Windows Embedded

High Confidence Computing with the New Windows Embedded Compact 7

Windows Embedded Technical Article
October 2010

Applies to: Windows Embedded Compact 7

Summary: Windows® Embedded Compact 7 raises embedded development to a new level of high confidence computing. Supports for symmetric multiprocessing and higher RAM limits enable the operating system to take full advantage of higher performing systems now in the embedded arena. New tools now enable developers to quickly debug any issues that occur during development. The new features and tool in Compact 7 provide a new level of confidence and reliability.

Introduction

Building a reliable computer system is always important, but for embedded systems it is essential. Windows® Embedded Compact 7 provides a highly reliable foundation for today's modern embedded systems. Compact 7 is the latest version of Windows® Embedded CE, complete with an updated kernel, a new network stack, updated tools, and a variety of other new features.

Compact 7 builds on years of involvement in the embedded marketplace for Microsoft. Its purpose-built embedded operating system is designed for high reliability, high confidence computing. Along with the updated operating system, the Platform Builder for Windows Embedded Compact 7 tool set has also been improved to provide top notch support for development, debugging, and support.

The Improved Kernel

The improvements in Compact 7 start with an upgraded kernel. Compact 7 is the first version of the classic Windows Embedded CE product line to support symmetric multiprocessing (SMP). This allows the kernel to simultaneously use multiple CPU cores by spreading different threads across the various cores of the CPU.

To demonstrate the new multi-core abilities of the operating system, a CPU load application was developed that is similar to what is available on the desktop versions of Windows®. The following figure shows the System Monitor usage of the application on a computer running Compact 7 while Windows® Internet Explorer® Embedded loads a web page. Notice how the kernel takes advantage of each core on this multi-core system.

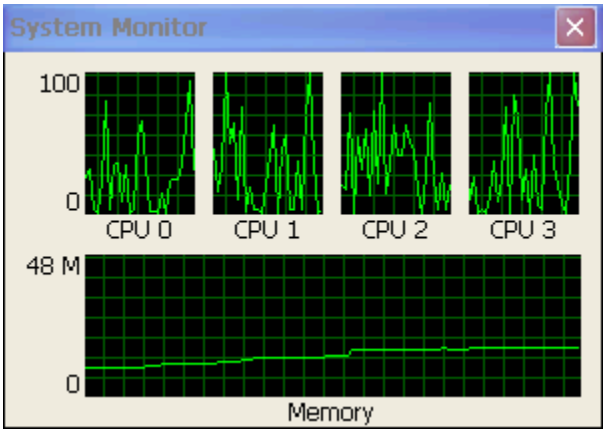

Figure 1

Aside from performance, the primary benefit of SMP is to prevent a single, runaway thread from severely affecting the global performance of the system. In the past, if a single thread did not block the system, even one running at normal application priority, it would noticeably affect the user interface (UI) and the performance of many other background threads.

In an SMP system, the runaway thread would consume the capacity of one of the cores, but the other cores in the CPU would remain available to the operating system for other applications. The following code sample provides a trivial example of such a nightmare scenario for an embedded system, in which an application runs a thread at a very high priority, but does not block the system.

```
//
// LockupApp.cpp : Defines the entry point for the application.
//
#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>
int _tmain(int argc, _TCHAR* argv[])
{
    // Set thread priority to highest
    CeSetThreadPriority (GetCurrentThread(), 0);

    // Infinite loop
    while (TRUE);

    return 0;
}
```

On a system with a single core, the above code would seriously degrade performance. In the following figure, the CPU System Monitor application displays the CPU usage of the Lockup application in this scenario. Notice how the load graph for CPU 1 is at 100 percent utilization. However, the other cores are still available. In this case, Internet Explorer runs as it downloads and displays a web page. Of course, the system is completely responsive to the UI and the other applications in the system, even though the lockup application is running.

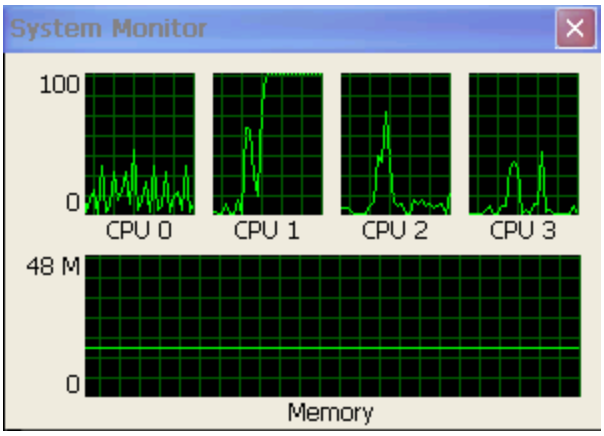

Figure 2

From a programming perspective, the addition of multi-core support does not affect single threaded applications. If interested, an application can query how many cores are available on a system using the function **CeGetTotalProcessors**.

While there is no need to worry about single threaded applications, multi-core systems can introduce consequences to multithreaded applications. Multithreaded applications not tested on multi-core systems may experience timing issues when their threads are truly executing simultaneously on separate cores instead of sharing a single core. To avoid these issues, if an application is compiled as an earlier version of the operating system, all the threads of the application will be assigned to the same core.

Applications compiled specifically for Windows Embedded Compact 7 are assumed to be tested for multi-core systems, and the application threads may be distributed across all the cores. If there is a need to manage specific thread assignments, the threads can be assigned to a core using the application programming interface (API) **CeSetThreadAffinity**. All the threads of a process can be restricted to a specific core using the function **CeSetProcessAffinity**.

Of course, there are the corresponding functions to query affinity, **CeGetThreadAffinity** and **CeGetProcessAffinity**. In addition, a new idle time function **GetIdleTimeEx** has been added that allows applications to query the idle time on a per-core basis.

In addition to managing processes and threads, the embedded engineer can manage the cores on the CPU. All but the primary core of the CPU can be powered down and then powered up as needed. The appropriately named APIs to manage the cores are **CePowerOffProcessor** and **CePowerOnProcessor**.

The addition of SMP support really improves the reliability of embedded systems. There is nothing like having a few extra CPU cores available to handle heavy load situations, as well as errant threads to calm the worried engineer. Simply put, multiple cores remove one of the primary causes of performance degradation on embedded systems.

Memory Manager Updates

SMP is not the only area of improvement in the kernel. RAM capacity is improved as well. The previous generations of Windows Embedded CE had a hard, architectural limit of 512 MB of physical RAM that the system could manage. Some systems used more, but the operating system had no knowledge of the additional RAM. These systems used the extra RAM as a simple input/output (I/O) buffer. The 512 MB limit came from the hardware architectures of the MIPS and SuperH-3 (SH3) processors that were the only CPU architectures Microsoft® Windows® CE 1.0 supported.

For a while, the RAM limit was not a big deal because Microsoft Windows® CE 5.0 provided only 32 MB of virtual space for each application. However, after Windows® Embedded CE 6.0 ripped away this virtual space limit by providing up to 2 GB of virtual space per application, the capabilities and footprint of some embedded applications increased to the point that some high end systems needed more than 512 MB of RAM.

Windows Embedded Compact 7 raises the supported RAM limit to 3 GB of physical RAM. Now, Compact 7 supports enough RAM to run hundreds of applications using as much RAM as they might need. The following figure shows a screen capture of the System Control Panel applet displaying a Compact 7 operating system running in a Windows Virtual PC with 2 GB of available RAM.

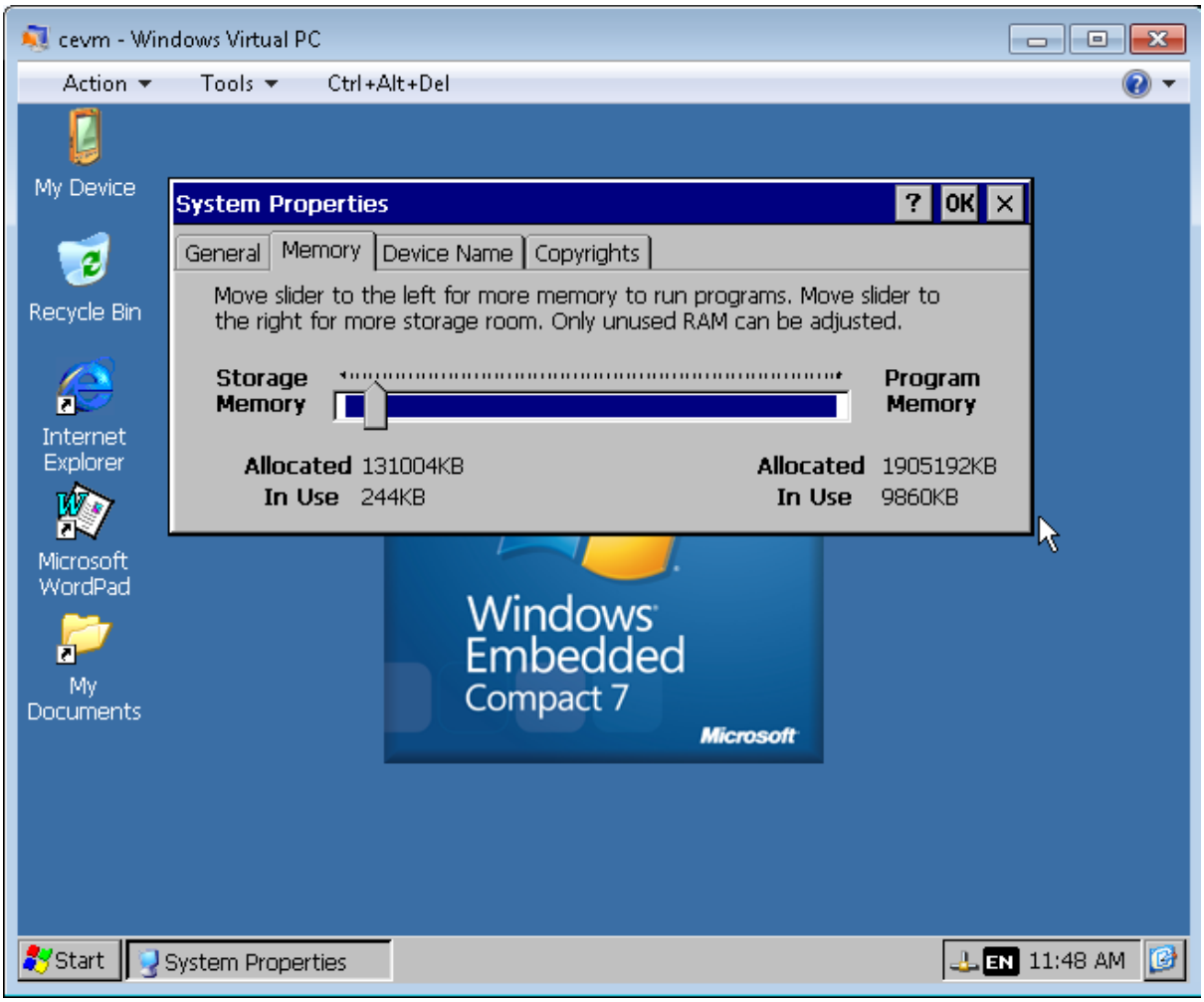

Figure 3

Those who are familiar with the internal workings of the operating system might wonder if supporting the additional RAM might break backward compatibility with older drivers.

Fortunately, the implementation of the new RAM support is optional. For systems that do not need more than 512 MB of RAM, the current design of using an *OEMAddressTable* to map the RAM into the system still works. So in this case, there is no need to change the Board Support Package (BSP).

For systems that need the additional RAM, the BSP needs to be configured to pass a special flag in the *OEMAddressTable*, and to provide two additional configuration tables: the *OEMRAMTable* to point to the additional RAM, and the *OEMDeviceTable* to map in any uncached hardware registers. In addition, using *VirtualCopy* to dynamically map in hardware still works in Windows Embedded Compact 7. The mappings can be designed to reduce or eliminate changes to the mapped virtual addresses of the hardware.

This additional RAM support is another way in which the operating system has been improved to increase stability for embedded systems. While the operating system does not require the additional RAM, some applications may need it. By supporting the additional RAM, Compact 7 avoids those tight memory situations that can be a problem in some systems.

One final improvement also helps in the memory area. Memory fragmentation can be a real problem for embedded systems that run for months or even years without a restart. Windows Embedded Compact 7 has a redesigned local heap manager that reduces fragmentation. This is one of the many improvements in Compact 7 that may not be visible, but still improves the reliability of the system.

Support for ARM v7, VFP, and NEON

Finally, the operating system has been recompiled to take advantage of the latest ARM architectures. Support for the ARM v7 opcode set along with NEON extensions has been added to Compact 7. NEON is a set of single instruction, multiple data (SIMD) instructions that can dramatically improve the performance of digital signal processing (DSP) and media decode routines.

ARM Vector Floating Point (VFP) support has been added to Compact 7 as well. This support is implemented at three levels: no VFP, VFP with applications calling the default C run-time (CRT), and VFP with applications using VFP operation codes directly. The VFP options are summarized in the following table.

Table 1. VFP Options Available in Compact 7

Scenario	Used when	Advantages
No VFP support	VFP is not supported by CPU.	Slow, but allows use of low cost CPUs without VFP support.
VFP and applications that call the CRT	Application compatibility needed with non-VFP systems.	10x performance improvement over systems with no VFP.
VFP with applications that use VFP directly	Ultimate performance is needed.	Best performance, smaller code size.

Compact 7 drops support for the ARMv4 architecture used by the classic, but no longer produced, StrongARM processor. However, Compact 7 still supports ARMv5 (used by the old XScale processor among others) as well as ARM v6.

Kernel Security Improvements

A common method of attack for malicious software is to guess the virtual address of code and data within an application. Windows Embedded Compact 7 protects against this type of attack by adding support for Address Space Location Randomization (ALSR). ALSR increases security by intentionally randomizing the load address of user mode dynamic-link libraries (DLLs) and other components within applications. ASLR is disabled by default, but it can be enabled by adding the following registry key.

```
[HKLM\init\BootVars]
 "AslrEnabled"=DWORD:1
```

From a build system perspective, simply setting the environment variable **IMGASLRENABLE** to **1** will add the ALSR registry value automatically during the build process.

Compact 7 also adds data execution prevention (DEP) on ARM 6 and ARM 7 systems. DEP prevents malicious code from injecting code into "data" pages, such has heap or stack pages and then jumping to and executing code on that page.

Communications Improvements

The kernel is not the only area of improvement in Compact 7. Communications is another important aspect of reliable computing that has improved in Compact 7.

Compact 7 now includes an updated browser called Internet Explorer Embedded. This web browser is based on Windows® Internet Explorer® 7 and includes some performance tweaks from Windows Internet Explorer® 8. For example, the scripting engine for Microsoft JScript® from Internet Explorer 8 has been back ported to Internet Explorer Embedded. iTracker support has also been included to better monitor for memory leaks.

Display performance has also been improved with an Image Caching service that provides quicker pan and zoom responsiveness. In addition, anti-alias font technology has been added for better font rendering. Aside from the performance improvements, upgrades have been added to improve web standards compatibility, including support for Extensible Hypertext Markup Language (XHTML) 1.0 with minor exceptions. This is a great upgrade from the Windows Internet Explorer 6-based browser in Windows Embedded CE 6.

The TCP/IP communications stack has also been upgraded. Windows Embedded CE 6 supported the classic Network Driver Interface Specification (NDIS) 5.1 standard network miniport driver, and the fairly outdated Winsock stack. Compact 7 upgrades the network driver to NDIS 6.1 and adds a newer network stack from the desktop versions of Windows.

This new stack provides greater flexibility to add to and extend the network stack with desktop-standard interfaces. In addition, the new stack has performance, security, and reliability improvements. From a developer perspective, the new stack has updated the Microsoft Win32® Internet (WinINet) API, and Layered Service Provider (LSP) programming interfaces.

High Confidence Tools

The improvements in Windows Embedded Compact 7 do not stop at the operating system. The Platform Builder for Compact 7 tool set supporting Compact 7 is also significantly improved. From the integrated development environment to the remote tools, as well as cool new tools to help with Microsoft Silverlight® integration, the tool set helps reduce potential bugs and increase the reliability of the final product.

For Compact 7, Platform Builder migrates from Microsoft Visual Studio® 2005 to Visual Studio® 2008. The integration is similar to the way it was done in Windows Embedded CE 6 with additional solution templates in the new project wizard, and additional menu and menu items added in the integrated development environment (IDE).

Build System Improvements

A new convenience of Compact 7 is the addition of "Checked" builds. Previously, Windows Embedded CE supported two build types, "Debug" and "Retail." *Debug* builds turn off compiler optimizations, insert asserts, debugging messages, and add arena checking for heaps. *Retail* builds enable optimizations, turn off the heap checking and disable asserts. Debug builds are great, but turning off optimizations greatly increases the size of the image. On some limited systems, debug builds can be too large for the hardware.

Checked builds enable compiler optimizations while keeping the debug messages, asserts and heap checking. This allows developers to get most of the benefits of a Debug build while still fitting the build into a small image size.

The compilers are also improved in Platform Builder to provide more efficient code and to take advantage of the new ARM architectures. The code produced by the new compilers is both faster and smaller than the code produced by the CE 6 compilers.

Platform Builder also contains an improved kernel debugger. This iteration of the debugger adds better conditional breakpoints, data breakpoints, and a better UI.

The Windows Embedded Silverlight Tool

Aside from the dramatic improvements to the kernel, the big news in Compact 7 is the maturation of Silverlight® for Windows Embedded. Silverlight for Windows Embedded is a revolutionary UI API that takes advantage of the power of the Silverlight presentation engine with the precision of a native, C++ powered code. While discussing most of the cool features in Silverlight for Embedded is beyond the scope of this white paper, there is one new feature that dramatically improves developer productivity and confidence.

For Compact 7, Microsoft added a new set of tools, aptly named Windows Embedded Silverlight Tools (WEST) that provides automated porting of Silverlight designs created in the Microsoft Expression Blend® tool into the native Silverlight for Embedded code. In earlier versions, developers had to hand port the Expression Blend code to C++. WEST removes this error-prone process with a simple menu selection in Platform Builder.

The new process allows full separation between the designer, who uses Expression Blend to create the UI, and the developer who uses Visual Studio to write, compile, and debug the code behind the UI. When installed, WEST adds a Silverlight for Windows Embedded application template to Expression Blend. The template narrows the available Silverlight controls to those supported by Windows Embedded Compact 7.

In addition, WEST can open an Expression Blend solution and auto-generate the C++ event handler code that corresponds to the events fired by the controls in the solutions. This is more than a simple one-off code generation. If the designer later updates the Blend project, the tool can be run again and the new handlers will be added to the C++ code.

WEST does more than simplify the link between the designer and developer. By automating the process in a very useful way, developers can now use the tool instead of trying to roll their own port of the back-end code.

This lowers the lines of code that the developer has to write, and therefore increases the reliability of the resulting Silverlight application.

Improved CTK

For Windows Embedded Compact 7, the Windows CE Embedded Test Kit (CETK) has been renamed the Compact Test Kit (CTK). Along with the new name comes a new interface for the PC-based integrated test harness. This new interface is a huge improvement over earlier versions. The new CTK provides a simple way to drive and monitor all aspects of the testing process. This interface is immediately apparent on the new home screen that is displayed in the following figure.

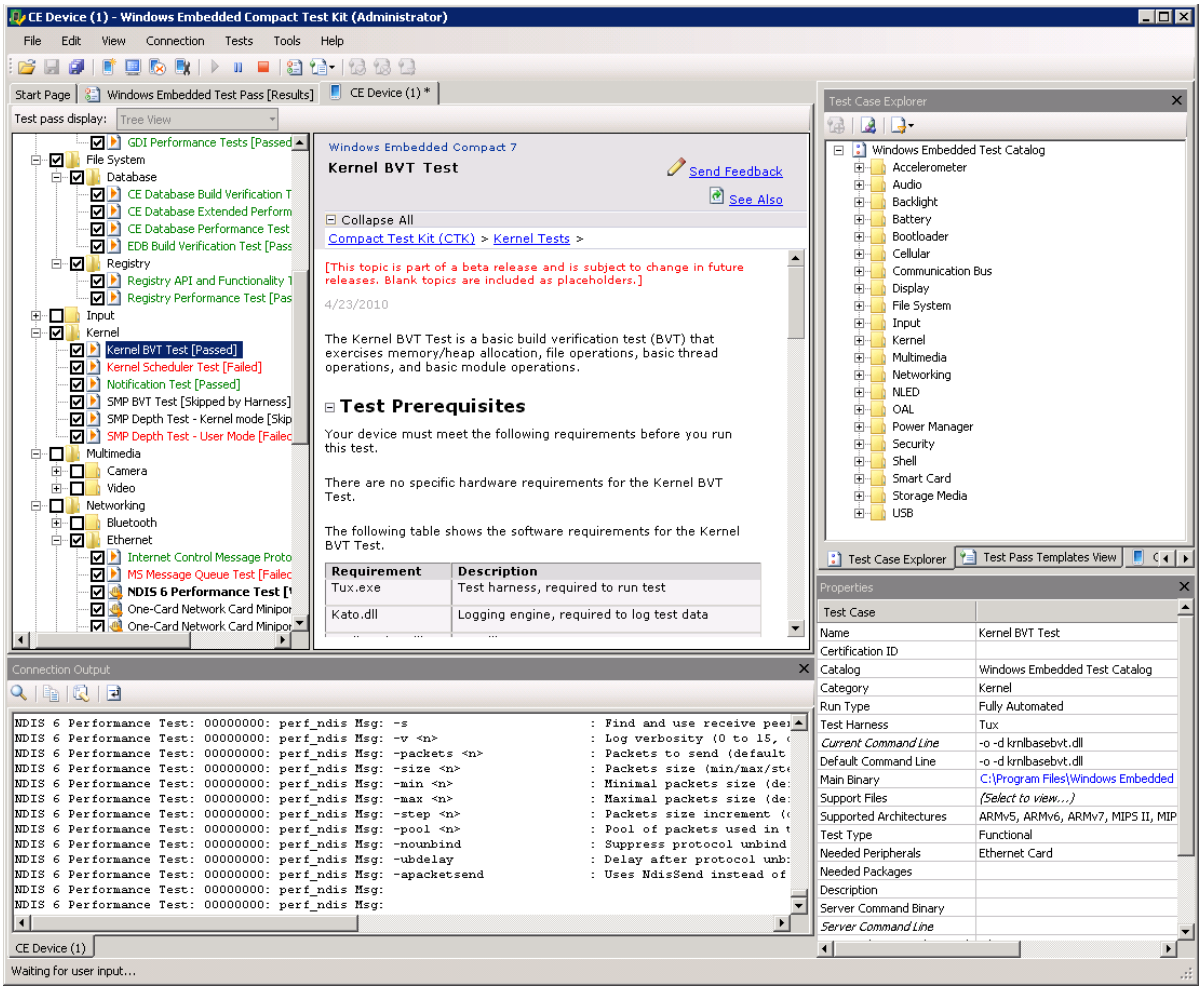

Figure 4

Instead of the bare look of the CETK window, the CTK provides an informative window that incorporates all aspects of the testing process in one place. The left side of the window displays the **Test Pass** window. This window displays tests to run and the results of each test run. The right side provides a **Test Case** window that displays the test tree, and below this window the properties of each test. Across the bottom, another window displays the real time output of the tests as they run. Finally, the middle of the screen displays a help screen with information about every test.

The CTK is bundled with test information for the following standard hardware:

- Accelerometer
- Audio
- Backlight
- Battery
- Bootloader
- Communication Bus
- Display
- File System
- Input
- Kernel
- Multimedia
- Networking
- NLED (Notification light-emitting diode)
- OAL (OEM Abstraction Layer)
- Power Manager
- Security
- Shell
- Smart Card
- Storage Media
- USB

Of course, in addition to the above tests, other custom tests can be added. Each individual test can be configured through the IDE to display the results in real time as the test runs.

Test results display in the **Test Pass** window marked as either **Pass**, **Fail**, or **Not Detected** if the hardware was not found. Hardware auto-detect can be disabled if desired. A new graphing tool also provides the results of the performance tests included in the CTK.

The new CTK now takes advantage of the CoreCon connection framework used by Platform Builder and the remote tools. This takes advantage of the current connectivity tools to eliminate the need to debug yet another connection path to a remote device. CoreCon is also extendable, so new connection methods can be added as needed. This new test environment, which is a huge improvement over the old CETK, is designed to enable test engineers and even common developers to better verify their devices.

Improved Remote Tools

For experienced Windows Embedded developers, the remote tools have been quite useful for interrogating target devices. The only problem with these remote tools is that they have not really changed in the last decade.

Windows Embedded Compact 7 changes this. The remote tools have been dramatically rewritten for this version of the operating system. Now, many of the tools are integrated into the Remote Tools Framework that provides a uniform interface. When starting the tools, a "Home Screen" appears (displayed below) that indicates possible actions for the developer.

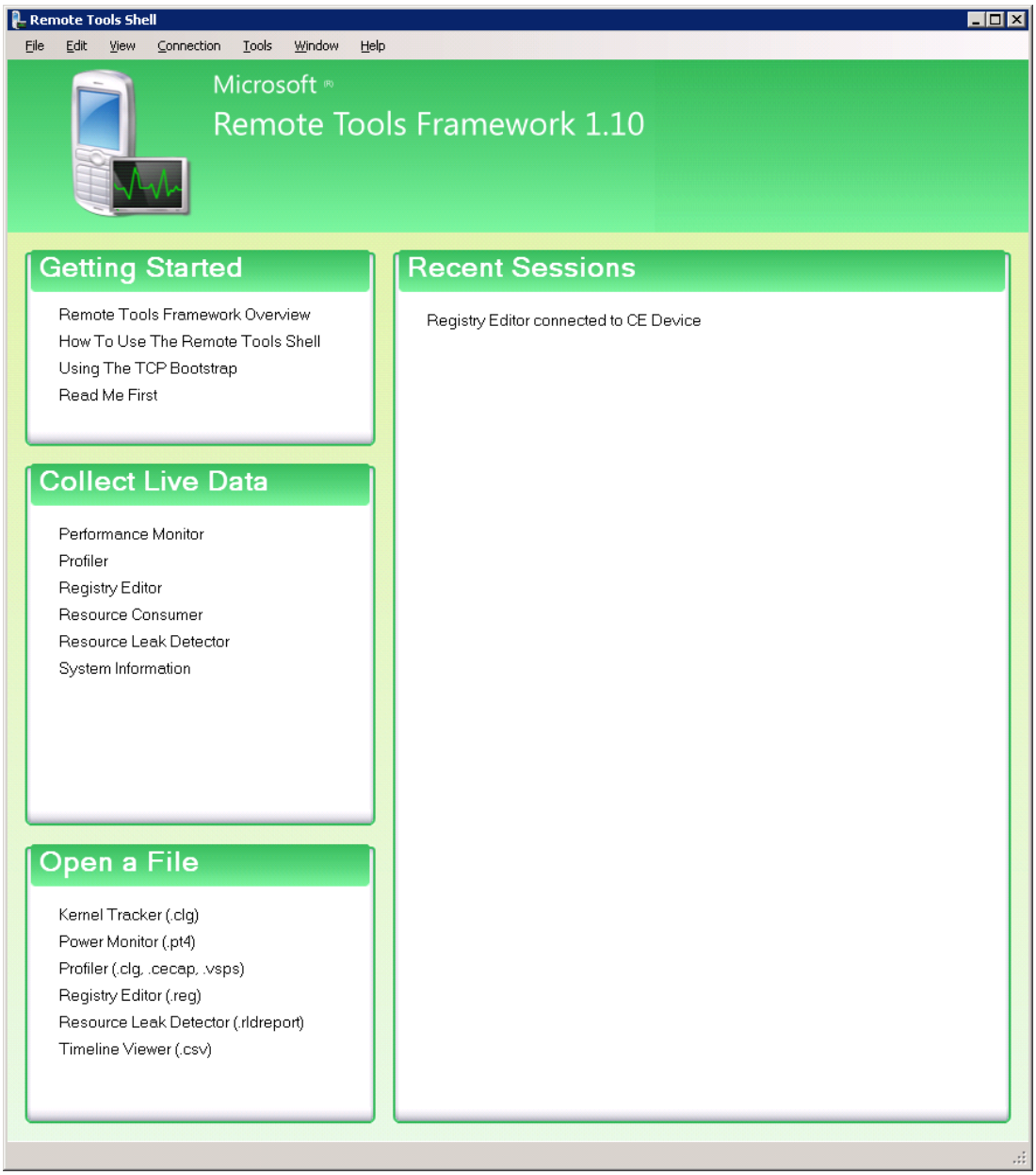

Figure 5

Each of the individual remote tools has also been improved as well. For example, the registry editor has a new state change tracking feature that monitors the registry for changes and then lists those changes. This new feature is a boon for developers who need to see registry changes that result from a given set of actions on the device. This new feature is quite useful. The following figure displays the registry change screen.

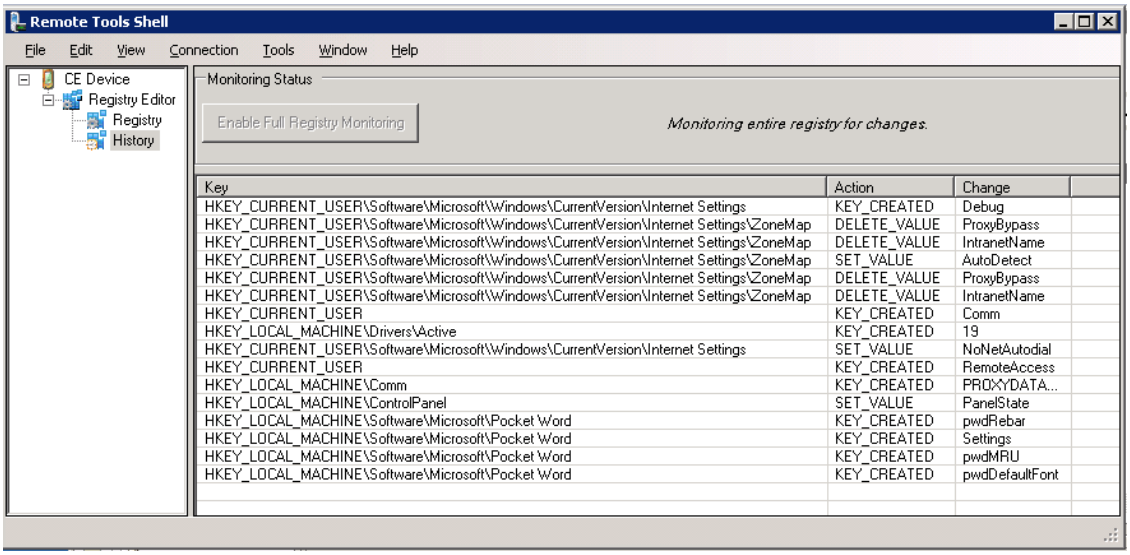

Figure 6

The performance monitor tool also has a new look. Like the performance monitor for the desktop, developers can use this tool to select a metric from a list of categories. Multiple items can be tracked in the tool. Analysis can be performed by time, or event, and the data can display in either table or graphical form as indicated in the following figure.

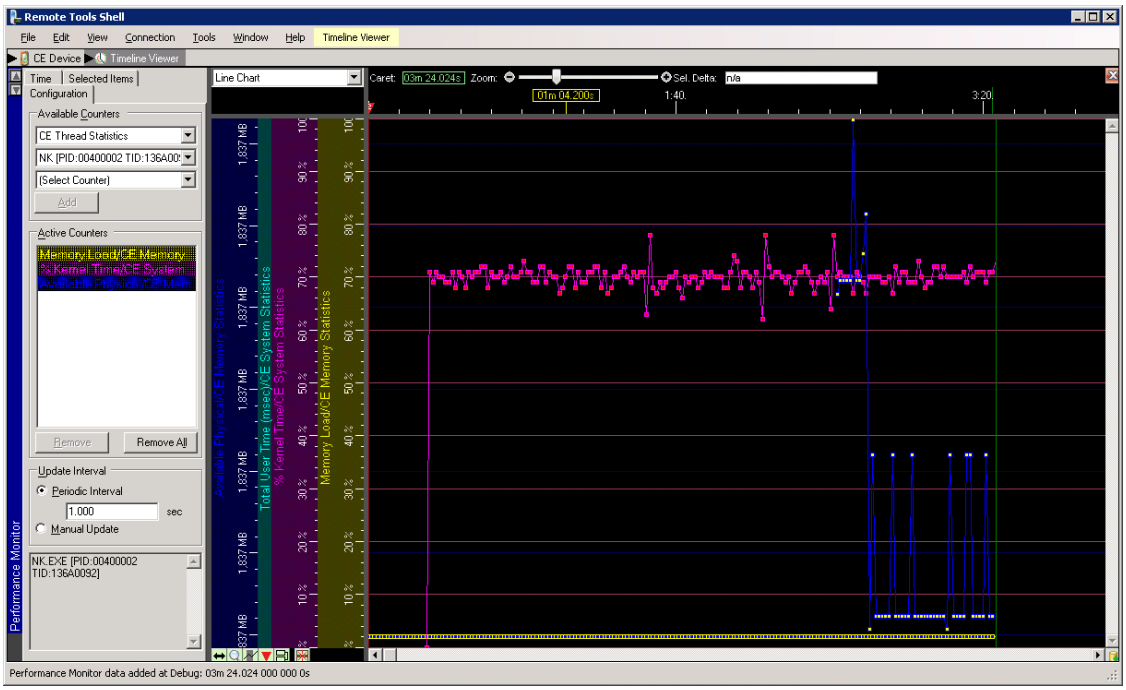

Figure 7

Among the remote tools, the tool with the most improvements is the Resource Leak Detector. This tool is a dramatically upgraded version of the old Application Verifier. The old tool was interesting and useful, but developers could be overwhelmed by information overload, and often missed leaks due to the massive amount of information that the older tool presented.

The Resource Leak Detector tool provides the information in table and graph form. The tool enables the developer to filter information by type, time, size, and other criteria. At any point, the developer can take a snapshot of the allocations, and then choose an individual allocation to see the call stack at the time of it allocation. The Resource Leak Detector is a welcome addition to the remote tool set. The following figure provides a snapshot of the Resource Leak Detector.

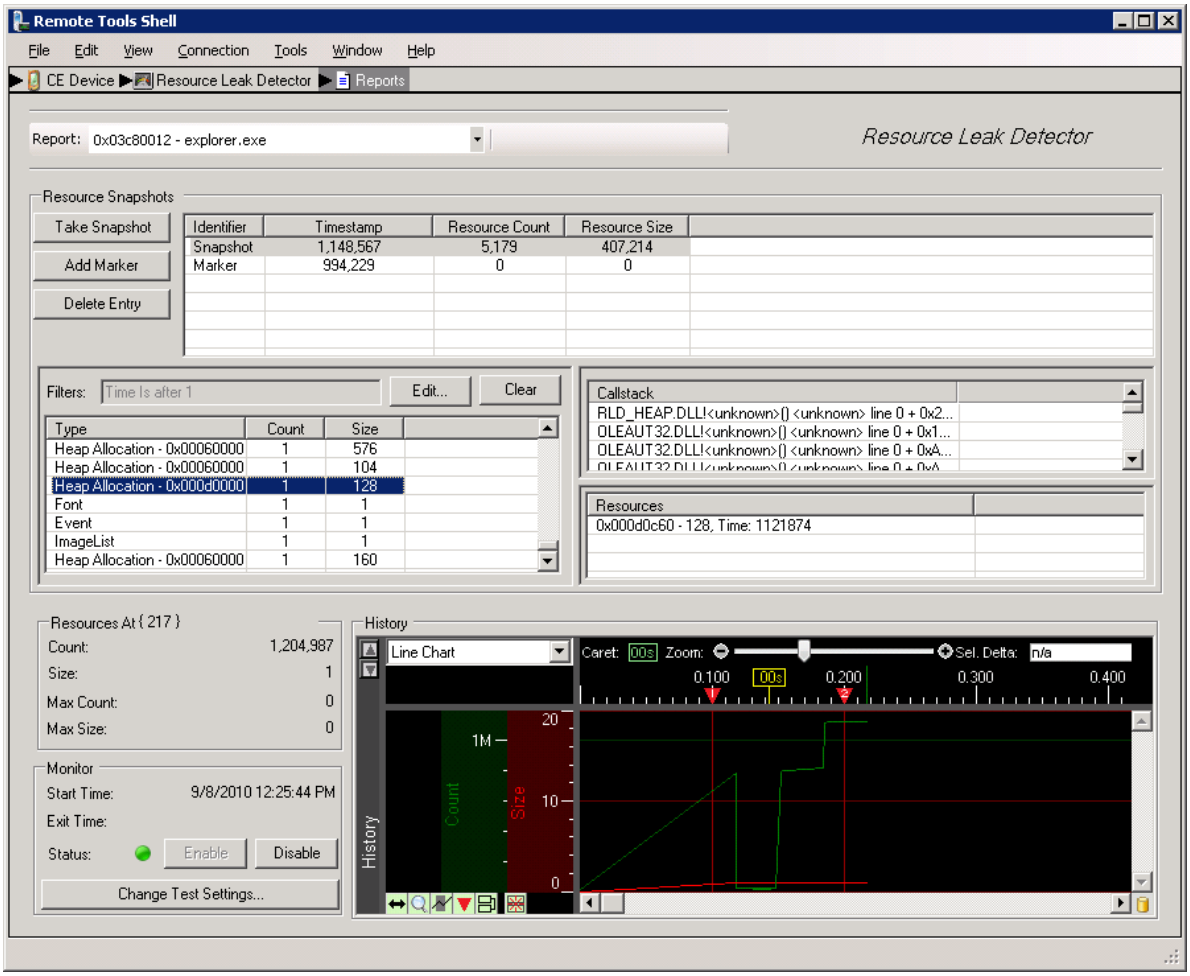

Figure 8

The upgraded remote tools dramatically improve the developer's ability to monitor their target devices. These new tools, along with the improved debugger and build system will definitely brighten the day of any embedded developer who moves to Windows Embedded Compact 7.

A few of the other remote tools are not integrated into the remote tools framework. The File Viewer, Heap Walker, Process Viewer, and ZoomIn tools are still available via a menu in Platform Builder. While not integrated in the remote tools framework, developers will be happy to see these old standards in Compact 7.

Conclusion

Windows Embedded Compact 7 raises embedded development to a new level of high confidence computing. Support for symmetric multiprocessing, and the higher limits for physical RAM, enable the operating system to take advantage of the higher performing systems that have moved into the embedded arena. The new tools are designed to enable developers to quickly debug any issues that occur during development. All in all, Compact 7 brings embedded computing to a new level of confidence and reliability.

For more information:

Windows Embedded web site: http://www.microsoft.com/windows/embedded/default.mspx

Copyright: