Building Windows for the ARM processor architecture

Steven Sinofsky | 2012-02-09T10:00:00+00:00

One of the notable aspects of Microsoft Windows has been the flexibility the architecture has shown through shifts in technology and expansion of customer usage over time. What started out as an operating system for one person working solo with productivity software is now the foundation of a wide array of hardware and software technologies, a spectrum of connected Windows products, and an incredibly flexible approach to computing. With *Windows 8*, we have reimagined Windows from the chipset to the experience—and bringing this reimagined Windows to the ARM® processor architecture is a significant part of this innovation. Expanding the view of the PC to cover a much wider range of form factors and designs than some think of today is an important part of these efforts. Windows on ARM enables creativity in PC design that, in combination with newly architected features of the Windows OS, will bring to customers new, no-compromise PCs.

This post is about the technical foundation of what we call, for the purposes of this post, *Windows on ARM*, or *WOA*. WOA is a new member of the Windows family, much like Windows Server, Windows Embedded, or Windows Phone. As with those products, WOA builds on the foundation of Windows, has a very high degree of commonality and very significant shared code with Windows 8, and will be developed for, sold, and supported as part of the largest computing ecosystem in the world. Today we'll focus on the development of WOA and introduce some of the features, along with how customers will experience it. As with x86/64 Windows 8, there are still announcements to be made relative to the business and marketing aspects of the product(s). Today's blog post is about making WOA, not marketing or selling it.

At the same time, while this post is exclusively on our work on WOA, we have had a deeper level of collaboration with Intel and AMD on the full breadth of PC offerings than in any past release. Windows 8 innovations on powerful and richly capable x86/64 processors, and work on new low-power processors such as those that Intel demonstrated at CES, require an equally strong commitment, even larger engineering investment, robust new designs, and improved architecture for Windows across these platforms. While discussing our engineering for ARM processors, it is important to keep in mind that in addition to all of the new work for the ARM platform we have done, much of the work discussed in this post applies directly to the x86/64 platform and Windows 8 as well. We could not be more excited or supportive of the new products from Intel and AMD that will be part of Windows 8—across a full spectrum of PC form factors including tablet, notebook, Ultrabook™, all-in-one, desktop, and more that all take advantage of the new capabilities of Windows 8 while Windows 8 takes advantage of new features in hardware.

Using WOA "out of the box" will feel just like using Windows 8 on x86/64. You will sign in the same way. You will start and launch apps the same way. You will use the new Windows Store the same way. You will have access to the intrinsic capabilities of Windows, from the new Start screen and Metro style apps and Internet Explorer, to peripherals, and if you wish, the Windows desktop with tools like Windows File Explorer and desktop Internet Explorer. It will have the same fast and fluid experience. In other words, we've designed WOA to look and feel just like you would expect. WOA enables creativity in PC design that, in combination with newly architected features of the OS, will bring to customers new *no-compromise* experiences.

As an in-depth engineering dialog, we tend to favor the long form for *Building Windows 8* posts, and this post is no exception. It does seem like a good idea to first provide a summary of the important items we are going to cover in detail in this post:

- **Windows on ARM, or WOA, is a new member of the Windows family that builds on the foundation of Windows**, has a very high degree of commonality and very significant shared code with Windows 8, and will be developed for, sold, and supported as a part of the largest computing ecosystem in the world. We created WOA to enable a new class of PC with unique capabilities and form factors, supported by a new set of partners that expand the ecosystem of which Windows is part.
- **WOA PCs are still under development and our collective goal is for PC makers to ship them the same time as PCs designed for Windows 8 on x86/64**. These PCs will be built on unique and innovative hardware platforms provided by NVIDIA, Qualcomm, and Texas Instruments, with a common Windows on ARM OS foundation—all running the same Windows OS binaries, a unique approach for the industry. PC manufacturers are hard at work on PCs designed from the ground up to be great and exclusively for WOA.
- **Metro style apps in the Windows Store can support both WOA and Windows 8 on x86/64.** Developers wishing to target WOA do so by writing applications for the WinRT (Windows APIs for building Metro style apps) using the new Visual Studio 11 tools in a variety of languages, including C#/VB/XAML and Jscript/ HTML5. Native code targeting WinRT is also supported using C and C++, which can be targeted across architectures and distributed through the Windows Store. WOA does not support running, emulating, or porting existing x86/64 desktop apps. Code that uses only system or OS services from WinRT can be used within an app and distributed through the Windows Store for both WOA and x86/64. Consumers obtain all software, including device drivers, through the Windows Store and Microsoft Update or Windows Update.
- **WOA can support all new Metro style apps, including apps from Microsoft for mail, calendaring, contacts, photos, and storage. WOA also includes industry-leading support for hardware-accelerated HTML5 with Internet Explorer 10.** WOA will provide support for other industry-standard media formats, including those with hardware acceleration and offloading computation, and industry-standard document formats. In all cases, Microsoft seeks to lead in end-user choice and control of what apps to use and what formats to support.
- **WOA includes desktop versions of the new Microsoft Word, Excel, PowerPoint, and OneNote.** These new Office applications, codenamed "Office 15", have been significantly architected for both touch and minimized power/resource consumption, while also being fully-featured for consumers and providing complete document compatibility. **WOA supports the Windows desktop experience including File Explorer, Internet Explorer 10 for the desktop, and most other intrinsic Windows desktop features**—which have been significantly architected for both touch and minimized power/resource consumption.
- **With WOA you can look forward to integrated, end-to-end products—hardware, firmware and WOA software, all built from the ground up.** Building WOA has been an ongoing engineering effort involving Microsoft, ARM licensees, PC makers, and developers of components and peripherals. These efforts spanned a wide array of subsystems that have been newly created or substantially re-architected for WOA. Partners will provide WOA PCs as integrated, end-to-end products that include hardware, firmware, and Windows on ARM software. Windows on ARM software will not be sold or distributed independent of a new WOA PC, just as you would expect from a consumer electronics device that relies on unique and integrated pairings of hardware and software. Over the useful lifetime of the PC, the provided software will be serviced and improved.
- **Around the next milestone release of Windows 8 on x86/64, a limited number of test PCs will be made available to developers and hardware partners in a closed, invitation-only program.** These devices will be running the same branch of Windows 8 on x86/64 as we release broadly at that time. These are not samples or hints of forthcoming PCs, but tools for hardware and software engineers running WOA-specific hardware.
- **The Windows Consumer Preview, the beta of Windows 8 on x86/64, will be available for download by the end of February**. This next milestone of Windows 8 will be available in several languages and is open for anyone to download.

This post is organized with the following sections: Working with partners, Providing apps, Engineering for ARM (which will go through the various subsystems), Developing for ARM, Delivering WOA PCs, and finally, Next steps.

We've also prepared a short video demonstrating WOA as described in the post.

Your browser doesn't support HTML5 video.
Download this video to view it in your favorite media player:
High quality MP4 | Lower quality MP4

Working with partners

Developing WOA begins as a partnership with companies that make ARM processors and package them together with the subsystems required to deliver the equivalent of a *motherboard.* Unlike the boards many are familiar with, you can think of a WOA board as a silicon package—a series of silicon layers bound together in an incredibly small form factor, called a *System on Chip* or *SoC*.

CES 2011 demonstration showing a System on Chip (SoC). (Julie Jacobson, Associated Press)

Each ARM licensee building these packages takes a different approach to selecting features, making product trade-offs, and designing the complete silicon package. These choices are what bring the diversity of different products built on ARM to the market. There is no single ARM experience, and as we have seen with other operating systems, even the same ARM CPU combined with different components, drivers, and software can yield different types or qualities of experiences. That is why from the start of the WOA project, we have been working with three ARM licensees: NVIDIA, Qualcomm, and Texas Instruments. Each brings different expertise and different approaches, and all will make a unique contribution to WOA. All of them have extremely successful ARM-based products in the market today, from tablets to smart phones to e-readers to embedded devices. We are fortunate to have the support of such amazing partners, and WOA is unique in working with such diversity.

A SoC package by itself is just the beginning. Delivering WOA PCs is a partnership with PC manufacturers who bring their expertise in manufacturing, system engineering, and industrial design and combine that with the engineering work of ARM partners to develop a complete PC. PC makers also bring expertise at selling PCs to consumers and businesses through a variety of channels and supporting those purchases over time.

Microsoft's role in this partnership is to deliver a Windows operating system that is tuned to this new type of hardware, new scenarios, and new engineering challenges. Our goal is to make sure that a reimagined Windows delivers a seamless experience from the chipset through firmware, through hardware, through the OS, through applications, and ultimately to the person interacting with the PC. This is a new level of involvement that brings with it a new level of engineering work across all of the parties involved. This new approach is about delivering a unique combination of choice, experiences, and a reliable end-to-end experience over the life of the PC.

Providing apps

Before we dive into the details of the changes to the underlying implementation of Windows to support ARM hardware, let's start at the top and talk about what apps Microsoft will offer (we're all happy to see a renewed use of the term "apps"—my first business card at Microsoft in 1989 read *Software Design Engineer in Apps Tools, Apps Division.*)

We have not yet announced the editions or SKUs we will have for any new Windows products, and during the pre-release phase we include almost all features in the software as a way of testing and exercising the full surface area of the product. You can expect that we will adjust the features included with the single WOA offering such that it is competitive in the marketplace and offers a compelling value proposition to customers of all types.

As we announced and demonstrated at //build/ and other forums, WOA has all the WinRT capabilities present in the Windows Developer Preview, and all the tools and techniques that you can use to build new Metro style apps for x86/64 are available to developers to also target WOA. Developers can use our tools to create native C/C++ code for maximal performance and flexibility, in addition to the C#, XAML, VB, and HTML5 based tools, to target apps for WOA, so long as their code targets the WinRT API set. Additionally, developers with existing code, whether in C, C++, C#, Visual Basic, or JavaScript, are free to incorporate that code into their apps, so long as it targets the WinRT API set for Windows services. The Windows Store can carry, distribute, and service both the ARM and x86/64 implementations of apps (should there be native code in the app requiring two distributions).

We have also previously demonstrated Microsoft's new Metro style apps for connecting to cloud-based services like Hotmail, SkyDrive, Messenger, and— through those services—a wide variety of third parties. For example, our mail app connects to industry-standard EAS, which covers an array of enterprise and consumer-based mail, calendaring, contacts. With existing Live Connect capabilities, you can chat with your Facebook friends, or keep up-to-date on your LinkedIn or Twitter feeds all in a Metro style app—these are just a couple of examples of over 100 different services globally that you can connect to your Microsoft account. These apps are provided with WOA, but of course, people can remove these, set different defaults, or use the Windows Store to get similar apps from third parties. In addition, any Metro style app in the Windows Store can work with any service it chooses, with or without using any Microsoft services—and this spans the range from sign-in, communications, in-app payments, to advertising services.

In the next pre-release of Windows 8 you will also see Metro style apps available from Microsoft that support a wide variety of industry-standard media and document formats, along with Internet Explorer 10 which supports the standard HTML5 web platform. We believe that the level of standards support provided in WOA is among the best in class, and comparable in scope to competitive products. And of course, our intent is to lead in the industry in providing end-user choice and control over the apps on your system and what you choose to run.

The availability of the Windows desktop is an important part of WOA. The desktop offers you a familiar place to interact with PCs, particularly files, storage, and networking, as well as a range of peripherals. You can use Windows Explorer, for example, to connect to external storage devices, transfer and manage files from a network share, or use multiple displays, and do all of this with or without an attached keyboard and mouse—your choice. This is all familiar, fast, efficient, and useful. You'll have access to a deep array of control panel settings to customize and access a finer-grained level of control over your system, should you want to. And if you've used the Developer Preview with a touch-capable PC, you know that the desktop user-interface has been refined for touch interaction with improved user-interface affordances.

At the same time, WOA (as with Windows 8) is designed so that customers focused on Metro style apps don't need to spend time in the desktop. Availability of the desktop incurs no runtime overhead. It is just there should you want or need it. Below, we will describe the technology behind the scenes that goes into making sure that the availability of the desktop does not compromise system security, reliability over time, performance, or power consumption of a WOA PC. To those of you who've tried out the Developer Preview, you'll notice that the user experience has continued to evolve and you will see a broad set of improvements in the upcoming Consumer Preview.

Some have suggested we might remove the desktop from WOA in an effort to be pure, to break from the past, or to be more simplistic or expeditious in our approach. To us, giving up something useful that has little cost to customers was a compromise that we didn't want to see in the evolution of PCs. The presence of different models is part of every platform. Whether it is to support a transition to a future programming model (such as including a virtualization or emulation solution if feasible), to support different programming models on one platform (native and web-based applications when both are popular), or to support different ways of working (command shell or GUI for different scenarios), the presence of multiple models represents a flexible solution that provides a true no-compromise experience on any platform.

Within the Windows desktop, **WOA includes desktop versions of the new Microsoft Word, Excel, PowerPoint, and OneNote, codenamed "Office 15".** WOA will be a no-compromise product for people who want to have the full benefits of familiar Office productivity software and compatibility, an industry-leading hardware-accelerated web browser, apps from Microsoft, and access to apps in the Windows Store.

This creates a WOA PC with the full power of apps, media consumption, entertainment, mobility, and productivity in one place—a true no-compromise experience. The new Office applications for WOA have been significantly architected for both touch and minimized power/resource consumption. This engineering work is an important part of being able to provide Office software with WOA, as these are not simply recompilations or ports, but significant reworking of the products with a complete and consistent user experience and fidelity with their new x86/64 counterparts.

You can learn more about the next version of Microsoft Office, codenamed "Office 15," on the Office Exec blog.

Engineering for ARM

Enabling Windows to run super well on the ARM architecture is a significant engineering task. We undertook this work because when you look to the future you can see that so many of the capabilities that have been added to Windows over the years are things that customers will inevitably desire or require in the types of devices supported by today's ARM-based products—changes in form factors and the desire for mobility only add to the scenarios and capabilities we all desire in our search for no-compromise PCs. While it is tempting to make bold statements about "starting over," we believe in the evolution of technology assets when the foundation is strong. The foundation of Windows, the core, is the most solid, scalable, and secure one around. Our desire to deliver a no-compromise experience motivates our efforts.

We also know that there are elements of Windows that require re-engineering in order to meet customer expectations for reliability over time, power consumption, resource utilization, and *instant* connectivity and availability. Obviously, all of this work is relevant to our Windows 8 on x86/64 product too, and much of what we have done for ARM will be applicable to the exciting new products coming from Intel and AMD (which are not the subject of this post). ARM affords us a chance to look at assumptions in OS behavior and programming model in order to deliver significant improvements.

One of the new aspects of WOA you will notice is that you don't turn off a WOA PC. WOA PCs will not have the traditional hibernate and sleep options with which we are familiar. Instead, WOA PCs always operate in the newly designed *Connected Standby* power mode, similar to the way you use a mobile phone today. When the screen is on, you have access to the full power and capabilities of the WOA PC. When the screen goes dark (by pressing the power button or timer), the PC enters a new, very low-power mode that enables the battery to last for weeks. All along, however, the system dynamically adjusts power consumption and is always on the lookout for opportunities to reduce power to unused parts of the system. For end-users, a unique capability of WOA is that *you* are in control of what programs have access to background execution so that those apps are always connected, and information like new mail is always up to date. Connected Standby permeates the engineering for WOA PCs from the hardware through the firmware, OS, WinRT platform, and apps. Connected Standby won't be limited to the ARM architecture and we are actively working on these capabilities for x86/64 SoC products as well.

Today, we are familiar with a PC experience where hardware that runs Windows built on x86/64 adheres to a set of technical specifications that allow one distribution of Windows code to install and run on a wide variety of PCs. This has enormous benefits of scale. This openness is also the hallmark of the PC revolution and represents the collective work of the industry since about 1980. When new hardware comes along that is broadly supported, these baseline specifications evolve, and the PC architecture moves forward. **Absolutely nothing about this approach will change for Windows 8**—as millions have experienced with our Windows 8 Developer Preview, **Windows 8 will run on every Windows 7 logo PC,** and will run all of the existing software and peripherals designed for and supported on Windows 7 (when supported on Windows 8 by the manufacturer, of course).

The approach taken by ARM Holdings, the licensor of ARM products is, by design, not standardized in this manner—each device from each manufacturer is unique and the software that runs on that device is unique. There is of course a standard instruction set and CPU architecture, one that is always improving (for example, adding 64-bit support and multiple cores), but many of the connections between the CPU and other components are part of the innovation each licensee brings to the ARM platform. Commonality across devices can occur under the hood, but is not applicable or significant to consumers. End-users are technically restricted from installing a different OS (or OS version) on a device or extending the OS, so this is generally not possible, and rarely supported by the device maker. Device makers work with ARM partners to create a device that is strictly paired with a specific set of software (and sometimes vice versa), and consumers purchase this complete package, which is then serviced and updated through a single pipeline. The cross-partner, integrated engineering of these embedded devices is significant. In these ways, this is all quite different than the Windows on x86/64 world.

With WOA, we set out to define a new way of developing a computing platform. We architected our approach to ensure that software and peripherals can all benefit from the diversity enabled by the ARM architecture, along with the choice of form factors and manufacturers, and the openness of the platform. At the same time we are making a commitment to customers that WOA will be consistent in capabilities, experience, and baseline performance across this spectrum. To those familiar with the Windows Phone 7 approach, the *chassis specification*, WOA shares some of those elements. The specifications being implemented for WOA allow for more diversity across many dimensions, combined with the same commitment to engineering and product excellence—all while running the same OS binaries across WOA PCs.

Engineering for ARM starts with the work we did to architect the Windows kernel so it could boot and run on ARM. As you might imagine, this was a significant effort. Some might believe that this is work along the lines of *porting* or merely *re-compiling* the code for a new instruction set. There's much more to the work than that when it comes to the kernel and the parts of Windows that connect with hardware. Along with the kernel work, we also had the work to develop the ARM compilers and tools (including Visual Studio), for building Windows.

At the higher levels—the application layers—the code is significantly portable because of our long history of running on multiple architectures (x86, x64, PowerPC, Alpha, MIPS, IA64, and so on). Even the kernel itself has a significant amount of code that can be ported. At the hardware/software seam and all the places that make assumptions about how an OS interacts with hardware, Windows has been reimagined for this new platform. To put some acronyms around this, the ARM definition does not require support for some common subsystems such as the PCI bus or SATA. There are analogous concepts implemented by each ARM implementation, but those are not always common. All of this was done over the course of iterating on three major revisions of ARM hardware since the start of the project.

Let's look at some of the types of work undertaken as part of this effort, which we referred to internally as "porting," despite the fact that it is so much more than that. Keep in mind that all of this work has been going on in parallel with the development of the user experience, Windows Store, WinRT, and new features across Windows 8.

Getting ready to port

Before the porting work could even begin, we needed an ARM compiler and tool chain for building Windows. Since other products at Microsoft (such as Windows Phone and Embedded) use ARM processors, we had these pieces, but further improved them to build Windows. These tools are going to be available to developers, and if you're using C#/VB/XAML/HTML5 in the Windows 8 Developer Preview, then you're already on board. C/C++ requires ARM native hardware for testing, which we'll talk about below.

Booting the core of Windows

Once we had the tools, we could start porting the Windows boot environment and developing system firmware specifications. We even prototyped the firmware ourselves. There are several pieces to this:

- **UEFI firmware** is the lowest layer of a WOA system and provides consistent services for loading the OS. For WOA, we created firmware to bootstrap the system that we handed off to our partners. WOA systems also include a firmware-based **TPM** for trusted boot and storage encryption. Using the TPM, for example, we've implemented trusted boot which verifies that the system hasn't been tampered with by malware.
- **ACPI firmware** is used for plug and play enumeration of devices in the platform during boot, and is also responsible for power management of devices outside the SoC (such as sensors, touch controller, etc.). Over the years, the PC has standardized with plug-and-play busses and ACPI, so that operating system software and drivers can "walk the tree" to find everything in a PC. With SoC embedded designs, there is no "tree" or ability to discover what is connected to a SoC, or even how the SoC is connected. During Windows 8, we worked to define a new standard to describe the configuration of the system with tables, so software can simply read the table and configure the system.

From the firmware, the system can then load the boot manager, boot loader, and in turn the kernel, HAL, and boot device drivers.

- The **Windows Hardware Abstraction Layer** (HAL) supports variations in core system resources (timers, DMA, interrupt controllers). Windows was designed from the beginning to support multiple instruction set architectures (ISA), and the HAL is key to adapting to different system architectures that often come with a new ISA. By abstracting the hardware layers, the OS itself doesn't have to be modified to accommodate a new SoC for core system resources. The variation across ARM platforms is significant enough that we architected the HAL to support a new level of capabilities of abstraction. New to the Windows 8 HAL is the ability for each of the core system resources to be plugged in via an extension to the HAL, kind of like a driver for the interrupt controller.

Devices and busses

In order to load device drivers and continue Windows boot, we had to build several new drivers for new types of low-power busses, plus device drivers that support connections to those busses.

Our device strategy uses standardized protocols and class drivers extensively. Our first example below is the HID over I^2C driver which we use for touch controllers and many sensors, another is the class driver for USB connected mobile broadband radios. Of course, Windows has many class drivers inside, which you experience when you plug in a wide variety of USB devices, such as storage, mice, or keyboards.

- Low power serial busses such as **I^2C / UART** will be normal on ARM PCs and less common on x86 PCs. These busses generally have a lower data transfer rate, but also use very little power, in some cases 10x less. Support for these busses is key to reducing the overall power use of WOA and extending battery life. Collectively, we call these busses **Simple Peripheral Busses (SPBs)** and we've developed new interfaces in WOA for them. Once we had the interfaces, we had to address a gap. In Windows, we have many device classes that are natively supported over USB via class drivers. These classes are undefined over I^2C, and hence they lack class driver support. One popular class of devices is Human Interaction Device (HID) protocol based devices. HID is the protocol of choice for devices like keyboards, mice, touchpads, speakerphones, buttons, touchscreens, etc. By defining a standardized protocol and

implementing driver support for HID over I^2C, we can work with partners to adapt the firmware of their I^2C-based devices to work with a single class driver. For example, by supporting HID over I^2C, touch controllers can use that interface and leverage the input support that Windows already has.
- **SD I/O** allows you to connect low power Wi-Fi radios. Radios in current PCs are connected via USB or PCI-E. We added SD I/O support to preserve high data rates (100 MB/s) while still improving battery life. Wi-Fi support on WOA also allows efficient offloading to maintain connections in connected standby while using very little power.
- **Embedded MultiMediaCard** storage (eMMC) is a *de facto* standard for storage on ARM devices (since most do not support SATA). This was an interesting challenge for us, since Windows expects a fast disk and very high bandwidth data transfer. In addition to supporting eMMC, we made several OS performance optimizations to reduce and coalesce storage I/O, resulting in fewer reads and writes to storage.
- The **General Purpose I/O** (GPIO) driver supports connecting buttons, interrupts or other I/O to the ARM processor.
- In addition to the GPIO driver, there's also a **button driver** for the Windows, power, and volume buttons. Buttons aren't standard on ARM devices. Each system requires a specific driver for all hardware buttons.
- We built a **new power framework** for managing SoC-wide power, total platform power, and the connected standby on/off usage model.

Getting to the Start screen

Once the firmware, HAL, boot services, boot devices, and busses were up and running, we were ready to bring up the rest of the system and get to the desktop and the Start screen.

- ARM SoCs for WOA have **DirectX capable GPUs** (DX) for accelerated graphics in Internet Explorer 10, in the user interface of Windows, and in Metro style apps. Taking advantage of a DX capable GPU is essential for delivering a responsive user experience. For each WOA target, the ARM partner has created a DX-compatible graphics driver. This is a significant undertaking of very complex code since today's GPUs are even more complex than the CPUs. To bring up Windows 8 on these new SoCs that did not yet have a graphics driver, and since ARM SoCs do not have the industry-standard VGA subsystem to fall back on for compatibility mode, our graphics team wrote a soft GPU driver that was capable of working directly against the hardware frame buffer. Aside from enabling development, it also enabled us to reimagine other things in Windows using the soft GPU driver when the normal GPU driver isn't available. For example, when running Windows Setup, or in those rare cases when Windows has a "bluescreen," we were able to give it a friendlier look and even localize it, so that even bad news can be presented more nicely across all platforms. This is a small example of work which is common to the x86/64 architecture as well.
- WOA PCs use hardware support for offloading specific work from the main processor to **integrated hardware subsystems**. This improves performance and battery life. For example, while watching a movie, the processing is done with multimedia offload (to a dedicated processor for example), and all other processing is minimized. Since the multimedia offload is optimized for playback, you can watch several movies without running out of battery or the PC could be designed to be even thinner and lighter. Another example is if you're working on a document and watching a movie at the same time, the movie is running on the offload hardware, which helps the overall system responsiveness. WOA takes advantage of several types of offloads including multimedia encode and decode as well as security offload for Bitlocker and EAS. This type of engineering also applies to x86/64, which also support offloading, and was introduced in Windows 7.

Connected device services

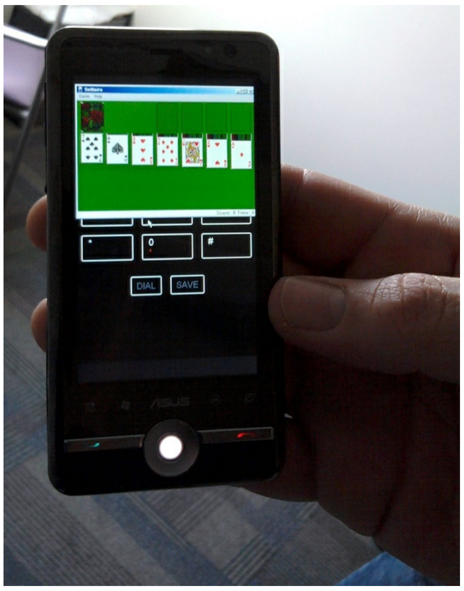
At this point we had the majority of the system running, and it was time to bring on the services to support the full breadth of Windows. These are common across the architectures that Windows supports, so that developers can take advantage of them in Metro style apps.

- **Mobile broadband** (MBB) class driver. By creating a class driver, we've made it much easier to add broadband capability to all Windows PCs. While WOA was a catalyst for this work, the entire ecosystem benefits.
- **Printer** class driver. For Windows 8, we rearchitected the print infrastructure to add class driver support. The majority of printers selling today are supported using the class driver, which means you'll be able to "plug and print" on WOA without additional drivers. While the new architecture was needed for many reasons, we had printing from WOA PCs in mind from the beginning.
- **GPS**. Windows offers a location provider that can triangulate a PC's location via Wi-Fi access points and a backing database. In addition, systems that have Mobile Broadband will also have integrated Global Navigation Satellite System (GNSS, aka GPS in the US) receivers to provide accurate location while navigating outdoors. The location platform plays a pivotal role in optimizing for power and accuracy by choosing the right location data provider to use based on the precision requested by the application.
- **Sensors (accelerometer, rotation, gyro, compass, magnetometer)**. A recent post described Sensor Fusion and how we've added support for sensors in Windows. This work also applies across all SoC-based architectures and utilizes the HID over I^2C protocol.
- **Bluetooth**. WOA supports Bluetooth LE and the same profiles as Windows 8 on x86/64 and connectivity to the Bluetooth radio using low-power UART.
- **MTP over USB and IP**. Windows on ARM provides users with the ability to connect their portable devices (like mobile phones, music players, cameras) to their systems using the Media Transfer Protocol (MTP). These MTP-compliant devices can connect over USB or IP by leveraging inbox Windows class drivers, and allow users to exchange data with their favorite Metro style apps.
- **Windows Update-based servicing.** For *all* platform code (OS, drivers, system and device firmware), each WOA system will be serviced through Windows Update (WU), from top to bottom. We've added support in WU for securely and robustly updating the system firmware on WOA systems, as well as driver targeting, which means that each device will get the drivers that have been verified to work best with it.

As you can see, some of this engineering work is strictly adapting to the new hardware platform. Some introduces substantially new types of hardware support. **In large part this work accrues to the x86/64 platform especially cutting edge products, such as the new low power ATOM® processors, demonstrated by Intel at CES**.

A significant amount also propagates to the application layer and becomes defining elements of the new WinRT APIs introduced at //build/. For example, while we engineered the kernel to support Connected Standby, delivering great battery life is really part of the overall WinRT application model and even the toolset, and all of that applies across WOA and Windows 8 on x86/64.

As we mentioned, a portion of Windows is generally built with code that can be made to work on ARM in a technically straightforward manner. These subsystems include the Windows desktop and applets and supporting APIs, though we needed to significantly re-architect all of them for better resource and power utilization. In fact, here is an early photo of an ARM device (an early Windows phone) running the full Windows desktop. Early in the development of WOA, the only hardware we had were existing ARM devices such as phones (ARM tablets didn't yet exist). We just thought you would enjoy a few **fairly early** photos I captured of debug WOA all loaded in RAM (unretouched). Note: *This is not a product plan or even a hint at a product*.



Testing

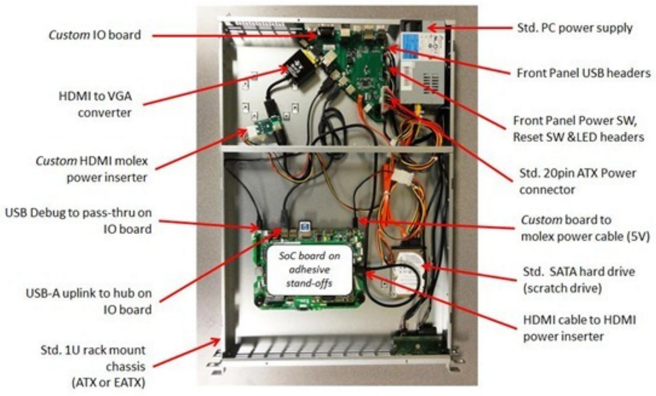
You might be wondering how we are testing WOA in our labs. For x86/64 we run massive labs (thousands of machines, real and virtualized) and highly automated test runs for every single daily build. WOA required us to reimagine our own lab and test processes. For testing x86/64, it is simple to just order thousands of rack-mounted servers, or even virtualize them; for ARM, there are no standard rack-mounted servers that can run WOA. Since we're doing highly integrated hardware/firmware/software development, and virtualization is not helpful, we had to devise our own approach.

We consolidated hundreds of ARM development boards along with a custom I/O board into a rack assembly connected to our testing infrastructure. Our original design focused on density supporting 300 ARM devices in one rack, but we ultimately preferred the diagnostics and availability of a custom I/O board in the 1U setup.

We designed our own 1U chassis that fits into a standard server rack. Either a full form-factor device or just the motherboard can be dropped into this chassis. Once fully assembled, the SoC board in conjunction with the IO board and chassis looks, feels and operates like a standard rack mount PC and fits right in with existing lab infrastructure.

Each 42U rack holds 32 WOA chassis plus network switches, debug host PC, and USB hubs. By March we will have over 100 fully populated *racks* for WOA testing.

We also had to port our test tool infrastructure and tests too, which was no simple challenge, but this ensured that we would cover WOA with the same rich automation used to validate Windows 8. Here's a photo of our newly devised test rack, and the board and debugging ports that it hosts:


Custom IO board
HDMI to VGA converter
Custom HDMI molex power inserter
USB Debug to pass-thru on IO board
USB-A uplink to hub on IO board
Std. 1U rack mount chassis (ATX or EATX)
SoC board on adhesive stand-offs
Std. PC power supply
Front Panel USB headers
Front Panel Power SW, Reset SW &LED headers
Std. 20pin ATX Power connector
Custom board to molex power cable (5V)
Std. SATA hard drive (scratch drive)
HDMI cable to HDMI power inserter

Developing for ARM

In practice all of this is *even* more in-depth than it looks. We also took this chance to do a very significant re-engineering of every Windows subsystem. In the course of building WOA and Windows 8, we invested a huge amount of energy into changing all of Windows to work better at minimizing overall power consumption and resource utilization, while simultaneously delivering improved real-world performance for existing application workloads. In previous posts on boot, power management, and memory usage, you have seen the results of some of this work.

Previously we have detailed that **WOA will not support any type of virtualization or emulation approach, and will not enable existing x86/64 applications to be ported or run**. Supporting various forms of emulation runs counter to the goal of delivering a product that takes a modern approach to system reliability and predictability—by definition, existing code has not been optimized for the platform the way WOA has. Virtualized or emulated software will consume system resources, including battery life and CPU, at unacceptable levels. Emulation and virtualization of existing x86/64 software also **require** the traditional PC environment of mouse and keyboard, which is not a good assumption for WOA PCs.

If we enabled the broad porting of existing code we would fail to deliver on our commitment to longer battery life, predictable performance, and especially a reliable experience over time. The conventions used by today's Windows apps do not necessarily provide this, whether it is background processes, polling loops, timers, system hooks, startup programs, registry changes, kernel mode code, admin rights, unsigned drivers, add-ins, or a host of other common techniques. By avoiding these constructs, **WOA can deliver on a new level of customer satisfaction: your WOA PC will continue to perform well over time as apps are isolated from the system and each other, and you will remain in control of what additional software is running on your behalf, all while letting the capabilities of diverse hardware shine through.**

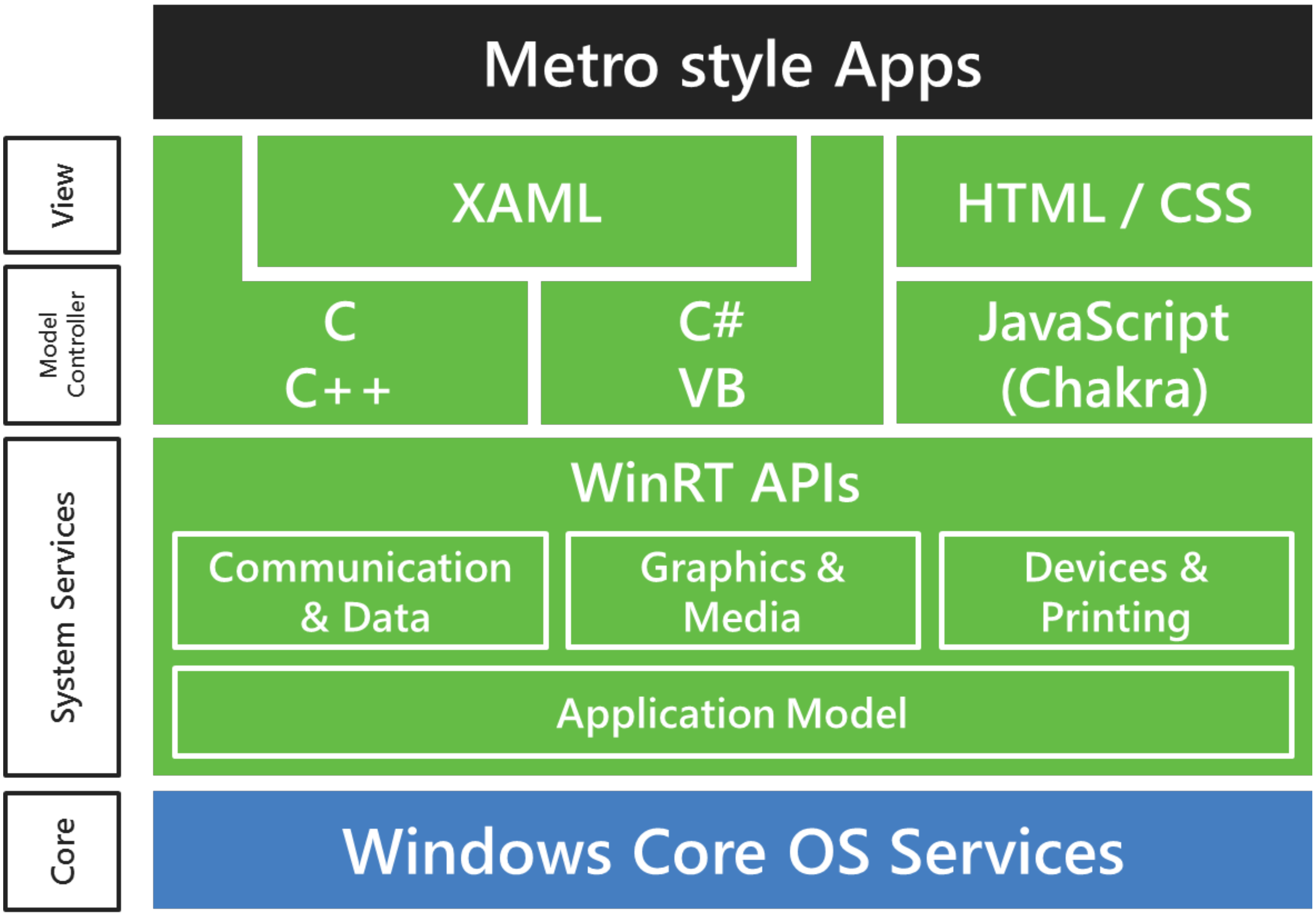
Our focus on delivering a new level of security for consumers using WOA is paramount. In one public event, we were asked if we would "make it easy for existing viruses and malware to run." Now you can see the answer is decidedly, "no." In fact, WOA only supports running code that has been distributed through Windows Update along with the full spectrum of Windows Store applications. As we all know, security is an industry-wide, multi-dimensional challenge and no system or platform can make broad claims without considering many factors.

If you need to run existing x86/64 software, then you will be best served with Windows 8 on x86/64. If you're already considering a non-Windows device, then we think WOA will be an even better alternative when you consider the potential of form factors, peripherals, Windows Store apps (and developer platform), and Office applications as well as a broad set of intrinsic Windows capabilities.

Of course, we recognize that many developers at Microsoft and elsewhere rely on existing programming techniques, and that a transition to WOA will require an investment. Developing apps based on WinRT addresses these and many other challenges from the start—WinRT is designed to provide the full expressive power required for modern software while avoiding the traps and pitfalls that can potentially reduce the overall experience for consumers and does so with a deep commitment to tools, languages, and developer support.

Developers wanting to reach WOA with existing apps have two options. Many apps will be best served by building new Metro style front ends for existing data sources or applications, and communicating through a web services API. This approach will be quite common for line-of-business applications and many consumer web properties, and represents the best way to tap into the power of a rich user interaction model where you can also interact across and share information with other new apps. Of course, these do not need to be just front-ends, but could operate on local data too, since WOA provides full access to files and peripherals. Other existing applications will be well served by reusing large amounts of engine or runtime code, and surrounding that with a Metro style experience. This will take some time, and represents a way for applications that are composed of significant intellectual property to move to WOA and WinRT. In all cases, WinRT represents the new set of Windows OS services that developers can use to build software that is *Designed for Windows 8*.

Returning to our architecture diagram from //build/, from a third-party developer perspective, the best way to think of WOA is as the expression of the Metro style platform, which shares the Windows Core OS with all the other Windows products. The Windows Core OS has been tuned and architected to support the ARM platform and is there to support the WinRT APIs and programming model used by third parties.



The topic of engineering for ARM is a broad one and has occupied many on the Windows team for the course of the project. The next step is delivering WOA code more broadly, but that starts with how we're going to bring WOA PCs to market.

Delivering WOA PCs

Since the conclusion of the Windows 7 project we have been working with PC makers on the evolution of Windows and the creation of Windows 8. There is a vast amount of joint work that goes on in order to bring new PCs to market—the "Designed for Windows" logo that you see on a PC represents the collective work of a wide array of partners sharing a commitment to bring new and exciting PCs to market. The model we used and will continue to use to bring x86/64 PCs to market as an industry is the same as we have always used—we will introduce new technologies such as USB 3.0, UEFI, touch, and sensors and support those in a new release of Windows with new hardware. This is a collaborative and ongoing effort, with a great many improvements to be introduced with this product cycle.

Delivering WOA PCs is building out a new system for the first time—a completely new ecosystem of PCs providing opportunities for PC makers to bring to life a new generation of PCs with new capabilities. **We describe these PCs as being focused on achieving new levels of capability along three dimensions: thin and light in industrial design, long battery life, and integrated quality.**

Because of the necessarily tight connection between SoC, peripherals, firmware, and the OS, WOA PCs should be thought of as joint engineering that goes well beyond industry partners merely collaborating. This is an effort where software people on the Windows team end up debugging silicon with soldering irons, and hardware engineers end up in Visual Studio, debugging timing issues with user interface code. Thus every WOA PC is a new engineering effort that starts with the selection of components and continues through with firmware, drivers, final assembly, and unique apps from PC makers. We also bring new ARM designs up on simulation and emulation platforms to get it right at the start, even before the silicon is available. And we are bringing the ecosystem together to do total platform design for low power, which includes not just a great SoC but efficient radios, sensors and even higher efficiency DC power infrastructure. It all matters for super thin and light PCs, with great battery life, and high-quality engineering providing a great experience with apps and services that are *Designed for Windows 8*.

While each WOA PC offered will be unique, the role of Windows is to present a consistent experience to customers while **allowing the unique and innovative hardware to shine through—the very definition of an OS**. To achieve this we have been working with multiple ARM licensees as mentioned—Texas Instruments, Qualcomm, and NVIDIA. Each has been working with partners that will bring WOA PCs to market. **These PCs have all been designed and manufactured expressly for WOA**. From the chipset through the firmware and drivers, the work is optimized to be great for WOA. Partners are working hard on creative industrial designs and form factors that will include more than tablets. These are all under development today. **Our collective goal is for PC makers to ship WOA PCs the same time as new PCs designed for Windows 8 on x86/64, using the latest generation of those platforms from low-power to high-performance.**

While not the topic of this post, we do want to assure you that, **when a consumer buys a WOA PC, it will be clearly labeled and branded so as to avoid potential confusion with Windows 8 on x86/64**. The PC will come with the OS preinstalled, and all drivers and supporting software. WOA will not be available as a software-only distribution, so you never have to worry about which DVD to install and if it will work on a particular PC.

WOA PCs will be serviced only through Windows or Microsoft Update, and consumer apps will only come from the Windows Store, so you never have to worry if a program will run because you are not downloading or installing from a DVD outside of the store experience. A WOA PC will feel like a consumer electronics device in terms of how it is used and managed. For example, as previously detailed, the new refresh and reset functionality will be available, and for WOA this provides the equivalent of a "clean install" or imaging.

Next steps

There's much more to this story, and we plan on more posts detailing the engineering of WOA and all the work that went into building the OS based on the dialog that follows this post. Many are keen to get their hands on the software. But of course there is no existing hardware to use as there is with Windows 8 on x86/64, which can make use of PCs designed for Windows 7. We are approaching a step in the project where we intend to broaden the distribution of the WOA software with development hardware.

To run this release, a low volume of test PCs specifically designed for WOA will be made available starting around the next Windows 8 milestone. These devices are for developers and hardware partners, and do not represent consumer form factors, by any stretch of the imagination. They have diagnostic tools and ports. They are designed to be opened and debugged. They do not have the final components or firmware (or power or thermal management) that a commercially available device will use. They are made of low-cost plastic. You might have seen devices similar to these on display at CES or demonstrated there, and all of our previous demonstrations have used some form of these test PCs. These PCs do represent WOA and the experience—but they no more represent the final experience than does the current state of x86/64 Windows 8. They will be running the same branch of Windows that will be made available to x86/64 testers at our forthcoming development milestone.

These PCs are expensive to make and distribute because they are basically low-volume custom PCs. They will be made available through our developer evangelism efforts. We are talking about this not to tease you or to solicit nominations, but because we know word will get out and images of these will be on the web. The devices are all already spoken for and allocated. On the one hand it seems a little cruel to dangle this in front of you, but on the other hand it is worth considering that this level of transparency is a hallmark of how we develop Windows. The scale of the Windows 8 project is significant, and combined with the degree to which we are forthcoming with information and dialog about our decisions, it is without precedent.

By the end of the month, the *Windows Consumer Preview* (the beta) of Windows 8 on x86/64, will be made available for download. We changed the name of the beta because recently the term has come to mean something very different than just a "testing release available for free to try out," so we did not want to add to the confusion. In keeping with the level of openness described, there is no pre-registration or admission to a test program—just download it and install it on a Windows 7 logo PC (although VMs are supported, that is not the best way to try out the consumer experience). We have made a ton of progress and there are many significant changes since the Windows Developer Preview 5 months ago. **As a reminder, we're still building Windows 8 and WOA, and there is much work to be done to go from pre-release to release. Quality remains our priority. The code is not done.**

We are very excited to be approaching this milestone. The responsibility of developing a new release of Windows is humbling, and the challenges of releasing an entire new platform such as WOA are both energizing and daunting. We look forward to welcoming everyone to the Windows Consumer Preview in short order.

On behalf of the Windows team,

Steven Sinofsky