

**The Graphical Development Environment
for Embedded Applications Using the RTXC Quadros[™] RTOS**

Introduction

VisualRTXC is a graphical productivity tool for engineers developing embedded applications based on the RTXC Quadros Real-time Operating System (RTOS). Its ability to generate source code from the model or generate the model from the source code makes VisualRTXC the round-trip engineering tool that delivers tangible productivity improvements— whether on a clean sheet design or updating an existing application. It assists the engineer in developing each stage of an embedded application including basic design, flowcharting, coding and documentation. The representation of the application model in graphical form uses familiar concepts and elements that are widely used to describe processes, data dependencies and object relationships that are intrinsic to all embedded applications. The graphical model elements describe the application dataflow as well as lower level details, using a combination of flowcharts and textual notations.

VisualRTXC vs Traditional Methods

General purpose development tools such as text editors and compilers have long been the mainstay of development engineers. But on their own, they are often inadequate to meet the needs of highly complex embedded systems and demanding project schedules. VisualRTXC supplements traditional development tools by bringing conceptualization of the application design to a higher intuitive level. The representation of the embedded application in graphical form represents a much higher plane of development which is easier to understand by the user because of its interrelated use of dataflow diagrams and flowcharting methods.

The output of VisualRTXC is working C source code that can be used with traditional compilers, linkers and debuggers to complete the development process. This means that individuals and/or teams can quickly realize the benefits by a better understanding of the entire system, improving intra-team communications, and shortening development time.

VisualRTXC Abstractions

The VisualRTXC environment separates the application's structure and the algorithmic details of the code elements, creating three separate but interrelated abstraction levels, corresponding to logical stages in the development process. Each of these is a distinct visual space in the graphical environment.

Application Level: The highest level of abstraction, offering a general view of the program structure and data flow. Here, the user works with visual representations of kernel objects (such as tasks and semaphores) and the connections between them (RTXC Quadros kernel services) to describe how data or control flows between code entities and various objects. In this abstraction, source code for the code entities is omitted.

Code Entity Level: The middle level of abstraction, where each code entity is graphically described via common flowchart symbols. VisualRTXC automatically generates flowchart elements for kernel services between the given code entity and kernel objects on which it operates. For user-entered flowchart elements, menu options allow the user to select symbols (icons) that provide looping, conditional, control or processing constructs. Captions can be attached to each flowchart symbol so that the intent is easily understood. The user may also directly select flowchart elements and add them to the Code Diagram, entering C code as well as comments and a caption for the flowchart element.

Textual Code Level: The lowest level of abstraction that is not directly available to the user. As the user defines the flowchart symbols within a code entity, VisualRTXC simultaneously creates source code that represents the flowcharts created at the Code Entity level. The user may add non-RTXC specific code fragments using VisualRTXC's built-in text editor.

VisualRTXC Development Environment

Figure 1 shows the VisualRTXC graphical interface, featuring dockable windows and multiple level displays. The development environment is divided into eight distinct areas: Main Menu, Toolbars, Workspace Window, Data-flow Diagram, Flowchart Diagram, Source Code, Output Window, and Status Bar.

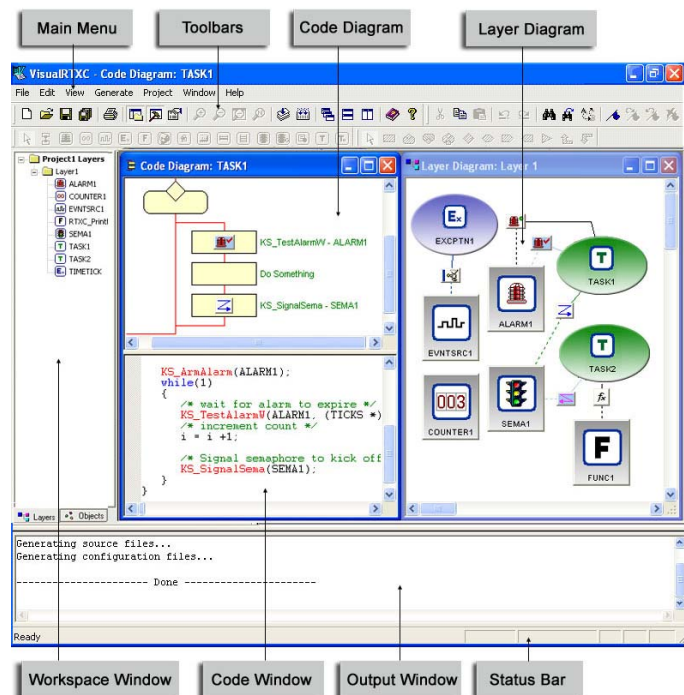


Figure 1— VisualRTXC Development Environment

Main Menu

In addition to standard Microsoft Windows menu options *File*, *Edit*, *View* and *Help*, VisualRTXC includes *Project*, *Generate* and *Window*. These three menu option provide the user with the capability to define aspects of the project,

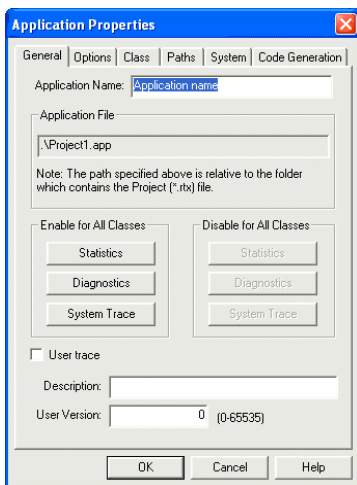


Figure 2
Project Display Window

to make configuration choices for the RTXC Quadros kernel, to generate application source code, and to select the style of the windows in the displays.

The *Project* menu option provides the same capability as RTXCgen, which is the standalone RTXC Quadros RTOS configuration utility. Figure 2 shows the primary display when configuring the RTXC Quadros kernel and Figure 3 depicts the application configuration basic display.

Toolbars

VisualRTXC offers the user a variety of tools to build embedded applications. Among them are tools for opening, saving, and printing projects, as well as help in getting assistance with features. There are also shortcut options for generating specific code elements or all code elements. There are tools for establishing favorite views by shaping the display with cascaded or tiled windows.

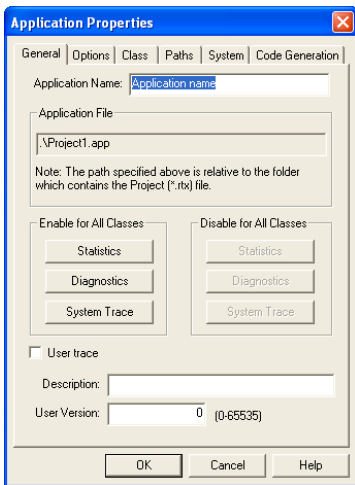


Figure 3
Application Configuration Window

In addition to the operational tools on the toolbar, there are VisualRTXC-specific tools. These tools are identified by specific icons for specifying the type of connection between RTXC Quadros code entities, kernel objects, and flowchart symbols. Each class supported by kernel services in the RTXC Quadros RTOS has an icon for its specific kernel services. The kernel service options available to the user are automatically restricted to the set that is legitimate for the class/zone of the code entity invoking the service. This feature eliminates a major source of errors often encountered with traditional development methods.

Workspace Window

The Workspace Window is composed of two views that offer efficient ways to handle applications which make use of multiple layers. Each view is a hierarchical tree showing the components in the layer(s) of the project. It is organized in a familiar hierarchical view.

Figure 4 shows the *Object View* grouping of objects by class within the project.

Figure 5 shows the *Layer View* of the Workspace Window in which the hierarchy of layers is shown and then the objects within the layer are shown alphabetically by object name.

Layer Diagram

The Layer Diagram in the Layer View is a graphical representation of the layers, showing basic structure and dataflow of the application. The user designs the layer by inserting code entity objects from the VisualRTXC toolbar along with the RTXC Quadros kernel objects they connect to and the kernel services used in those connections.

Figure 6 (following page) shows a simple layer diagram for two tasks, TASK3 and TASK4, connected to a semaphore, SEMA2. TASK4 is waiting for the semaphore to receive a signal and TASK3 signals the semaphore. Note how the Workspace Window shows the Layer View and the objects in Layer 2.

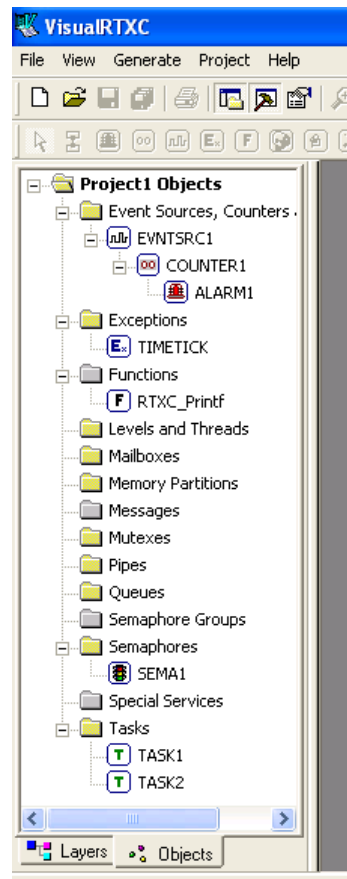


Figure 4
Workspace Window
Object View

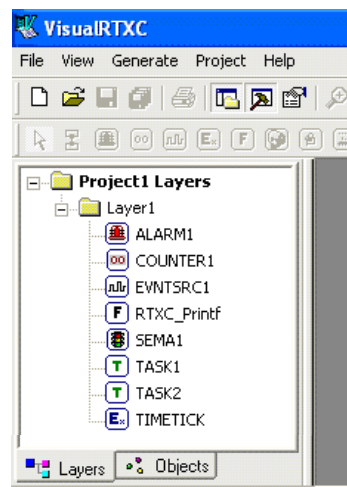


Figure 5
Workspace Window
Layer View

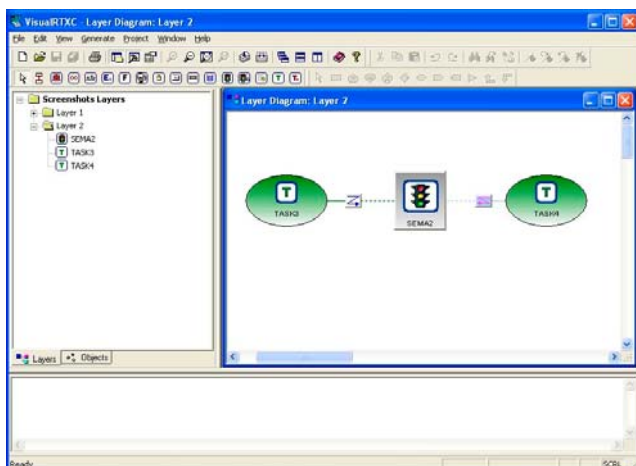


Figure 6
Simple Layer Diagram

Graphical Editor

Creation of the Layer Diagram is simple. The user merely drags and drops icons from the VisualRTXC toolbar into the Layer Diagram wherever appropriate. However, there is more to the graphical editor than just dragging and dropping. To facilitate the construction and editing of the Layer Diagram, VisualRTXC incorporates a powerful graphical editor that includes such features as:

- Zoom In, Zoom Out
- Multiple selection of objects
- Overlapping of graphical objects
- Dragging and re-sizing of layer objects using a mouse
- Automatic scrolling when objects are moved or re-sized outside of window boundaries
- Drawing of connections using oblique, vertical or horizontal lines
- Creation and removal of connections dynamically
- Icon-based separation of kernel objects and services
- Windows, toolbars and pop-up menus for selecting and editing kernel objects and associated services

Object Definition

In the Layer Diagram, the user can specify statically or dynamically created objects. Each static object will have some properties that the user will need to define. These object properties are described through windows in the same manner as RTXGen. The user may accept default property val-

Figure 7
Task Property Definition Box

ues or enter specific property values where indicated, applying them when finished. Every static object's properties definitions are handled in the same manner. Figure 7 shows the properties definition of a Task object.

Dynamically declared objects are created at runtime and their properties are declared by application code that make specific kernel service calls to apply the defined properties. VisualRTXC also supports the use of dynamic objects in almost all kernel object classes.

Code Diagram

While the Layer Diagram depicts the visual representation of the application, it does not show the logic of the code entities or temporal relationships between the various invocations of kernel services. The Layer Diagram is essentially a dataflow diagram, a prototype, and thus needs refinement to represent the application.

That refinement comes in the form of the Code Diagram for each code entity, Task, Thread or Exception, specified in the Layer Diagram. The code diagram is in the form of a flowchart that uses widely accepted and common symbols. Each flowchart symbol appropriate to the C language has a corresponding icon in the VisualRTXC toolbar. Icons exist for sequential processing, change of execution flow (*while*, *do*, *for*, *if*, *continue*, etc.) and others.

Figure 8 shows part of the Code Diagram for TASK1 in Layer 1 as shown in Figure 1.

The sequence of operations begins at the top of the diagram and proceeds downward. Indentation of the flow

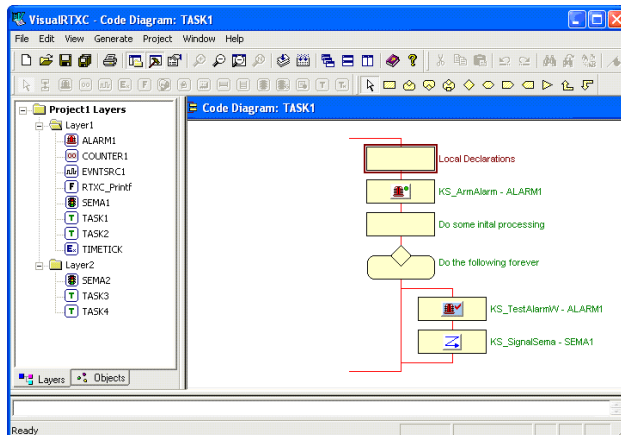


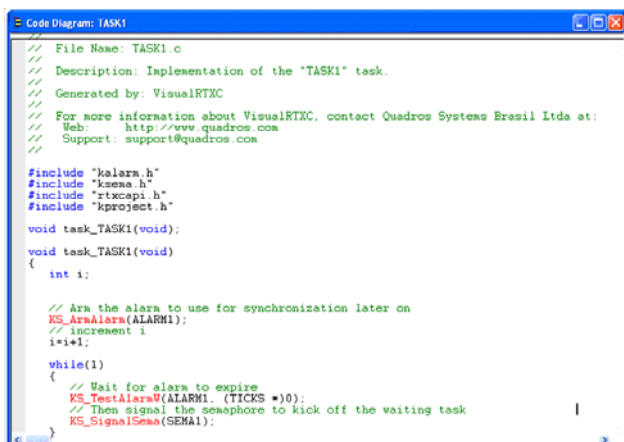
Figure 8
Code Diagram

represents command nesting. For example, the code in Figure 7 encounters a symbol for a *while* statement in which the code loops while executing two successive RTX Quadros kernel service calls.

Two empty symbols are used where there is no graphical content in the Layer Diagram. These blocks represent processing sequences in the C language that must be entered by the user, using the text editor included in VisualRTXC.

Code Window

The Code Window shows the C language code that represents the Code Diagram of the given code entity. Unlike the Layer or Code Diagrams, the Code Window is not accessible by the user. The only way to affect it is to make appropriate changes to the Code Diagram or to the Layer Diagram or to both. This restriction enforces concordance between the Layer Diagram, the Code Diagram and the actual source code. Figure 9 shows the Code Window generated for the Code Diagram in Figure 8.



```

Code Diagram: TASK1
// File Name: TASK1.c
// Description: Implementation of the "TASK1" task.
// Generated by: VisualRTXC
// For more information about VisualRTXC, contact Quadros Systems Brasil Ltda at:
// Web: http://www.quadros.com
// Support: support@quadros.com

#include "ksalarm.h"
#include "ksema.h"
#include "rtxcapi.h"
#include "kproject.h"

void task_TASK1(void);
void task_TASK1(void)
{
    int i;

    // Arm the alarm to use for synchronization later on
    KS_ArmAlarm(ALARM1);
    // increment i
    i=i+1;
    while(1)
    {
        // Wait for alarm to expire
        KS_TestAlarm(ALARM1, (TICKS *)0);
        // Then signal the semaphore to kick off the waiting task
        KS_SignalSema(SEM1);
    }
}

```

Figure 9
Code Window

Note that VisualRTXC adds the necessary statements to include the C header files (.h) for the object classes used by the task. During the actual code generation operation, those C header files will be output and will contain the handles of the objects used in the Layer Diagrams. Also note that comment styles are variable. The user can choose between standard C comment style (*/* comment */*) or C++ style (*//comment*) as shown in Figure 9.

Output Window

The Output Window serves as a display for messages from VisualRTXC to the user such as errors, status of on-going operations, and source file generation.

Help Window

VisualRTXC offers extensive Help assistance to the user. The integral help files are presented in fully searchable form. The familiar interface allows the user to view by Contents, Index, Favorites or full keyword Search. From basic "getting started" assistance, as shown in Figure 10, to more complex topics as defining the scope of a kernel object, getting the needed information is just a mouse-click away in a easily understood Help environment.

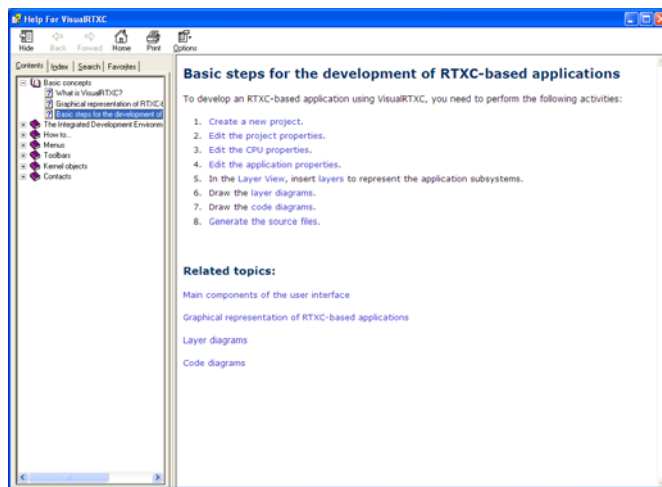


Figure 10
Help Window

Summary

VisualRTXC provides tangible productivity improvements to embedded application development by allowing the engineer to apply a high level modular approach. Whether it is a small or a large project, the application can be conceived and implemented graphically as a hierarchy of abstraction levels. This approach is universal and allows VisualRTXC to be used independent of programming tools. The user has the freedom to use VisualRTXC as a quick prototyping tool or as a complete round-trip development environment when coupled with compilers, linkers and other traditional development tools.

For more information on the VisualRTXC development tool please contact Quadros Systems, Inc. via email at info@quadros.com or on the web at www.quadros.com.



Quadros Systems, Inc.
10450 Stancliff Rd., Suite 100
Houston, TX 77099
Sales: (832) 351-2830
Toll Free: (866) 879-RTXC

www.quadros.com