Summary

The Tilcon Interface Development Suite (IDS) is a multi-platform user interface development solution (GUI/HMI Builder) that delivers robust, highly interactive user interfaces for real-time, embedded and mission critical applications. This introduction is intended to provide a high level technical overview of the Tilcon IDS. The first part of this chapter will provide a basic overview of the software. We will also detail the compiler, processor and operating systems supported. We will then provide a quick overview of how our technology works from the user's perspective followed by a more in-depth examination of the architecture of the Tilcon system and its relationship to a user application. Some of the strong points of the architecture will be reviewed as well as portability and the ability to reconfigure a Tilcon application.

The Tilcon IDS has four components namely the Tilcon Embedded Vector Engine (commonly referred to as EVE); the Tilcon Interface Builder, our main screen design tool; the Tilcon API which is the the protocol between the application and the EVE, and the Tilcon Utilities. We will take a closer look at these various components in later sections of this White Paper. We will also talk about how Tilcon facilitates the integration of design and functionality.

The Tilcon Map module is used for displaying and manipulating geographic information. It has its own raster and vector databases and viewer utilities that tie-in with Tilcon’s graphics solutions. We will discuss the features of the Map module in a later section.

The following list is a quick look at some of the key features that have made the Tilcon Interface Builder one of the leading GUI Builders of the day:

Key Features

General Features:
1. Exceptional quality of graphics
2. Easy to learn and use
3. Ease of maintenance of user interface
4. Portable - Write once and run on all supported platforms
5. Excellent prototyping tool and all screens and code fully re-usable
6. Internationalization

Embedded Vector Engine Features:
1. Runs at full native speed
2. Richly featured API
3. Identical API on all supported platforms
4. Clean separation of user interface from application code
5. No compiling needed to change look
6. Small, modular and scalable
7. Support for multiple architectures and chipsets
8. Built-in TCP/IP connectivity
9. Ability to switch language on the fly
GUI Builder Features:
1. Dynamic creation and editing of objects
2. Powerful graphics editor (design tool)
3. Thirty eight leading-edge objects
4. Twenty one draw objects - building blocks for creating custom objects
5. Effortless management of events
6. Exceptional display speed
7. Supports standard C/C++ compilers
8. Onscreen, Offscreen, and area redraws are supported
9. Custom objects
10. Object animation
11. Anti-aliasing
12. Rich 256 customizable color palette
13. bmp, .ico, .jpg, .gif and .wmf support

Tilcon Map Features:
1. Raster and Vector Databases
2. Viewer Utilities
3. Support Multiple platforms
4. Built for embedded system’s real-time performance
5. Support embedded RTOS
6. Integrates with our IDS and general graphics solutions
7. Layer and symbol overlay support
8. ADRG, CADRG, DTED formats supported

New Features in Tilcon IDS v.5.5:
1. Alpha Blending feature
2. Needle Editor for producing your own custom meter needles
3. Enhanced performance of Progress meter and other objects/platforms
4. Windows Metafile Format support
5. Redhat Enterprise Linux v.4 support

Customers and Markets
The Tilcon IDS is the ideal solution for custom GUI/HMI building, SCADA development, virtual instruments and embedded device interfaces for a variety of industries.

Creating custom graphical interfaces that include high quality virtual instruments is difficult, code intensive and requires very specialized skill sets not readily available in the market. Tilcon frees you from the burden of coding complex user interfaces and helps deliver graphically impressive applications and instrument clusters. You can quickly prototype and immediately test your custom user interfaces moving your project from concept, to prototype, to product in record time.

Tilcon has gained considerable recognition for its development tools and has built up an impressive list of Fortune 500 clients using Tilcon tools to develop their products. This
includes a wide range of industrial machinery and devices, such as plastic injection moulding machinery, semiconductor manufacturing equipment, mail sorting equipment, steel & paper mill machinery, sophisticated test benches for engine testing and many more.

Tilcon is used in a wide range of on-board vehicle systems in equipment such as aircraft, armored vehicles, locomotives, submarines and other similar applications. Tilcon has several major design wins with medical device manufacturers helping them produce very sophisticated user interfaces for devices such as anaesthesia monitoring equipment, radiation therapy equipment, dialysis machines, patient monitors etc.

With the rise of portable computing, Tilcon’s IDS is increasingly in use in a variety of handheld and other portable devices. Tilcon’s small size, scalability, speed of development and the exceptional graphics display are all important factors.

Tilcon’s powerful mapping module and the excellence of its virtual instruments is of considerable interest to automobile manufacturers who are moving in the direction of fully integrated electronic vehicle systems.

**Compiler Support**
Tilcon officially supports the following compilers:

- Microsoft Visual C++
- Microsoft Embedded Visual C++
- GNU

**Platforms Supported**
The following platforms are officially supported by Tilcon:

- VxWorks 5.5 and VxWorks 6.2
- Windows CE 5
- QNX 6.3 (Neutrino) SP2
- Windows XP
- Redhat Enterprise Linux 4

**Windowing Systems Supported**

- Photon microGUI
- WindML
- X11

**Chip Architectures Supported**
The chip architectures that are supported by Tilcon are list below:

- X-86
- StrongARM / XScale
- PowerPC
Custom support is available for architectures such as MIPS. For full details contact Tilcon Sales at infonews@tilcon.com.

How it Works

From the user's perspective, the process of user interface development with Tilcon looks like this: You use our design tools (Tilcon Interface Builder) to build screens (twd files). Drag & drop user interface objects from the toolbar. Right click to open property sheet to customize the objects.

All changes are immediately reflected in your work window. At the same time you have the opportunity of visually setting much of the functionality and event management for the object. Click on test button to see what your windows will look like at runtime.

Your C/C++ application will include a main event loop to control the user interface. Your application will start the EVE (a set of threads separate from your application threads) and opens a line of communication with it using TRT_Start. After performing some initialization commands, your application will then instruct the Engine using API commands to load and possibly display the introductory windows of your interface or create them dynamically.

Typically, the application will then go into a main loop and wait for events to occur. When a GUI event occurs (i.e. button is clicked), the event is either processed directly by Tilcon (i.e. triggers) or Tilcon passes a structure to your control application describing the event for handling by the application code or by calling a callback function.

Your control process (your application) will pass instructions (API commands) that are queued into the Engine. Most of the typical commands are asynchronous (unless asking
for information) and the control process does not have to wait for the display to be updated before returning to its task of monitoring/controlling the real-world system.

The *EVE* isolates the GUI calls from the control application. It is a separate process to allow the control application to be real-time and enhance system reliability.

Once your screens are done and control application written, you compile your application and link to the Tilcon API and are ready to run.

**Tilcon Architecture**

The Tilcon development system consists of three principal components: the Tilcon *Embedded Vector Engine (EVE)*, a platform independent API and a visual design tool, the Tilcon *Interface Builder*.

The *EVE* is a small, event-driven kernel that has the task of rendering the user's screens. It is a separate process from your application and manages all screen display and user events. Your application needs only to focus on its core real-time functionality. The separation of the user interface makes your user interface very portable, easy to modify and maintain.
The second component is a comprehensive set of platform independent API calls. These provide the communication between your application and the EVE. They support the dynamic creation of objects and offer total control over object attributes, while remaining simple and easy to use.

The Tilcon Interface Builder allows you to build complex screens using simple drag and drop from a broad selection of powerful objects. The appearance and functionality of these objects can be fully customized using the Builder's straightforward property sheets.

**Portability**

The Tilcon process is separate from your application and handles all interfaces to the windowing system. Tilcon's API library is the same on all platforms (windowing systems). A clean separation between application code and the windowing system is maintained.

It is important to understand that the Interface Builder does not generate any code. When you use our Interface Builder to build a screen and save it to our twd file format. What is being saved is more in the nature of a resource file - a collection of labels, coordinates and instructions that the EVE knows how to interpret and render.

Inside the EVE are generic button routine, a listbox routine, a chart routine. From the data in the .twd file, the EVE knows where to put the window, how big to make it, how many buttons there are, where they are, what their labels are, etc. As these .twd files are data files rather than C code, you can change these files with the Interface Builder without having to recompile a thing. This approach shields the developer from volumes of generated code, minimizes time-consuming recompliations and makes the application very portable. The files are independent of operating or windowing system and usable on all supported platforms.

The Tilcon EVE has its own drawing capabilities. All of the objects are Tilcon objects and not borrowed from the various windowing systems. As a result the screens are identical from platform to platform. Minor exceptions arise out of limitations of individual operating systems. Objects can be created using the Interface Builder and saved as part of a twd file or they can be saved as Tilcon object files in the Tilcon library. Objects can also be created dynamically and in fact entire screens can be created, edited and saved dynamically. Regardless of how they are created, the look and feel will be the same on all the supported platforms. The only part of the system that is platform specific is the actual Tilcon EVE supplied for individual platforms by Tilcon.

The portability of the user interface has many implications and how important these are varies from application to application. As you can make changes to the user interface without recompiling, maintenance and upgrading is simplified and for some applications remote maintenance becomes a possibility.

All screens are entirely reusable. Your prototype becomes your application. You only need to write your application once and it is immediately available under all platforms that Tilcon supports. If you are developing a product and want to maximize your market by releasing on multiple platforms, Tilcon's capabilities are ideal. Many device manufacturers like the fact that although their actual machine or device may be using an RTOS,
their sales force are able to walk around with the same application in their Windows laptop. Same considerations apply to training systems.

Another major advantage is cross-platform development. The choice of tools in some platforms may be more limited. It may be difficult to get specialized developers. The need to constantly download to a target may be inconvenient. The ability to do the bulk of the development on a standard operating system with your favorite development environment and better tools with only fine-tuning on the target is a real advantage.

The Tilcon design tools (Interface Builder) is very user friendly and designers, artists and developers can easily work in concert even in a mixed environment. With built-in TCP/IP networking, you can display and control in a heterogeneous environment. Your data acquisition or device can be on an RTOS and it can easily integrate into your desktop environment. Having the ability to easily move to another operating system may provide leverage and freedom of choice in the future.

**Tilcon Embedded Vector Engine**

Tilcon's EVE is in effect a very sophisticated graphics driver. In addition to executing all API commands and rendering the screens, it is responsible for maintaining the main data structure that describes all of the object spaces and objects and processes window messages, triggers, notifications and callback functions.

It is the go-between for your application and the windowing system. It knows how to read a .twd file (the screens you create with the Tilcon Interface Builder), how to create a window from it and how to create all the buttons, text, charts, meters and other objects within the window. It knows what windowing system commands to use to update an object, how to get the value stored in an object and how to detect when the user changes an object. It runs side-by-side with your application and in response to its API commands loads and displays the required windows, puts values into the meters, charts and other objects. The **Engine** tells the application about text modifications, button clicks and other user interactions with objects in your interface window.

Triggers are in effect API commands that were built into the objects themselves when they were created. In response to specific events, for example, a checkbox becoming un-checked, the **Engine** executes the trigger that was built into the selected object. In this way, a button could be used to dim several fields and display a window, without the need to communicate with your application.

The Server is the part of the EVE that handles the graphics services for the user application. It replays and translates API calls to instructions understood by the **Engine**. It also receives, translates and replays UI events and data to the application.

**Application Programming Interface**

The API (Application Program Interface) is the protocol between the application and the EVE. The heart of the Tilcon API is a handful of fundamental functions. These are the functions that initiate and terminate communication with the **Engine**, query for notifications, create and delete objects and set or query object attributes. They are as follows:
1. TRT_Create
2. TRT_Delete
3. TRT_Exit
4. TRT_GetInput
5. TRT_GetValues
6. TRT_SetValues
7. TRT_SimpleCreate
8. TRT_StartEx
9. TRT_WindowLoad

These core API commands in combination with a name/value paired list of attributes affecting all of the objects characteristics enable the creation or editing of all objects. The result is an unparalleled level of dynamic configuration abilities and fine tune control to the developer over the appearance and behaviour of objects at runtime.

The key functions needed to interact with an object's properties are:

- **TRT_SetValues** - to set the format etc. of the attribute of the object
- **TRT_GetValues** - to obtain actual values for the attribute

These functions take a variable number of arguments; the last argument must always be a NULL pointer to show that the list is complete. The arguments in these functions are paired. One parameter specifies which attribute is to be changed or queried; it is immediately followed by the value to be assigned or, for TRT_GetValues, by a pointer to a variable that will hold the value of the queried attribute.

**The Tilcon Interface Builder**

The Tilcon Interface Builder is a set of powerful integrated tools designed to facilitate the creation of the user interface screens. In short, it allows you to drag and drop user interface objects onto a work window and customize all of the objects to your exact requirements. You can then save the screen to be called and displayed as part of your application. In addition to all standard user interface objects (buttons, text, combo boxes, listboxes, tree etc), Tilcon provides an array of advanced components such as table, meter, charting, spectrograph, map object and you are able to build custom objects suited to your needs as well.

All objects are easily edited by filling in the object's resource sheets. This allows you to fully customize the appearance (i.e. color, label, borders, images, fonts etc) of the object. Equally important is that much of the behaviour of the object can also be specified. Initial state, blinking, text wrap modes, display focus, help bubbles, pop up menus, cursors, shortcut keys etc. A powerful object management paradigm is employed that allows you to effortlessly specify object relationships and implement important functionality such as notification, callbacks, object linking and triggers.

Triggers are embedded API commands that allow you to act upon (delete, display, hide, dim etc) other objects. In effect much of the user interface programming is handled for you.

**Tools / features of the Interface Builder**, include the following key ones:
1. Add objects to library
2. Import files of type bmp, ico, wmf, gif, jpg, twd, tsy, two
3. Grouping and group alignment (new in Version 5.5)
4. Precise placement and sizing tools (nudge tools)
5. Control over redraw (double buffering) and anti-aliasing
6. Dynamic language change facilities
7. Built-in object browser
8. Grid control
9. Zooming
10. Cut/copy/paste/duplicate
11. Moving object Front/back
12. Patterns, line styles, line caps
13. Vector drawing capabilities
14. Control over curve smoothness
15. Alpha Blending (new in Version 5.5) allowing you to add fade-in fade-out effects to your objects and images
16. Fountain fills and non-linear fills
17. Full graphics and images integration
18. Custom toolbar
19. Customized color palette
20. Image and resource sharing manager as means of memory conservation
21. Font manager
22. Building customized
23. s with the Needle Editor (new in Version 5.5)
24. Save as bmp, twm, etc.
25. Test Feature

There is a host of other features that you will definitely find useful, but the above list is to give you an idea of the capabilities of the Interface Builder. It is itself an application of the EVE. Whatever you can do in the Interface Builder you can also do in your application. Some developers need to create their own customized development tool or editor. To facilitate this task, Tilcon does make its Interface Builder source code available to qualified developers. Please contact Tilcon Support at support@tilcon.com (agreement required).

For more on this see Resources section of the white paper or contact your Tilcon account representative.

**GUI/HMI Object Descriptions**

**Objects**

In simplest terms an object is any window, shape or text that has an ID. Each object has specific attributes such as its size and common attributes like colour that control its appearance and behaviour.
The following are the Tilcon objects: (more fully described below):

**Table 1: Table of Tilcon objects**

<table>
<thead>
<tr>
<th>Editable Text</th>
<th>Message Text</th>
<th>Label Text</th>
<th>Multi-line Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menubar</td>
<td>Button</td>
<td>Radio Button</td>
<td>Checkbox</td>
</tr>
<tr>
<td>Toolbar</td>
<td>Panel</td>
<td>Numberbox</td>
<td>Listbox</td>
</tr>
<tr>
<td>Combobox</td>
<td>Scroll Area</td>
<td>Tabbed Notebook</td>
<td>Tree</td>
</tr>
<tr>
<td>Slider</td>
<td>Scale</td>
<td>Directory Box</td>
<td>Table</td>
</tr>
<tr>
<td>HTML File Display</td>
<td>Raw Image</td>
<td>Clock</td>
<td>Charts</td>
</tr>
<tr>
<td>Spectrograph</td>
<td>Animation Area</td>
<td>Animation Object</td>
<td>Fill Meter</td>
</tr>
<tr>
<td>Needle Meter Body</td>
<td>Needle</td>
<td>Color State Object</td>
<td>Multi-state Object</td>
</tr>
<tr>
<td>Color Listbox/Combobox</td>
<td>LCD Text</td>
<td>Map</td>
<td>Progress Meter</td>
</tr>
<tr>
<td>Draw Tool</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Button**

Buttons can respond in a variety of ways to *click* depending on the button type - spring, toggle, repeating, on/off and instant button types are available. Use vector and raster images to complement or replace textual labels. Customize shape, shading, dimming, blinking and help bubbles. Notify your application of events, invoke callback functions, and/or build a list of commands to be triggered in response to the 'clicks'.

**Radio Button**

Radio buttons manage a group of buttons; selecting one item automatically deselects the others in the group. Apply vector/raster images and help bubbles to each label. Customize layout, bullets and shading. Triggers and callbacks are available for individual items, as well as notification of the selected item by the group.

**Checkbox**

Tilcon’s checkmark object offers an array of checkmarks, layouts and shadings. Add vector/raster images and help bubbles to the label. Respond to user actions by notifying the application, invoking a callback, or triggering a predefined list of commands.

**Scroll Area**

These containers with scroll bars allow you to group many or large objects into a small area of your window. Any object can be placed into a scroll area, including background images (e.g.) over which other objects can be layered. Choose the viewed size, total size, shadings, and scrollable options.
Directory Box
Directory boxes allow the user to choose from a list of files. Drive, directory, filename and file type are shown; file types can be limited so only certain types are visible to the user. An invaluable tool for 'Open' and 'Save As' forms!

Toolbar
Toolbars attach themselves to any edge of a window - top, bottom, left or right. They provide space for vital information that should be displayed even when the main contents of a window change. Clocks, warning messages and status indicators are a few objects commonly placed in toolbars, which can be customized in layout, shading, as well as in content.

Tree
Tilcon’s tree object provides a visualization of a hierarchical list of items. Trees are built quickly and easily: each item is added as a child or sibling of a previous item. Branches expand and collapse in response to user 'clicks'; selected items can trigger commands or send notifications to the application. Customize each item by adding an image, changing its shading or providing a help bubble.

Color Listbox/Combobox
Color Listboxes or comboboxes are simple tools for providing a choice of colors to the user. A palette of 256 colors is available through these listbox and combobox objects.

Listbox
Listboxes allow the selection of one or more items from a list. Tailor the listbox behavior to your needs by choosing multiple/single selection, specifying the orientation and sorting of the items, and setting the conditions under which notifications are sent, callbacks invoked, or commands triggered. Specify multiple columns with individual widths. Provide images, help bubbles, hiding, dimming and colors for each item.

Combobox
Comboboxes combine a text field and a listbox. Choose a popup or static listbox, editable or non-editable text, and an array of options for both listbox and text components: layout, colors, images, shading, help bubbles, to name a few.

Numberbox
Numberboxes are the perfect tool for numeric input. Control the layout of the editable text, repeating arrows and label. Specify minimum, maximum values, decimal accuracy, number of digits displayed, justification, arrow increments and help bubbles. Set the conditions when notifications are sent, callbacks invoked, or commands triggered. Customize the appearance by setting colors, shading and other options. Link to other objects to share a common value.
**Tabbed Notebook**
Tabbed notebooks allow you to group objects into related categories, each contained in its own folder. Selecting a tab changes the active folder, allowing many choices in a form while using little screen real estate. Tabbed Notebooks options include setting the location, shape, font, default colors, highlight colors, and labels of each tab, as well as the colors for each folder in the notebook.

**Panel**
Panels are tools commonly used for grouping objects. These containers are surrounded by a 3-D border with an optional label at the top. All objects in the panel can be moved as one, hidden, cut, copied, pasted - by performing these actions on the panel itself. The type of 3-D border, the label and all colors can be customized.

**HTML File Display**
HTML File Display will, as the name implies, display an html page given its URL. Use this to create on-line help applications, or maybe even your own web browser! Font, scrollbar and notification controls are built in (where applicable).

**LCD Text**
Tilcon’s LCD Text object simulates the segmented text of an LCD or LED screen. Ideal for realistic-looking displays, e.g. for calculator applications and consumer appliance displays. Customize the segment sizes, margins and colors.

**Raw Image**
Raw Images are used to import vector and raster images into a window. Like all objects, they have their own frame, can be moved about the window, and can send notification of mouse clicks... they can even display a help bubble.

**Text**
Editable, message, label and multi-line forms of text are available to meet the need for textual objects. Editable and multi-line text can respond to user interaction and events - gaining focus, losing focus, individual keystrokes - by sending a notification, invoking a callback or triggering a command. International text is supported, as are input masking, formatted output, and a wide assortment of fonts and sizes. Other attributes include justification, colors, dimming, hiding, linking to other objects, blinking, and help bubbles.

**Table**
Tables provide spreadsheet-like control of your data I/O. Each column can contain its own type of object, including editable text, message text, raw images, numberboxes and comboboxes. Every cell has all the options and attributes available to the type of object it contains, so tables provide vast amounts of built-in functionality and configurability. In addition to which the table has many attributes of its own, including row/column dimensions, scrollbar options, grid line options and colors, and sorting. And, as with all objects, every attribute can be modified dynamically!
**Fill Meter**

Fillmeters are invaluable tools for process-oriented applications where a value is represented by the fill level within an outer body. Updates are smooth, fast and flicker-free. In addition to the predefined shapes (square, rounded, triangular, thermometer), vector images can be imported. Fully configurable scales are automatically attached; calculations can be built in; and the value can be linked to other objects. The meter can fill from its minimum or from 0; it can also change its fill color automatically as its value crosses 0. Other features include control of the colors and borders of each component.

**Needle Meter**

Needle meters are formed after adding one or more Needles into a Needle Meter Body. An array of body shapes is available, defining the behavior of the needles. Meter bodies can also be hidden so the needles are shown over imported images, increasing their realism. As with fill meters, scales are automatically attached, and needle updates are smooth and flicker free.

**Needles**

Needles offer an array of predefined shapes and widths; vector images can also be imported to increase realism. Links can be made to other objects; calculations can be applied to values before they are displayed. Other options affect colors, tip/tail lengths, and pin shape and size. Anti-aliasing options are currently available for all predefined needles. This feature is being extended to meter bodies and will soon also be extended to user defined needles.

**Color and Multi State Object**

Color and Multi-State Objects have numerous applications, the most obvious as alarms in process control applications. They are configured to have one or more 'states' corresponding to a range of values. When the value in the object crosses the boundary from one state to another, either the color or symbol of the object changes, and the object can trigger a command or notify the application of this event. Vector images are imported and used as the symbols. Blinking can be set for each state individually; links can be made to other objects; and calculations can be applied to values before state changes are assessed.

**Scale**

Numeric, alphanumerical, date and time stand-alone scales provide the perfect complement to drawings, animation and other objects. With the choice between linear or circular shapes, linear or logarithmic increments, a tri-colored alarm indicator, and a battery of configurable options for the text and tickmarks, the scale offers complete versatility.

**Menubars & Pop-up Menu**

Placed at the top of the window, the menubar provides multiple levels of drop-down menu items. The cascaded sub-menus provide command, toggle and radio button behaviour for each of their items. Each item can send a notification, invoke a callback or trigger a command upon its selection. Additionally, Pop-up Menus can be created in most
objects; these appear when a 'right-click' occurs on an object that has defined a pop-up menu.

**Slider**
Sliders provide a visual method for numeric input. The slider offers predefined and custom handle images, optional endbuttons, attached scales, and a choice of directions. Customize the layout and appearance of each of its components. Set the conditions under which the slider responds to user input - by notification, callback or trigger.

**Clock**
The clock object allows for the display of current date or time in a variety of formats. Fonts and colors are also configurable.

**Chart**
Tilcon charts represent data as a function of an independent (x) variable, where each y-series provides a data point for each xvalue.

The x-data may be numeric, alphanumeric, date or time; attached scales can represent each of these cases. Charts offer multiple ways of plotting their data: as scattered points, as connected lines, as solid areas, as pie slices, as vertical or horizontal bars, as stacked bars or areas when more than one series is supplied. Series can also be used to create x/y error bars or high-low plots. Other features include auto-scaling, dual y-scales, markers for each point, individual colors for each line/area/marker, limits above/below which plotted colors change, linear/log increments for each scale, and many other options affecting the chart's appearance. Functions are provided to update the chart one point at a time or in groups of points (like an oscilloscope) or to scroll data onto the chart (like a strip-chart recorder). Like in other objects, every attribute of the chart is fully dynamic and can be changed either during development or programmatically.

**Spectrograph**
Tilcon's spectrograph object displays a raster image in an array of cells. However, this data is supplied by directly setting the colors for each cell rather than through the use of an image file. With a configurable cell size, variable number of rows and columns, and the ability to scroll the spectrograph's data towards any edge, this object provides a simple yet powerful alternative to raw images for applications that have immediate access to the data that will form the image.

**Animation Area and Object**
Animation areas and objects provide smooth motion for vector images and raster images, and smooth sizing and rotation for vector images. The area defines the space in which the objects may move and sets up a coordinate system scaled to that area.

Object position, and the size and rotation of vector objects, are calculated with respect to the animation area coordinates.
**Draw Tool**

The Tilcon Interface Builder offers powerful drawing tools with 24 unique drawing shapes, full bezier and polygon editing, rotate, skew, mirror, attach arrows, line widths and types are provided allowing you to create the symbol shapes, logos and designs you need in your application. Combined with the ability to import vector images and modifying them dynamically, gives you powerful animation capabilities.

The twelve universal drawing tools are Bezier curve, polygon, trapezoid, diamond, ellipse, parallelogram, rectangle, roundcornered box, straight line, triangle, arrow and arc. In addition to these, there is a selection of twelve standard flow chart tools to help you create multi-component charts quickly and easily. The flow chart tools are: manual input, gate, manual operation, display, preparation, on-line string, punch tape, keying, document, off page and transfer tape. Each drawn shape can be moved, resized, extended, reduced and manipulated alone or as a group to produce your drawing. Each shape created with the draw tools can be reshaped with the available option features.

**Map**

This object is completely described by its components. The Map APIs (q.v.) enable users to build applications that manipulate and display map data on this object.

**Progress Meter**

A Progress Meter, or a P-Meter as it is commonly called, can be used in place of a needle meter - it functions in the same way as the needle meter does. For this type of meter, it is possible to change the colours, sweep angles, start and end angles, radii, etc. and link it to other objects in the group.

**Components**

Objects can contain components. For example, the button object can have alarms which are components.

<table>
<thead>
<tr>
<th>Table 2: Table of object components</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Alarms</strong></td>
</tr>
<tr>
<td>Borders (Frames)</td>
</tr>
<tr>
<td>Common</td>
</tr>
<tr>
<td>Fields</td>
</tr>
<tr>
<td>Indicators</td>
</tr>
<tr>
<td>Popup Menu</td>
</tr>
<tr>
<td>Scrollbars</td>
</tr>
<tr>
<td>Triggers</td>
</tr>
</tbody>
</table>
Integration with Graphics

Tilcon prides itself in making it very easy for its customers to build graphically impressive and fully custom user interfaces. Part of those capabilities comes from the range of attributes and the handling of colors, transparencies, fills, anti-aliasing, double buffering and other features of the user interface objects by the Tilcon's Embedded Vector Engine.

Tilcon provides a range of clip arts in its clipart directories. Support is provided for various image formats allowing customers to use existing assets. Extensive demos are provided, any part of which (including buttons etc) is fully reusable by licensed customers.

A developer who has the help of a competent graphics artist will have no difficulty in recreating any of the effects seen in the Tilcon demos. Tilcon has in-house graphics artists who are skilled with Tilcon tools and they are available for consulting work on projects if your project has no graphics artist resources.

Dynamic Language Update and Internationalization

Tilcon supports international character sets in all text tools. In addition to TRT_SetValues function that allows the change of attributes one-by-one, Tilcon provides a mechanism that makes it easy for your application to globally change the language of the labels in your user interface. The function that implements this mechanism is called TRT_ChangeLanguage.

This function uses the contents of a file to replace text labels, whether of GUI objects, text fields, help bubbles, menus or window titles. At development time, you would start with the Save Language File option found in the Tilcon Interface Builder. Selecting this option generates a Tilcon language file (.twl) for that window. A .twl file is encoded in UTF8 and each line contains the ID of the Tilcon object followed by all the text to use within that object. The language files can be merged with other .twl files, or individual entries can be added or deleted until you have a satisfactory template. As you need a language file for each language you want to support, you would then make a copy of this file for each language and then have the files translated (make the language specific changes to the labels in each copy).

TRT_ChangeLanguage will read and display the labels from a language file (.twl) and changing language is simply a matter of pointing to the right language file.

The Scalability Utility

The Tilcon IDS enables you to create a custom-configured graphics engine, which is composed only of the components required by your application. With this scalable kernel option, you can compile a much smaller kernel using less RAM. The scalable option allows you to discard any unnecessary software overhead. This of course, is especially significant in embedded systems where memory is at a premium. An application, for example, that never uses animation does not require a graphics engine kernel with animation capabilities.

Using a utility in the IDS and with a simple point and click interface, you can select the desired capabilities from the available set of Tilcon libraries. The utility then produces a
makefile that can be used to link the appropriate libraries to create a scaled down graphics engine kernel. Additional details and memory requirements of individual objects are appended. The other utilities that are available to you are the Tilcon Resource Dictionary, utility, the TTR Utility and the Trtver Utility.

**Memory Requirements**

The following table indicates approximate memory requirements for each component. The memory size for a given component, includes the memory requirements for the other included or required components.

<table>
<thead>
<tr>
<th>Component Object Name</th>
<th>Other Required Components</th>
<th>MS XP (KB)</th>
<th>Vx 6.2 DKM PPC (KB)</th>
<th>Vx 6.2 RTP x86 (KB)</th>
<th>QNX 6.3 PPC (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel size, all objects</td>
<td></td>
<td>1441</td>
<td>3227</td>
<td>3479</td>
<td>2419</td>
</tr>
<tr>
<td>Kernel Size, Minimal Objects (Defaults)</td>
<td>*Scroll Area, Edit Text, Label Text, Message Text</td>
<td>673</td>
<td>1631</td>
<td>1789</td>
<td>1170</td>
</tr>
<tr>
<td>Slider</td>
<td>Scale, Defaults</td>
<td>84</td>
<td>162</td>
<td>159</td>
<td>120</td>
</tr>
<tr>
<td>Spectrograph</td>
<td>Defaults</td>
<td>8</td>
<td>16</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td>Multistate Object</td>
<td>Defaults</td>
<td>28</td>
<td>63</td>
<td>70</td>
<td>45</td>
</tr>
<tr>
<td>Tabbed Notebook</td>
<td>Defaults</td>
<td>20</td>
<td>51</td>
<td>62</td>
<td>40</td>
</tr>
<tr>
<td>Progress Meter</td>
<td>Defaults</td>
<td>36</td>
<td>84</td>
<td>76</td>
<td>64</td>
</tr>
<tr>
<td>Table</td>
<td>Button, List Box, Number Box, Combobox, Defaults</td>
<td>148</td>
<td>375</td>
<td>434</td>
<td>285</td>
</tr>
<tr>
<td>TextEditable</td>
<td>This is a default object</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TextLabel</td>
<td>This is a default object</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TextMessage</td>
<td>This is a default object</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Toolbar</td>
<td>Defaults</td>
<td>4</td>
<td>11</td>
<td>17</td>
<td>8</td>
</tr>
<tr>
<td>Treeview</td>
<td>Defaults</td>
<td>52</td>
<td>120</td>
<td>138</td>
<td>94</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Defaults</td>
<td>0</td>
<td>NA</td>
<td>NA</td>
<td>0</td>
</tr>
<tr>
<td>Map</td>
<td>Defaults</td>
<td>8</td>
<td>NA</td>
<td>NA</td>
<td>15</td>
</tr>
<tr>
<td>ODBC</td>
<td>Defaults</td>
<td>28</td>
<td>NA</td>
<td>NA</td>
<td>0</td>
</tr>
<tr>
<td>OPC</td>
<td>Defaults</td>
<td>40</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3D Panel</td>
<td>Defaults</td>
<td>4</td>
<td>13</td>
<td>16</td>
<td>608</td>
</tr>
<tr>
<td>Animation</td>
<td>Defaults</td>
<td>24</td>
<td>48</td>
<td>51</td>
<td>34</td>
</tr>
<tr>
<td>Button</td>
<td>Defaults</td>
<td>20</td>
<td>50</td>
<td>57</td>
<td>39</td>
</tr>
<tr>
<td>Chart</td>
<td>Scale, Defaults</td>
<td>172</td>
<td>408</td>
<td>389</td>
<td>312</td>
</tr>
<tr>
<td>Clock</td>
<td>Defaults</td>
<td>8</td>
<td>16</td>
<td>17</td>
<td>11</td>
</tr>
<tr>
<td>Combobox</td>
<td>List Box, Defaults</td>
<td>64</td>
<td>152</td>
<td>184</td>
<td>114</td>
</tr>
<tr>
<td>Directory Box</td>
<td>Treeview, List Box, Combobox, Defaults</td>
<td>92</td>
<td>223</td>
<td>271</td>
<td>164</td>
</tr>
<tr>
<td>Component Object Name</td>
<td>Other Required Components</td>
<td>MS XP (KB)</td>
<td>Vx 6.2 DKM PPC (KB)</td>
<td>Vx 6.2 RTP x86 (KB)</td>
<td>QNX 6.3 PPC (KB)</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------</td>
<td>------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td></td>
<td>Kernel File Name:</td>
<td>trtkmlw.exe</td>
<td>trtkernel.o</td>
<td>trtkernel.a</td>
<td>trtkrnln.exe</td>
</tr>
<tr>
<td>HTML</td>
<td>Defaults</td>
<td>4</td>
<td>NA</td>
<td>NA</td>
<td>9</td>
</tr>
<tr>
<td>LCD</td>
<td>Defaults</td>
<td>16</td>
<td>35</td>
<td>33</td>
<td>29</td>
</tr>
<tr>
<td>List Box</td>
<td>Defaults</td>
<td>40</td>
<td>92</td>
<td>102</td>
<td>74</td>
</tr>
<tr>
<td>JPEG Image Support</td>
<td>Defaults</td>
<td>64</td>
<td>46</td>
<td>47</td>
<td>54</td>
</tr>
<tr>
<td>Menu</td>
<td>Defaults</td>
<td>24</td>
<td>51</td>
<td>59</td>
<td>39</td>
</tr>
<tr>
<td>Meter</td>
<td>Scale, Defaults</td>
<td>152</td>
<td>301</td>
<td>281</td>
<td>243</td>
</tr>
<tr>
<td>Number Box</td>
<td>Defaults</td>
<td>12</td>
<td>36</td>
<td>45</td>
<td>24</td>
</tr>
<tr>
<td>Radio Button</td>
<td>Defaults</td>
<td>24</td>
<td>57</td>
<td>64</td>
<td>46</td>
</tr>
<tr>
<td>Scale</td>
<td>Defaults</td>
<td>56</td>
<td>108</td>
<td>98</td>
<td>81</td>
</tr>
<tr>
<td>Scroll Area</td>
<td>This is a default object</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

This information is provided for illustration purposes only and not warranted. Data is based on specific version under test at the time. Actual memory requirements will vary from version to version as code changes are made and new features added.
Tilcon Networking Support

Tilcon has implemented a powerful channel object, which gives the user built-in support for many alternatives in application and network architecture. The channel object is in effect a communication nexus encapsulating the transport mechanism, a mailbox if you will through which messages enter and leave the program. The transport mechanisms currently implemented include TCP/IP and IPC (inter-process communications). IPC is used for Tilcon threads or tasks within the same host.

The IPC mechanisms supported are as follows:

- Message passing - (Neutrino, QNX)
- Pipes - (VxWorks)
- Shared Memory - (Windows NT, CE)

1. Within the same host, the channels support communication between the Tilcon Engine with one or more user application programs. The topology architecture looks like the one in Figure 2 below:

   Figure 3 - The Tilcon Engine with one or more user application programs

   P1, P2, P3, P4 and the Tilcon EVE are on the same host. This architecture is the most common work form. P1, P2, P3, P4 are Tilcon applications. They call Tilcon APIs to connect to the Tilcon EVE.

2. Support for communication between the Tilcon EVE and remote client(s).
C3, C4 and the EVE are not on the same host. They must be connected by a communication line. At present, built-in support for TCP/IP connection is provided. Requests for support for additional networking protocols will be considered.

3. One Tilcon application can set up connection with more than one EVE. The default connection limit is 64, but the user can set a different limit value by configuration file. The EVE(s) can be located on different hosts. The client application can be located on the same computer where one of the graphics engine is located.

Communication support is provided for Tilcon GUI applications (TGA) to communicate with any other application.
• Application – to – Application communication is currently available using IPC only
• Application to application communication using TCP/IP is under development
• Threads cannot share channels
• Channels are implemented using SRR (Send->Receive->Reply) providing advantages for real time applications
• Tilcon channels are modular and designed to accommodate additional protocols
• Engine to Engine ditto feature where application sends to one graphics engine and a command

Redraw options (offscreen draw or double buffering)

In order to update an object, it must be redrawn. All or part of the object must be cleared and then drawn again to reflect new values or settings. This re-draw of the object may be done directly on the screen, or it may first be produced as an offscreen image, which is copied to the screen. There are differences between Onscreen and Offscreen Redraw.

Redrawing objects Onscreen is the quickest way to update them. However, because the old setting must be cleared from the screen to draw and display the updated settings, the user often sees a flicker in the application where the clearing occurs. The Onscreen option is best used when the application must make a quick update of an object that does not change very frequently.

By drawing first into the offscreen image and then copying that image to the screen, the user never sees the screen cleared, and therefore sees no flicker. This is a recommended method for updating objects which are constantly changing such as a needle meters and fillmeters.

If the object to be updated has other superimposed objects, you must decide whether you want to draw just the object or the area that encompasses it.

Redraw Object

This is the choice when an object stands alone, or when no other objects are layered above or beneath it. Drawing only one object is much faster than redrawing its area, but that object will be drawn over any other object at that location.

Area redraw

This option redraws the object that you are updating and all other objects, which are layered on top and below it. In effect, it draws the "area" which is affected by the update. In this way, an object is redrawn in its correct position amidst the layers. If the area is drawn Onscreen, flickering will appear for the updating of the area. If it is done Offscreen, the layers are drawn from the back to the front, and appear much smoother when displayed.

Never Redraw

This option is useful for objects you do not wish to have redrawn when you know another update is forthcoming. 'Never' objects do not redraw themselves when they are updated, only when the area they are in must be redrawn.
The Tilcon redraw options can be fixed as the screens are being designed in the Graphics Editor or programatically using api commands.

**Map Module**

The Map module enables users to manipulate and display geographic information; support is provided for the import of standard ADRG, CADRG and DTED (Digital Terrain Data), (DTED implementation not finished as of the date of this paper); Tilcon's Map API comprises over 80 API function calls to manipulate geographic data. This API currently addresses the manipulation of map data originating from paper documents, that is, paper maps that have been scanned. The API functions allow the registration of such data to selected geographic projections (below), and the creation of databases of registered map files. The databases use a tiled file structure.

Tilcon's map library also includes functions for displaying grids and digitizing and playing back routes, Zoom, Pan and Animate objects (these calls are implemented in the demo and illustrate how to use the API to achieve the related functions); Utility functions for handling commonly-used geographic coordinate systems;

Rasterized map data are stored as Microsoft Device Independent Bitmaps (BMP files) for which the Tilcon Engine provides support. Geo-referencing of these files is handled by structures embedded in the map library. Coverage databases comprise matrices of georeferenced BMP files.

Utility functions to change reference datums to and from WGS84-based datums as well as converting between geographic coordinates and projection coordinates are provided. At the moment the widely used UTM projection is fully supported together with MGRS-encoding of the UTM values, as are the UPS projection for polar regions, the Albers equal-area conic and the Lambert conformal conic projections with two standard parallels, Generic TM projections such as Gauss-Kruger and the MTM projections is forthcoming.

Functionally for removing systematic distortions from scanned paper map products is an important aspect of the map library which enables users to register scanned products to the projection coordinate system, and is a first step to the creation of the coverage databases. Additional functionality to create a matrixed database of map tiles from scanned maps is also provided. Functionality for merging adjoining databases is also provided.

A table of datums and associated ellipsoids and geographic areas is supplied to assist users in selecting appropriate values for their areas of interest. A table of coefficients for datum shifts to WGS84 using the MRE method is also supplied together with a table containing parameters of thirty-four commonly used ellipsoids.

A database of Map data is created by a utility. This database comprises many .bmp files (Tiles) as well as a .toc file, which describes the contents of the database.

An image object is created in the display window. This object is referenced to the real-word. API calls seek out the area of interest in the dbase and create an image in memory, which is displayed.
The application may then instruct the viewer to manipulate the map to achieve the desired results using the API calls provided in the library.

The API also contains functions for forward and reverse conversions between geographic coordinates and projection coordinates (for the supported projections) as well as functionality for converting to and from WGS84-based datums.

The full Map Module Reference Manuals is available for download in .pdf format for additional details.

**Other Utilities**

**TTR**

The TTR utility is used to produce reports about the contents of a .twd file (the window or screen contents). A report can list all the objects in the .twd file in a hierarchical format, showing all the objects and their parents, siblings and children. Another report provides detail on attributes (object properties) within the .twd file. The user is able to define which of the properties to report on. TTR Utility’s main purpose is to facilitate generation of documentation about the contents of the user interface.

**Resources Available**

**The Main Tilcon Demo**

The Tilcon demo is a suite of sample Tilcon applications intended to demonstrate the range and capabilities of the Tilcon Graphics Engine. It includes applications such as a car dashboard demo, reconfigurable meters, oscilloscope, reconfigurable charts, strip charts, medical device demos, process control (beer factory), calculators, cellphones and others.

All of the source code for the demo comes with the download allowing you to further examine the structure of Tilcon applications.

If you also downloaded a Tilcon evaluation system, all of the demo screens (twd files) can be opened in the Tilcon Interface Builder and essentially reverse engineered to see exactly how the screens were built. All you would do is open the file (using file open in menu of editor), select any object or component and right click to open up the property sheets for that object. The object browser (button on the toolbar if not already open) allows to easily pick out objects that are small, layered or obscured by other objects.

**The Tilcon Map Demo**

The Tilcon Map Demo a fully integrated map viewer demonstrating the display and manipulation of matrixed databases of maps which includes features such as ROAMER (to quickly navigate the entire map database), 8 button scrolling, Zoom, Map Centering, Soft Zoom, Route Digitization and Playback, Sites of Interest database and other features. As the map database is quite large, this is a substantial download and separate from Tilcon’s main demo. The map demo source code is also available.
**Tilcon Evaluation Systems**

Tilcon makes available a 14-day evaluation system in its standard configuration on multiple platforms. Although the evaluation systems are time limited, they are fully functional versions of the software. Full online help is provided, the Tilcon Tutorials are provided as well as the sample and clipart libraries.

You will need to contact Tilcon in the event of your encountering any licensing or software problems.

**Tilcon Tutorials**

The tutorials are designed to be systematically stepped through and will teach you step by step how to create screens with the Tilcon *Interface Builder* and how to bring them to life with your C/C++ application. In so doing it explains much of the basic functionality and features of the software and are highly recommended to new users.

**Tilcon Examples**

The TE4 directory that comes with the Tilcon evaluation system includes basic application templates as well as more complex items such as TCP/IP examples. The source code from the demo is also very useful and contains a multitude of code examples.

You can also contact Tilcon Technical Support to obtain additional examples for specific issues.

**Tilcon Documentation**

Tilcon’s full product documentation is contained within the online help. This includes complete documentation for the *Interface Builder* and all of the Tilcon API functions. The documentation for the Map Library and various other utilities are also available.

**Tilcon Technical Support**

In addition to your Tilcon account manager, you also have free access to the engineers in our Technical Support Department for the duration of your evaluation period. After your purchase, Technical Support is also available. As you will note from the customers comments on our website, the Tilcon Technical Support team has received very high praise from our customers and they are invaluable in helping you get up to speed and productive quickly.

**Graphics Editor Development**

Occasionally, developers enquire about using the Tilcon as a graphics engine to construct their own specialized editor or toolkit for dedicated applications. They want to be able to construct their own form or graphics editor, to be able to give their end users extensive abilities to modify screens and on their own label. This can be achieved using the Tilcon *EVE*. In fact, that is precisely what we did! The Tilcon *Interface Builder* is nothing else but an application of the Tilcon *EVE*. It is a reasonably complex application and depending on the extent of functionality that a customer requires, could be an extensive project to undertake.
To make it easier for customers to build their own fully custom graphics or forms editor, Tilcon has available for sale the source code for the Tilcon Interface Builder.

Consulting And Training On The Tilcon Interface Builder
Tilcon has highly trained and experienced engineers and trainers who can be of invalu-
able assistance in getting your project off to a fast start. They are highly knowledgeable
in Tilcon internals and can also be of great help in the fine-tuning and optimization of
your application prior to commercial release.

Graphics Services
Tilcon has in-house graphics artists who are experienced with Tilcon software tools and
available to assist customers in creating a custom look and feel for their application.

Sample application
Here is the skeleton of an application for use with Tilcon. Communication is opened with
TRT_Start, and an initial window is created and displayed. Within the main loop,
TRT_GetInput waits for the Tilcon Engine to report an event. Your application provides
some form of ProcessTilconNotification function to be called when notifications are
received from the Runtime; TRT_GetInput takes care of processing callbacks.
TRT_GetInput can also be set up so that messages from other processes can be
received, or it can wait for semaphores to signal events, allowing for the Engine’s pro-
gram to communicate with other processes. At some point, in response to some event,
the program will break out of the infinite loop and call TRT_Exit, which will tell the Engine
to terminate.
Figure 6 - The flow of the application
This is a very simple application that uses the Embedded Vector Engine V 5.x. It creates a window and two buttons. On one of the buttons the window will change colour. To compile you must link this to trtapi.lib.

**Table 4: Sample Application**

```c
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <tilcon/TRTConst.h>
#include <tilcon/TRT_API.h> // Tilcon API functions

#ifndef CC_TRT_DOS
#include <windows.h>
#endif

#define MAIN_WINDOW_FILE "filename"
define MAIN_WINDOW_ID "windowID"

pid_t        TRT_cid; // Identifies Tilcon Runtime
TRT_ReceiveData rec_data; // Record from API

>Main Entry Point

#include CC_TRT_DOS
int PASCAL WinMain (HINSTANCE hInst, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int nCmdShow ) // MS Entry Point
{
#endif
#else
int main ( int argc, char *argv[] ) // NTO Entry Point
{
#endif
long errorcode = 0;
int c;
int ContinueLooping = TRUE;
TRT_StartData StartData;
#ifndef CC_TRT_VXWORKS
StartData.Os_Env = TRT_VXWORKS;
#endif
#ifndef CC_TRT_NTO
StartData.Os_Env = TRT_NTO_PHOTON;
#endif
#ifndef CC_TRT_DOS
StartData.Os_Env = TRT_MS_WINDOWS;
#endif
```
ifdef CC_TRT_WCE
    StartData.Os_Env = TRT_WCE;
#endif

ifdef CC_TRT_LINUX
    StartData.Os_Env = TRT_LNX_X11R6;
#endif

sportnalse------------------------
Initialize the StartData structure
sportnalse------------------------/
StartData.Display = NULL;
StartData.IPAddr = NULL;
StartData.AppName = "TUTORIAL";
StartData.Userprog = "TUTORIAL";
StartData.Flags = 0;
sportnalse------------------------
Start the Embedded Engine
sportnalse------------------------/
errorcode = TRT_StartEx (0, &StartData); // Start the runtime and connect to it
TRT_cid = StartData.TRT_CID;
if(errorcode)
{
    printf("Cannot Start Runtime\n");
    exit(0);
}

sportnalse------------------------
Call TRT_Debug
sportnalse------------------------/
if(!errorcode)
    errorcode = TRT_Debug(TRT_cid, 3); //forces all API commands to be syn-
chronous

sportnalse---------------------------------------------------------------------
Load Tutorial Window
sportnalse---------------------------------------------------------------------/
errorcode = TRT_WindowLoad(TRT_cid, MAIN_WINDOW_FILE);
// Load the Tutorial Window
if(errorcode)
{

char ErrorMessage[] = {"Cannot load the Main Window! File NOT found"};
TRT_MessageBox( TRT_cid, NULL, "Example",
ErrorMessage, NULL, "OK", NULL,
TRT_MessageBox_FontHelvetica, 14,
TRT_MessageBox_Exclamation | TRT_MessageBox_Wait, 0, NULL,
NULL);
TRT_Exit (TRT_cid);
exit(0);

/*******************
Display Tutorial Window
*******************/
errorcode = TRT_WindowDisplay(TRT_cid,MAIN_WINDOW_ID);
// Display the Tutorial Window
if(errorcode)
{
  char ErrorMessage[] = {"Cannot Display Main Window"};
  TRT_MessageBox( TRT_cid, NULL, "Example",
    ErrorMessage, NULL, "OK", NULL,
    TRT_MessageBox_FontHelvetica, 14,
    TRT_MessageBox_Exclamation | TRT_MessageBox_Wait,
    0, NULL, NULL);
  TRT_Exit (TRT_cid);
  exit(0);
}

/**********************
Main Loop
Wait for communication from Engine
Act on notification
**********************/
while(ContinueLooping)
{
  c = TRT_GetInput(NULL, 0, NULL, 0, &rec_data, TRT_BLOCK);
  // Wait for a notification
  switch(c)
  {
    case 0:
    {

switch (rec_data.code)
{
    case TRT_window:
        if (rec_data.state == TRT_window_quit)
        {
            ContinueLooping = FALSE;
            break;
        }
    case TRT_button:
        break;
    case 1:  // Callback
        break;
    case -1:  // Non Blocking
        break;
    default:
        break;
}

Cleanup and exit

TRT_WindowDelete (TRT_cid, MAIN_WINDOW_ID);  // Close Window
TRT_Exit (TRT_cid);  // Notify Runtime to exit
return(0);
System Requirements

The recommended system configuration to use the Tilcon IDS has been outlined here. The requirements vary according to the platform on which the application is run; therefore the minimum requirements for the various platforms are listed below:

*Table 5: Recommended minimum system requirements*

<table>
<thead>
<tr>
<th>Platform</th>
<th>Minimum Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP</td>
<td>3.5 MB RAM for the Engine&lt;br&gt;1.5 MB RAM for the Builder&lt;br&gt;32-bit Kernel&lt;br&gt;Windows running&lt;br&gt;VC++ Compiler</td>
</tr>
<tr>
<td>Windows CE</td>
<td>3.5 MB RAM for the Engine&lt;br&gt;1.5 MB RAM for the Builder&lt;br&gt;32-bit Kernel&lt;br&gt;Windows running&lt;br&gt;Embedded VC++ Compiler OR,&lt;br&gt;CE Platform Builder&lt;br&gt;or, Board Manufacturer development system</td>
</tr>
<tr>
<td>Wind River VxWorks 5.5</td>
<td>3.5 MB RAM for the Engine&lt;br&gt;1.5 MB RAM for the Builder&lt;br&gt;32-bit Kernel&lt;br&gt;Windows running&lt;br&gt;VxWorks 5.5.1&lt;br&gt;Tornado 2.2.1&lt;br&gt;WindML 3.0.3</td>
</tr>
<tr>
<td>Wind River VxWorks 6.2</td>
<td>3.5 MB RAM for the Engine&lt;br&gt;1.5 MB RAM for the Builder&lt;br&gt;32-bit Kernel&lt;br&gt;Windows running&lt;br&gt;VxWorks 6.2&lt;br&gt;Workbench 2.4&lt;br&gt;WindML 4.1</td>
</tr>
<tr>
<td>QNX 6.3</td>
<td>3.5 MB RAM for the Engine&lt;br&gt;32-bit Kernel&lt;br&gt;QNX 6.3 SP2</td>
</tr>
<tr>
<td>Redhat Enterprise Linux v.4</td>
<td>10 MB RAM for the Engine&lt;br&gt;10 MB RAM for the Builder&lt;br&gt;32-bit Kernel&lt;br&gt;Embedded VC++ Compiler&lt;br&gt;X Window System</td>
</tr>
</tbody>
</table>