Applieddata.net
Embedded Computer Systems

White Paper

The Road From Desktop to Pocket
Converting NT Applications to Windows CE

Chris Tacke, Lawrence Ricci, Applied Data Systems

Abstract:

Now, with Intel supporting a powerful 32 bit XScale CPU family as well as the flagship Pentium line, more and more OEM's are looking at the possibility of migrating their NT/desktop applications over to portable, handheld devices using XScale. What exactly are the issues associated with this transition? Under what situations is code portable, and under what situations is a recode required? This paper will provide some basic, first-principles guidelines OEM's can use to estimate the work to convert.

Figure 1
The Phraselator represents a very demanding Windows 98 application that was ported to Intel RISC hardware. The system incorporates workstation level automatic speech recognition, and was completed (including the hardware and OS platform) in only 126 days. It is shown here in its original SLA (prototype) case which is how the initial, fully functional units were deployed to Afghanistan only 90 days after start of work. Next to it, the MidTech Legacy 2000 is a complex real-time 'precision farming' system which also required transfer of code from the NT/Laptop environment

Applied Data Systems
9140A Guilford Road
Columbia, Maryland 21046
Phone 301-490-4007 Fax 301-490-4582

Travel Where Gordon Moore Establishes the Speed Limit

For whatever reason, it seems Moore's law is driving hardware/CPU technology down the road faster than application software providers can keep up. Moving to a new platform usually means porting existing code as ground up application development is often too difficult. The 20 year smooth ride of the x86 CPU has, in a way, exacerbated the issue since generations of languages, libraries and other application enablers have been born, flourished and died within the x86 roadmap. The application programs built on these components are still vital and current, with practical application in new portable, ubiquitous computing environment supported by the XScale processor. What, exactly, are the pitfalls an OEM must avoid when planning to move an application across to the new XScale architecture?

Hundreds, even thousands of applications, have been moved over to the fully embedded operating systems supported by the XScale. This is particularity straightforward within the Microsoft family of operating systems; changing from Windows NT to Windows CE. While both OS's share common heritage, a common API and many other features, Windows CE was a ground-up recode so while the road from NT to CE is pretty broad and straight, there are a few potholes and even a roadblock and detour or two to either avoid or build into your trip plan.

Détour at DOS, CPM and Old Abandoned Roads

When planning the trip from a x86 application to Windows CE/RISC the first thing to do is check and see if the application runs on, or is dependent on elements of DOS, CPM, concurrent CPM or other legacy operating systems. The embedded space is where old OS's go to die, and there are still many DOS based systems out there embedded into machines, controllers, kiosks, etc. Sometimes, applications are dependent on DOS or other legacy OS just for boot up and initialization. For example, some real time systems running under windows are really DOS solutions booting in and running TSR "Terminate and Stay Resident" after they hand off to Windows. Indeed, until a recent release, Windows CE itself, in its x86 or "CEPC" variant was dependent on DOS to load. Just because a system looks like Windows does not mean all its code is Windows.

DOS and Windows users will see CE is a true 32 bit system, with new code from the ground up. Try as you like, you can not open a DOS window. Usually a DOS based application must be recoded to run on CE. You may have language commonality (C or C+; Delphi or other) but that would be as far as it goes. All the OS calls are different; and CE is a full multi-tasking, multi-programming environment that differs from DOS as a 2 Gigahtz Pentium differs from an 8086.

Fortunately, if the program can run on DOS it is probably pretty simple by today's application standards. Most DOS or CPM based systems required significant code to create a UI or network connection that might be trivial to recreate in the Windows platform, especially a multi-tasking CE platform that allows modular system development. At the end the OEM can create a great UI, with robust network and real-

time features. In other words, the functional recode required for DOS is often the purpose for moving the application, not a non-value added 'roadblock'

Rest Area at Floating Point Calculations

Before getting on the road to XScale the engineer needs to understand the calculations in his application a bit. If he is heavily dependent on floating point instructions, there may be an issue. XScale comes from the world of low-power RISC, and RISC is typically fixed point math; high wattage hardware floating point is just not used. That said, XScale is fast, 400Mhz, and Intel offers IPP (Integrated Performance Primitives) to help meet most applications.

The IPP are a set of carefully coded low level routines that use algorithms optimized for the particular Intel CPU family. While full floating point emulation is available, the pragmatic solution is typically to adopt a scaling convention to keep variables within the range of 32 bit integers. Integer math is fast and accurate for addition and subtraction, but integer multiplies (A x B = C) can easily lead to an 'off scale' value for C and integer divides (A / B = C) can give rounded, inaccurate values for C. Intel's IPP's offer a easy way to 'scale' floating point values to a fixed point representation, with a certain number of bits for 'before' and 'after' the decimal point (e.g. binary point or 'radix'). This technique will typically be more than 20 times faster that a typical floating point library, and comparable to floating point hardware. Since the resolution/range of 32 bit integers is very great (one part in four billion) IPP based scaled mathematics will work very well for most problems not dealing with cosmology.

Function Group	Itanium Architechture	IA-32 (includes Pentium 4 Processor)	Intel StrongARM Microarchitecture	Intel XScale Microarchitecture
Signal Processing	now	now	now	Now
Image Processing	now	now	now	
JPEG	now	now	now	Now
Speech Recognition	now	now		
Computer Vision	now	now		
Audio Codecs	V2.0	V2.0	now	Now
Video Codecs	V2.0	V2.0	now	Now
Matrix	V2.0	V2.0		
Vector Math	V2.0	V2.0		
Speech Coding	future	future	now	Now

Figure 2- Intel Integrated Performance Primitives
The Above Table Indicates Status of The Integrated Performance Premiums. Notice that in absence of dedicated and power-hungry hardware for audio and video in the RISC world, Intel has encouraged the use of IPP's (source: Intel™ Integrated Performance Primitives (IPP); August 24, 2001).

Slowdown! Dangerous Curves at Graphics

Advanced video-game level graphics are applications where the typical Pentium desktop has a real edge. High performance "game" oriented graphics cards have no equivalent in the RISC space. Applications using Direct Draw and Intel's MMX would have to be recoded to run on CE, and then would have much lower relative performance.

XScale and StrongARM are fine for a standard "WIMP" (Windows, Icons, Menus, Pointers) interface Windows CE. These CPU's can directly support VGA screens up to 16bit color depth provided the driver is efficient and optimized. You can even do

SVGA/24 bit color or more with a good platform. But for an inefficient 'reference platform' quality of driver there might be flicker at VGA, limiting use to PocketPC sized QVGA screens and smaller. But with a good driver, VGA is fine.

Past the simple WIMP interface, high performance games and multi-media features require even more attention. Desktop Pentiums today are running full screen, 24 bit color depth, full motion video. Embedded systems such as a PocketPC are in comparison limited, typically providing video in "QCIF" format (144 lines and 176 pixels per line). The practical limits for the typical industrial OEM are even more stringent.

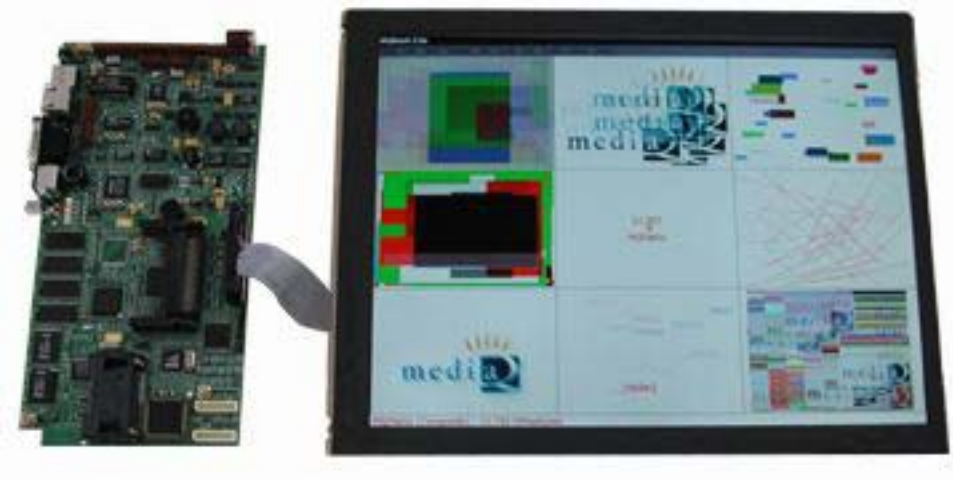

Figure 3 Small System; Big Graphics
This system incorporates a RISC CPU plus a powerful graphics coprocessor to deliver PC level graphics in a low-power, single board design's

The PocketPC achieves its multi-media performance by exploitation of the Intel Performance Primitives within the GAPI (Games API). Now, here is the problem: GAPI is a value-added part of the Pocket PC license, NOT the general Windows CE license as used by most OEM's. Bottom line, an 'out of the box and typical' CE/RISC solution will run multimedia and graphics much slower than a normal desktop.

To mitigate this issue, Applied Data Systems is providing XScale and StrongARM systems with additional Graphics processors, providing richer and faster graphics that meet most application needs. But costs exist and the OEM planning a recode needs to assess his need for high performance graphics in the embedded space. Special co-processor designs like the ADS Display Master can get CE/embedded graphics well into the PC range with XGA, 24 bit color depth and thousands of polygons per second. Likewise, special purpose media players based on CE abound. But the OEM should recognize that this is the kind of graphics performance then is designed into the hardware, OS and application, it just does not 'happen' with a simple re-compile or even recode.

Yield to Data Access /Data Store

If the legacy system to be recoded was disk based, it should be clear that some level of recode may be necessary to move the application to an embedded, Flash-based CE platform. While disk file read/writes to disk can look very much like file read/writes to Flash memory, the timing is different (Flash can be slower) and there are issues with wear on Flash chips. Whenever disk storage was part of the application, a short review of the application architecture is warranted, perhaps distributing some data into

memory-resident arrays, or locating certain programs in flash and even running them from flash directly.

Data formats need to be examined as well. If the application uses SQL or OLE-DB, then the transfer to the Windows CE space should be easy since both are well supported. But for other formats, such as ODBC there is an issue. It is interesting to note that the for the 'forward looking' formats of XML and SOAP CE is very parallel, or in fact maybe a bit ahead, of Microsoft Desktop operating systems.

Finally, in the embedded world most databases are distributed. This is definitely an application design issue. Intermittent connection, reconnection and syncing must be understood. Fortunately, Microsoft has provided MSMQ (Microsoft Message Queue), Active Sync and other tools to assist in this effort. Moving a database from a monolithic system or LAN to portable or WAN or intermittently connected devices is at some level a recode, but the tools are there and the path is well trodden.

Your Application May Have Troublesome Passengers

Many legacy applications bring with them 'hitchhikers' in terms of third-party sourced, embedded software components. These can include I/O Drivers, USB Device Drivers, DLL's, COM Objects, ActiveX Controls and Subroutine Libraries. Moving the application from a Desktop to an XScale system will require (at least) re-compiling these objects for the new CPU. What ran on x86 may not play on the ARM instruction set. This implies at least access to the source code of these objects, and probably support from the supplier.

Of course, the library of available third party software in the CE space is constantly increasing, especially with the advent of C# which complies to a universal and common "common language" CL that is shared by such programming environments as FORTRAN.NET and COBOL.NET. The engineer might find it preferable to secure the component he needs in the target domain, and recode to the new format.

CE is not NT; That is the Question

CE was designed to be a modular, compact system and as such makes no effort to recreate certain NT features, for example NT's multi-user file and data access controls. To the extent your embedded device needs extensive security administration, you will have to build it yourself.

System event logs are also lacking; again for this feature you will typically have to 'do it yourself' by embedded logging features into the application and passing the needed record to some sort of home brew log manager for storage.

Some NT Applications may be configured as NT Services; there is no analog in the CE environment. This is usually not a problem as the application can be configured as a driver, a DLL, an application in the startup folder or linked into a custom 'shell'.

One commonly used NT Feature used to be an issue but is fine now. For a long time, Windows CE supported the Windows API but not the MFC's (Microsoft Foundation

Class) modules that were employed by so many applications. Now, with CE 4.1, this is not an issue. The MFC library is quite complete and many application programs can be brought over straightaway.

In general; the user will find CE has what it needs. The features brought over from NT were selected to be appropriate for use in embedded systems. If some NT feature is vitally important to the application, there are probably good reasons why embedded XP, not CE is a good choice.

The VB Surprise: Novice Programmers Frozen in the Headlamps

Interestingly enough, Microsoft's move to the "Object Oriented" .NET programming environment changed the world of embedded Visual Basic quite a bit. Until now, Visual Basic was a functional, but simple, programming language with limited data types and very free structure. A lot of inelegant, but functional code has been written. With the transition to .NET, VB has been moved to center stage for the serious programmer. The toolset has been updated, and VB programs share the same CLI with C# and the other .NET languages. All this sounds good, but the problem is VB code written for the old, unstructured world will not run. VB in the new, .NET platform is intensely object oriented with strongly typed data and enforced OO structure.

So, for the case of a legacy VB program, the engineer has the choice of either a recode, or perhaps a simpler transfer to an older version of Windows CE (like CE 3.0). While CE 3.0 is well supported for the older StrongARM chip, good CE 3.0 OS ports to XScale are harder to find.

Application Hardware: Mixed Traffic On This Highway

Embedded, 'ubiquitous' computers come in all shapes and sizes, and the steering wheel is not always on the left. When engineers start working in the true embedded space, they soon realize the true limits of the 'write once, run anywhere' mantra. A comprehensive program with graphic UI that might run on a PC will not typically run on a pocket MP3 player- the hardware architectures are just too far apart. All desktop PC's are more or less the same, the configuration is limited to a few standard BIOS's and options that can be sensed and configured for the OS at install time. But the embedded space is a hardware free for all. Displays can be any color depth from 1 to 48, and screens can be any pixel configuration from QCIF and below to wall size. Embedded systems like the ADS Graphics Master might support 7 serial ports, compared to the standard 4 on the PC. All sorts of networks, including multiple Ethernet networks, may spring from a single board embedded computer. Typically, the application needs to be configured for all of these options.

Power For the Highway

Perhaps the most important change to the embedded application environment comes with battery powered devices. The RISC world of XScale is dominated by the concept of

power management, getting the maximum MIPS from the minimum Watts. The desktop application designer seldom thinks of his computers power consumption. The handheld application designer seldom thinks of anything else. Power miser features shape the application space. The computer has 'on' and 'off' modes for sure, with the expected initialization routines, but in addition XScale has Turbo, Run and several reduced power modes including a deep 'sleep' that lets the CPU and memory go into a very low power state, but spring to action quickly.

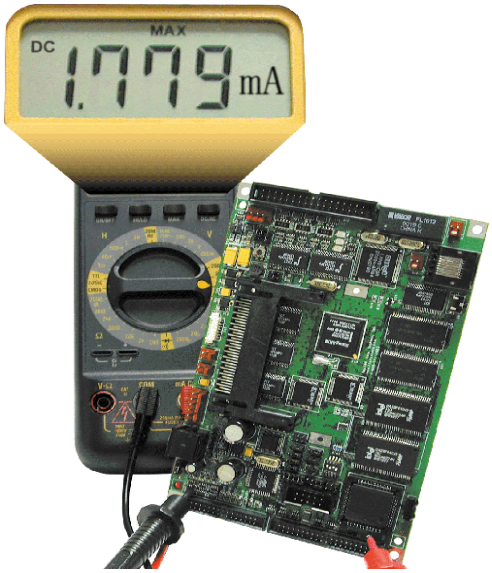

Figure 4- The Tools
In the Embedded Space programs must sometimes be tested with a Meter

Not only the CPU requires power management. Systems are now designed with power-partitioning, literally turning on and off peripheral devices on an as-needed (application-level) basis. The application code may find itself switching Flatpanel's backlights on and off, powering up a USB controller, or might become involved with battery charging protocols. Even systems that are powered off the AC line, such as automation and Kiosk systems, contain powerfail-restart sequences that are a complete, and vital, subset of application design seldom considered on the desktop.

We have often become jaded to the 'logical' way embedded systems like televisions and phone answering machines restart after power failure. The only reason we can forget the problem is engineers have done their job well. Power management has to be designed, and coded in. No cross complier can do it for you.

Driving 24 by 7- It Can be Slippery When Wet

There is one more, very important application consideration when an embedded application is intended to run 24 x 7. Non stop applications are subject to memory leaks. A memory leak occurs when a program reserves a piece of memory, and then fails to release it for general use. Gradually, this 'garbage' can accumulate, and finally the system will fail to find sufficient scratchpad and will crash.

For a desktop or even PocketPC application that is frequently started and stopped, leaks may never be visible. The system will typically flush them all out each time the application exits. But in a 24x7 application, no matter how small the leak, eventually it will crash the system.

Leaks are pernicious and ubiquitous. They can be found even in trusted drivers for name brand modem cards from well respected companies with hundreds of thousands of units installed. In fact, at this writing ADS has tested six PCMCIA modem/communication cards, all from first tier suppliers, and found three of them leaked. The simple fact is that even with tens of thousand of units sold, very few, or maybe none of these cards, were in true 24x7 operation.

When you realize that these leaks occur 'on the other side of the compiler' you can appreciate how focused a programmer must be to avoid them. In fact, entire programming environments such as Java and C# have been developed with 'automatic

garbage collection'. For a class of application, these languages make leak-proof code easier, but at a cost. Since the system periodically surrenders control to a 'garbage collection' process, these languages are not useful for deterministic real time applications; unfortunate since often embedded systems are exceed to be deterministic and real time. For Real Time, 24x7 systems the programmer is left between a rock and a hard place with only hard work and attention to detail as a solution.

Good Trip Planning Requires a Guide

Sound Scary? Don't worry about it; it is not so hard if you travel with someone who has been down the road before. Work with a platform provider with a proven, application ready interface. Make a few tests, check you base code and write a bit more on the new environment. Bottom line is some applications should stay on a Pentium Microsoft embedded NT might be the right OS. But for low cost hardware, battery management, portability and non-PC architectures, XScale and Windows CE is often the best solution. Every year Applied Data Systems helps hundreds of engineers transfer their application from a x86/NT code base to RISC/Windows CE, very few fail and many manage to port their code only a few weeks.