Windows® 10 IoT Core Running on Intel® Architecture Platforms – One Year Later!

By Sean D. Liming and John R. Malin
Annabooks – www.annabooks.com

November 2017

After 1 year, Windows 10 IoT Core running in Intel Architecture is taking shape. Two releases have been made since we first discussed IoT Core's ability to run Windows 10 IoT Core on Intel Architecture platforms using version 14393. The latest release 1709 (16299.x) has many improvements and addresses several of the findings we discussed previously.

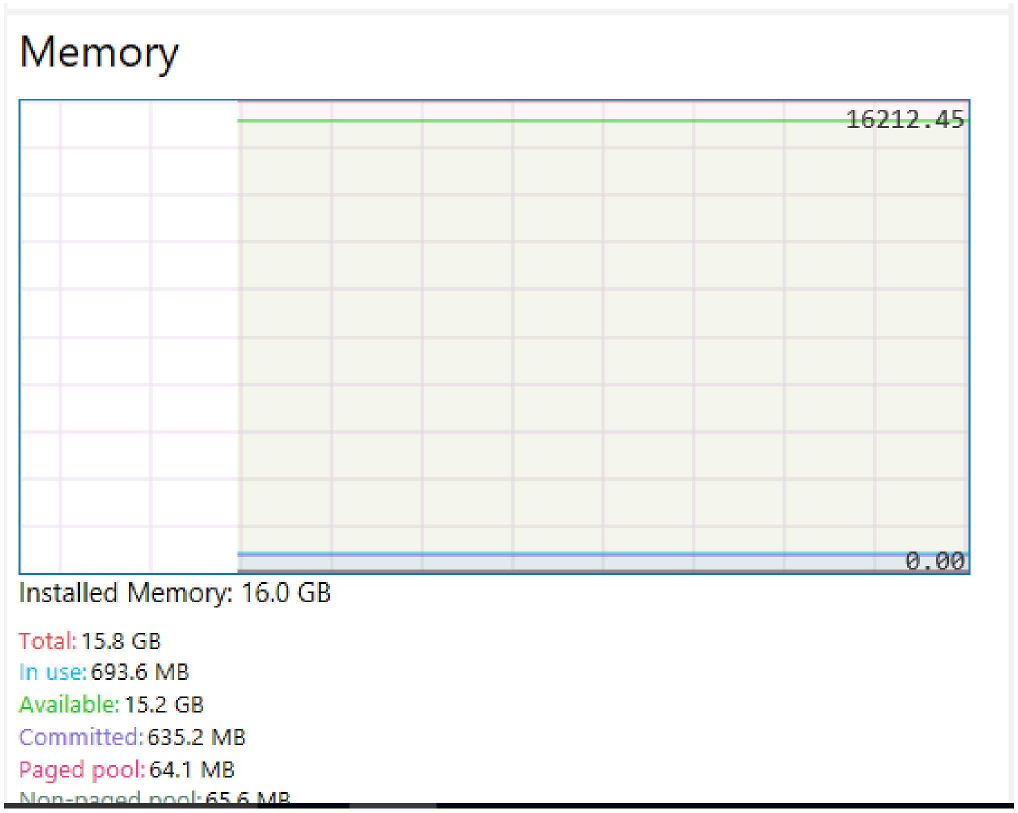
A Little Background First
Before we go further, we need to provide some background, if you didn't see our first article. Windows 10 IoT Core is a cut down version of Windows 10 that only runs Universal Windows Platform (UWP) and traditional command line applications. Since the launch of Windows 10, we have had a number of clients interested in the Windows 10 IoT Core, because the bigger Windows 10 IoT Enterprise might have too much overhead. Although the core version could be used, the clients were trying to use Intel® Core™ i-Series processors. A the time, the biggest issue holding Windows IoT Core back was the processor and board support being limited to a few hobbyist platforms: Raspberry Pi2/3 (Broadcom BCM2837 ARM® Cortex® A7), Dragonboard 410c (Qualcom Quad-core ARM® Cortex® A53), and the MinnowBoard Max (Intel® Atom™ E3800). The only industrial board comes from Toradex, which has the Nvidia® Tegra 3 ARM® Cortex®-A9. The reason for limiting support is very logical. Learning from the difficult lessons about Windows CE porting, Microsoft is controlling the port of the IoT Core. OEMs who want IoT-Core to run on a specific processor must contact the processor vendor. The processor vendor must then work directly with Microsoft to get a port created. The OEM will get the BSP from the processor vendor to build the image. The Microsoft / processor vendor collaborative approach saves the OEM time, money, and headaches in trying to port to a processor. The drawback is a chicken and egg. It leaves it to the processor vendor to pursue the support. If the processor vendor doesn't see the value, a port never gets made.

ARM processors come in different flavors and require some porting. Intel processors are a set architecture. Intel has a range of different processors to address specific devices; but in the end, they all have the same basic platform architecture. This makes the support for the Intel Atom E3800 very interesting, since there are many industrial PC platforms using the E3800 family. With all of our clients in mind, we set out to investigate what was needed to get IoT Core running on Intel Architecture platforms. What we found were the following:

- UEFI firmware and operating system bitness needs to match. If you want to run Windows IoT Core 64-bit, the firmware has to be 64-bit. If you want to run Windows IoT Core 32-bit, the firmware has to be 32-bit. Pure UEFI boot in the firmware is required. For most Intel boards, 64-bit firmware is installed by default, thus the release of 14393 support of 64-bit allowed us to run IoT Core on industry PC platforms.
- Intel i-Series processors meet the minimal requirements to run IoT Core. We did a basic test on an i7 platform and IoT Core booted just fine.
- Creating device driver packages was a challenge. The conversion tools to convert existing driver packages required some effort to get the packages correctly built and integrated into the image.
- Memory was limited to 4GB max. The i7 platform had 16GB RAM installed, but IoT Core was fixed internally to only support 4GB, which was the limitation.

What a difference a year makes!
Here is what we are finding with the 16299.x so far:

- 4GB limit is gone – The biggest concern with IoT Core Build 14393 on Intel Architecture was the 4GB RAM limit. For the latest release Build 16299, the RAM limitation has been removed and we can access the full 16GB for an i-Series processor.



Installed Memory: 16.0 GB

Total: 15.8 GB
In use: 693.6 MB
Available: 15.2 GB
Committed: 635.2 MB
Paged pool: 64.1 MB
Non-paged pool: 65.6 MB

- Creating device driver packages is easier. The iot-adk-addonkit contains the scripts to build the packages and the image. The kit has been updated to create a simple inf2cab.cmd that removes the previous two step process.
    - We previously covered an Intel Graphics driver for i-Series platform, which had some challenges to create a CAB file package. The new inf2cab solution is a much improved process over the previous process. The old process created an XML file with all the files in the INF. The new process creates an XML with just a pointer to the INF, so it lets Windows PnP do the work of installing the driver. A log file lists what was found and where it will be placed.
    - For other device driver INF files, there are still issues with missing and extra files, but this was the fault of the INF and not of the parser algorithm. INF files that have been built up over time could call out legacy files that are no longer provided. There are also extra files like MSI and setup.exe files that could accompany a driver and be called out in the INF. Depending on the device driver, these installer files might not be needed. Editing the INF and commenting out the missing or additional files solves the issue.

A Process to Create 64-bit IoT Core Image for IA Platform

Starting with WES2009 and now into Windows 10 IoT Enterprise, we developed a process for Windows image development. The first step is to integrate the device's drivers and the second step is to perform the applications and OS architecture integration. The same process can be applied to Windows 10 IoT Core. Here are the high level steps:

1. Install Windows 10 Enterprise on the target Intel Architecture system
2. Install all the device drivers. If you have to, run Windows update to download the latest device drivers for the system.
3. Open a command prompt and run the following:

   ```
   DISM /Online /Export-driver /Destination:c:\drivers
   ```

4. Copy the "drivers" folder to your development machine, and run the inf2cab script on each driver INF to generate the cab files. Here is an example

   ```
   inf2cab.cmd c:\Intel-Video\igdlh64.inf GBi3.Video
   ```

Note: There is an example IoT Core BSP package at http://annabooks.com/SW_Intel_Atom.html that uses a script to create multiple packages for a group of device drivers.

5. Create a new BSP and link the CAB files to the BSP's FM.xml file.
6. Create a new product using the new BSP. Make any changes to the OEMInput.xml file.
7. Build the image.

Summary – The Door is Open for IoT Core on Intel Architecture Platforms

There has been steady progress with Windows 10 IoT Core for embedded/IoT systems. IoT Core 14393 added 64-bit Intel Architecture support. IoT Core 15063 and 16299 fixed the RAM issue and improved the development process. Our previous investigation ended with things looking promising for IoT Core on IA platforms. With an improved development process and full RAM memory support, IoT Core is ready to take over as the small Windows operating system for embedded/IoT projects.