentioned earlier, the new libraries created in this last step of the development and porting process can be re-used both to deploy an unlimited number of applications to a target system and to facilitate the iterative creation and test of displays on the target hardware without having to worry about costly updates to hand-written code.

## Creating Next-Generation Display Applications for Real-Time Embedded Systems

VAPS XT is the evolution of HMI software with its ease of use and rapid display creation while embracing new technologies and new levels of openness. By building on the proven concepts of VAPS, the most deployed embedded display creation tool, it gives users a complete workflow for bringing their projects to completion faster and in a much more efficient manner.

Beyond its accomplished technology upgrade and ease-of-use, VAPS XT will strive to deliver new capabilities in upcoming releases that will allow users to create displays of increasing complexity and interactivity.

## SUMMARY

With its flexible runtime architecture, VAPS XT can produce displays that will run on a variety of software and hardware environments. This approach enables display manufacturers to create rich applications using a visual development environment and also provides them with the capability to change hardware solutions at any point in the development process without having to completely redesign the application. Combined with its flexible, object-oriented approach and integrated UML state chart capability, VAPS XT is the ideal solution for the efficient development of the human-machine interfaces of tomorrow.

## About the Author

Yannick Lefebvre is the Team Leader for North America Application Engineering at Presagis. With a background in computer sciences and eleven years of experience in the training and support division, Yannick has provided counsel on hundreds of simulation and embedded display programs globally and is considered an expert in the HMI tools industry. Questions and comments can be directed to: [yannick.lefebvre@presagis.com](mailto:yannick.lefebvre@presagis.com)

## About Presagis

Presagis is a world leader in commercial-off-the-shelf (COTS) modeling, simulation, and embedded display graphics software and is the creator of the first unified COTS modeling and simulation software portfolio for the global aerospace and defense markets. As an independent, wholly-owned subsidiary of CAE, Presagis is built on a combined legacy of innovation and service, delivering superior customer value through rapid technology advancement. The company services more than 1,000 active customers worldwide, including many of the world's most respected organizations, including Boeing, Lockheed Martin, Airbus, BAE Systems, and CAE.