Windows CE Development for RISC Computers Made Easy

By: Thomas Walters and Earle Foster

Windows CE (a.k.a. Windows Embedded Compact), originally released in 1996, can be an attractive alternative to the full Windows operating system offering system designers the ability to control power, size, and cost in a new product. For example, the runtime license cost of the CE operating system alone can save as much as $150 over the license cost for standard Windows. Additionally, the componentized nature of CE is recognized as more stable and reliable than the Windows for desktop machines.

However, software developers often overlook the option for Windows CE when choosing an operating system. Sometimes this is the result of making hardware selection first and choosing a platform not suitable for CE. More often, lack of understanding of CE and how to use it causes system designers to ignore this option. By considering both hardware and software early in new product planning, an optimal system design using CE can be the result.

CE is usually associated with small, low power devices with no or minimal display and processing capabilities found with handheld devices. While CE is indeed useful for this type of product, advances in processors and improvements to CE itself make it useful for a wider range of applications. A selection of fast, low power RISC processors offers an inviting choice. Sealevel Systems, Inc. has developed a series of products using Atmel ARM9 processors including single board computers, compact industrial computers, and Human Machine Interfaces (HMIs). Each of these hardware platforms can offer significant advantages over similar products designed with x86 architecture including:

- Low power consumption
- Low cost
- Small size
- Wide temperature range
- Solid-state operation

Sounds good, right? But what about software development? Many Windows developers just aren’t sure what’s involved with programming for Windows CE. However, by picking a platform that is designed for Windows CE, developers can create applications in familiar environment.

An example platform is Sealevel’s HazPAC™-R9 flat panel computer system. The HazPAC-R9 supports the Microsoft .NET Compact Framework enabling developers to quickly and easily develop and run managed applications on an embedded platform. The .NET Compact Framework supports Visual Basic and Visual C# development. In this White Paper, we take a look at the HazPAC-R9 and how to connect, debug and deploy an application on the system.
HazPAC-R9 Description

The HazPAC-R9 combines a rugged RISC computer with an LCD and touchscreen for a complete HMI solution. The system is based on the 400MHz Atmel AT91SAM9G45 microprocessor boasting a 32-bit ARM instruction set for maximum performance. Approved for Class 1 Division 2 and NEMA 4/IP65 certifications, the system is a perfect choice for harsh or hazardous environments. The HazPAC-R9 features a bright LED-backlit 8.4” TFT LCD with resistive touchscreen in a rugged panel mount design. Standard features include Ethernet, serial, USB, and digital inputs and outputs. A wide operating temperature range of -30 – 70°C is supported with no heaters or cooling fans required.

To provide the fastest time to market, the HazPAC-R9 software package is equipped with the Sealevel Talos I/O Framework, which offers a high-level object-oriented .NET Compact Framework (CF) device interface. This interface provides an I/O point abstraction layer with built-in support for the specific needs of analog and digital I/O such as gain control and debouncing.

Connecting to a Host

Software development for the HazPAC-R9 is done on a PC and the compiled code is downloaded to the HazPAC-R9 for test, debug, and eventually deployment. For a host PC running Windows Mobile Device Center, Microsoft’s ActiveSync application is required to establish connection to the HazPAC-R9. After installation of ActiveSync on the host, a negotiation will begin between the PC and the HazPAC-R9 and the ActiveSync main dialog box appears on the host. Clicking the “Explore” icon opens a standard Windows Explorer where the default file contents of the HazPAC-R9 can be read or written. (See Figure 1).

![Figure 1. Connecting with ActiveSync](image)

Application Development

Since Microsoft’s .NET Compact Framework is coupled with the Sealevel Talos .NET Framework, C# and VB.NET, programmers can develop powerful embedded applications on the HazPAC-R9 in a familiar environment. Writing .NET applications for the HazPAC-R9 is very similar to writing desktop or console applications for Windows XP, Vista, and 7. The only difference is the amount of resources available. Because the memory footprint is smaller compared to a desktop computer, care should be taken where allocation of memory is concerned, such as large object creation. The Talos Framework allows access to the more specific I/O sections of the HazPAC-R9 such as digital I/O points and serial ports.
Development Requirements

- Visual Studio Professional 2005 or 2008
- .NET Compact Framework 3.5

Once all configuration options are complete you will see a console application template called HelloWorld in Visual Studio. (See Figure 2).

You can now add the references to the Talos Framework. Right click on the “References” and click the "Add Reference..." selection. (See Figure 3).

Now that the Talos Framework has been referenced, you have access to all the I/O points exposed on the HazPAC-R9 device.

For this simple HelloWorld application, we will just echo the string “Hello World” in the console window. This can be accomplished by adding the following code to the automatically created Program::Main() method. This code will echo “Hello World” and then pause for 5 seconds.
static void Main(string[] args)
{
    Console.WriteLine("Hello World");
    System.Threading.Thread.Sleep(5000);
}

From Visual Studio’s menu bar, select “Build ➔ Build HelloWorld”. After the build process has completed select from the same menu bar, “Build ➔ Deploy HelloWorld”. A “Deploy HelloWorld” dialog will appear for you to choose the appropriate target. Choose “Windows CE Device” then press the ‘Deploy’ button. (See Figure 4).

After the deployment phase, the “Hello World” message will appear on the Debug Serial console output (See Using the Debug Port section).

Examples can be found from the installation directory under ‘.\R9 Development\Samples\C#’ and ‘.\R9 Development\Samples\VB.NET’.

Debugging your HazPAC-R9 applications is a simple process that requires a USB cable or Ethernet connection, Microsoft Device Synchronization software, and Visual Studio.

**Debugging an Application**

Once the HazPAC-R9 has been successfully attached to the host PC, it is easy to begin debugging an application on the HazPAC-R9. This section will describe how to attach the Microsoft Visual Studio debugger to the HazPAC-R9, show the use of breakpoints in the debugger, and show how to access useful information while debugging an application.

We will be using the GPIO example application found in the "samples" directory of the Talos Framework installation. The same methods will apply to debugging any application on the HazPAC-R9.
Attach the Debugger
Once your solution is opened, it is necessary to specify the device target that you would like to use in conjunction with the debugger. The default option is an emulator. Click "Windows CE Device" from the target device drop down. Then click the "Connect to Device" button. (See Figure 5).

Now select the “Connect to Device” icon to initiate synchronization between Visual Studio and the HazPAC-R9 device. (See Figure 6).

Breakpoints
Setting breakpoints allows you to stop execution of your application at any point and examine the state of the application. A breakpoint may be set by selecting a line and pressing the "F9" hotkey. (See Figure 7).
To begin debugging the application, click the "Start Debugging" button. (See Figure 8).

Although you previously set up the target device, upon starting the first debug session, you will be prompted to select the device to deploy the application to. Select the "Windows CE Device" as was done earlier when selecting the target. (See Figure 9).

Once the application is deployed to the HazPAC-R9, it will begin execution. As soon as the first breakpoint is reached, execution will cease and you will gain full control over the running application. You may use the debugging options to continue execution, execute a single line, or execute multiple lines. You may view the status of each variable by either hovering over it with the cursor or by examining the windows at the bottom of Visual Studio just as you would with a desktop application. (See Figure 10).
**Watching Variables**
When program execution is halted due to a break point condition being met, the debugger will display the state of all local variables. In addition to those variables, class specific variables can be grouped together as a view to aid in debugging your application. This is accomplished by right clicking on a variable and selecting "Add Watch". Each addition appends a tab to the “Watch n” window where n is incremented for each variable added. (See Figure 11). Each watch window provides a convenient tree type structure for viewing hierarchical class variables.

![Figure 11. Watch view](image)

**Target Deployment and Execution**
After your application is built using Visual Studio, either a debug or release executable, it may be desirable to copy it onto the SDCARD or NAND Flash. This allows a means to store and execute the application without the need for connectivity to a host computer.

![Figure 12. Application Placement](image)

The HazPAC-R9 Runtime Image comes pre-loaded with a utility program called “SpringBoard”. This utility provides a solution for automatically running your applications at startup. Rather than copying your application files to `/Windows/Startup/` - which is in volatile memory - the executables should be copied to `/Storage Card/startup/` or `/nandflash/startup/`. (See Figure 12). After Windows CE runs, SpringBoard automatically starts applications located in the NAND Flash followed by applications in the Storage Card.
Conclusion

Windows CE can offer significant advantages in size, reliability and cost over standard Windows, and a RISC computing hardware platform makes the perfect host. Today’s RISC systems provide performance capabilities that can rival low-end x86 products while offering advantages in cost, power, and operating temperature range. Along with a well-supported software development environment, a RISC-based hardware solution running Windows CE can be an ideal solution for new product designs.