Windows® 10 IoT Core Pro Boot Media and Sudden Power Loss Review

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

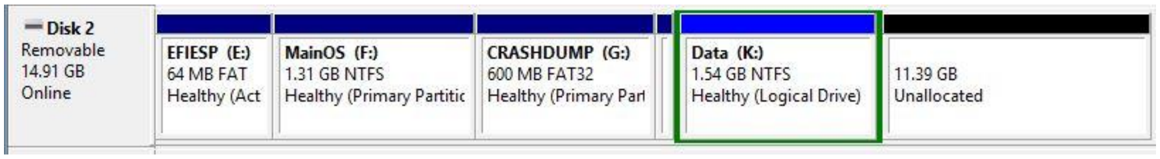
October 2016

Microsoft offered an opportunity to get an early hands-on review of Windows IoT Core. After the presentation about what Windows 10 IoT Core is and that it is still Windows 10, I asked about the lockdown features like Unified Write Filter (UWF). Before I could explain the reason why UWF would be needed to protect flash life and sudden power failure, the presenter said that IoT Core had a special partition scheme that is used in Windows Phone and that UWF would be too much overhead. Not pressing the issues, I decided to check on this information on my own. As soon as IoT Core was released, I tested the image on a Raspberry Pi 2. Everything seemed to be working as expected, and then I decided to pull the power. As expected the system didn't boot when power was reapplied. Only after re-flashing the image, could the system be rebooted. Over the year since the initial launch of Windows 10, others have posted to the Windows IoT forum about the sudden power loss issue. Microsoft quietly released a download for UWF in December 2015, and made UWF a selectable feature in the IoT Core anniversary release. Something changed their minds to about adding UWF to IoT Core.

Having worked with Windows Embedded going all the way back to Windows NT Embedded, I know that Windows doesn't like it when the power gets pulled. Over the years, there have been many techniques developed to help mitigate image corruption on sudden power loss. In this paper, we will look at the special partitioning, boot media support, and ways to mitigate image corruption from sudden power loss. With the anniversary release, 14393, we took a closer look at boot media, sudden power loss, and why adding UWF is important for overall system architecture.

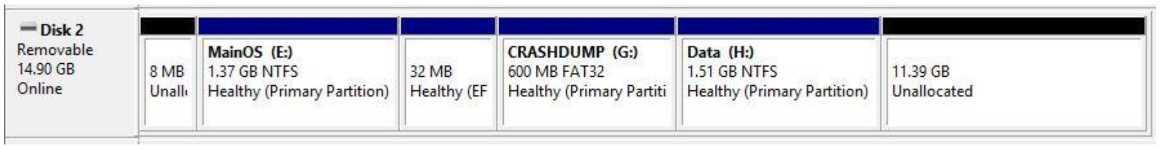
Image Partitioning and Wear-Leveling

After flashing an image to an SD card, you can see how the partitions are set up. There is the expected small boot partition, the main OS partition, a partition for crash dumps, and a partition for data. The crash dump and data partitions are accessed from the main partition via volume mount-point folders of the same name.



Disk 2 Removable 14.91 GB Online	EFIESP (E:) 64 MB FAT Healthy (Act	MainOS (F:) 1.31 GB NTFS Healthy (Primary Partitic	CRASHDUMP (G:) 600 MB FAT32 Healthy (Primary Part	Data (K:) 1.54 GB NTFS Healthy (Logical Drive)	11.39 GB Unallocated

Figure 1 - Windows 10 IoT Core Raspberry Pi2 Media Layout



Disk 2 Removable 14.90 GB Online	8 MB Unall	MainOS (E:) 1.37 GB NTFS Healthy (Primary Partition)	32 MB Healthy (EF	CRASHDUMP (G:) 600 MB FAT32 Healthy (Primary Partiti	Data (H:) 1.51 GB NTFS Healthy (Primary Partition)	11.39 GB Unallocated

Figure 2 - Windows 10 IoT Core MinnowBoard Max Media Layout

Anyone who has worked with Windows NT Embedded, XP Embedded, WES7, etc. knows that multiple partitions on small flash drives is an incorrect implementation and prone to corruption in a sudden power loss. Two features are at play here. The first is that flash drives implement wear-leveling in order to keep the mean time before failure of the device high. If you look at the Wikipedia page on wear-leveling, different types of flash drives have implemented wear-leveling differently. The second is that NTFS is a very "chatty" file system. This is apparent with the SSDs that are currently available and being deployed in desktop PCs. The manufacturers are supplying configuration utilities that tweak the system settings to minimize how much the file system beats up the flash. Some flash disks have better wear-leveling implementations than others. As the OS and in particular NTFS is constantly reading from and writing to the disk, the wear-leveling is moving data to different flash blocks so there are not concentrations of high use that would cause the flash to burn out quickly. As the blocks are being moved around, so is the information about where all the partition and file locations start and stop. If power goes out, there is a chance the image could get corrupted. On a Windows Phone, you would never run into the issue since the phone is powered by a battery which is not easily removed. For larger SSD and mSATA drives, there are more blocks, more sophisticated internal controllers, and better wear-leveling algorithms making it possible for multiple partitions to work. Flash technology has improved. New flash drives have a low failure rate compared to flash drives from 15 years ago.

The Raspberry Pi2/3 and the MinnowBoard Max use microSD cards as boot media. microSD/SD cards were intended to store data like pictures or music. They are not robust enough to handle the constant reads and writes from NTFS and act as an operating system boot drive. When I explored the Yocto Project to create custom Embedded Linux distributions, I burned through several microSD cards on the Beaglebone platform.

Not all flash drivers are created equal. Different brands and models offer different performance and robustness. Industrial grade is better than commercial grade for boot drives. Some manufacturers have power sense technology that detects power going down and latch up address lines. I have worked with a few clients that have gone through their own internal studies on the best makes and models. It seems that every new generation of flash chips and flash drives changes. The question comes down to “what is the best boot media for an operating system?” Since embedded / IoT systems vary in design and function, there is no one correct answer, but the next two sections provide some basic data to help you with your own conclusions as you research your own specific system.

Boot Speed Review

System performance is governed by processor performance, the amount of RAM, bus speeds, and boot media. Embedded / IoT systems have a wide range of performance requirements. Higher-end processors are going to perform better than smaller, low-powered processors. One of the important requests we hear from customers is fast boot time. Rather than focus on operational performance, we decided to test different systems with different boot media, and get a measure of the boot time. From cold power-on until the IoTCoreDefaultApp application is up and running on the screen.

Platform	IoT Core Image	Boot Media	Boot Time (Seconds) ±0.5
Gigabyte BRIX i3-5010U	14393 – 64-bit Custom Image	mSATA - Patriot Pyro m3	14.3
Gigabyte BRIX i3-5010U	14393 – 64-bit	mSATA - Patriot Pyro m3	14.7
MinnowBoard Turbot	14393 – 64-bit	mSATA - Patriot Pyro m3	31.2
		SSD SATA – Intel SSD 320 40GB	31.6
		USB Flash – Memorex 8GB USB 2.0	75
		microSD HC – Samsung 16GB pro Grade 1, Class 10	39.1
		microSD HC – Patriot 8GB Grade 1, Class 10	55.4
Raspberry Pi 2	14393 – 32-bit	microSD HC – Samsung 16GB pro Grade 1, Class 10	49.4
		microSD HC – Patriot 8GB Grade 1, Class 10	65

Table 1 - Boot Time Results

Note:

- All images were built from IoT Core Pro release 14393.
- For Gigabyte BRIX, a custom IoT Core Pro image was built based on 14393 MBM that also includes video and wireless device drivers for the platform.
- Images had already completed the initial boot, so all boot times are based on normal condition.
- The times are based on the average of 3 cold boots.
- USB 2.0 vs. USB 3.0 flash disks - The deployment to USB 3.0 disks corrupted the USB disk so the disk wouldn't boot. It was also a challenge to recover the disk via the diskpart.exe utility. We went with an older USB 2.0 flash disk.
- SSD and mSATA completed initial boot and setup of the image the fastest.
- All systems were connected to a wired network.

Sudden Power Loss, Flash Life, and Unified Write Filter (UWF)

The next test was sudden power loss, and we decided to keep this simple. The Samsung SD card was used in the sudden power-off test for the Raspberry Pi 2 and the MinnowBoard Turbot board. Only the mSATA was used in the BRIXi3 platform. Three power-off scenarios we tested: OS loading at power up, normal operation, and power down.

- Over all, there was no noticeable disk corruption for any platform when pulling the power during the test. We didn't have to re-image the systems during the test.
- All 3 systems did not show any issues when power was pulled when the OS was loading.
- All 3 systems had issues when power was pulled during normal operation and power-down. The issues were not consistent:
    - Most of the time, the systems would boot normally.
    - On one occasion for each the MinnowBoard Turbot and the BRIX i3 the systems booted to a black screen. Hit the F8 key, the system would continue to boot.
    - The Raspberry Pi 2 came to a black screen, pulled the power and plugged back in the OS booted normally.
    - For the Raspberry Pi 2, we would sometimes get disk check on startup. We never saw a disk check with the MinnowBoard Turbot and BRIX i3

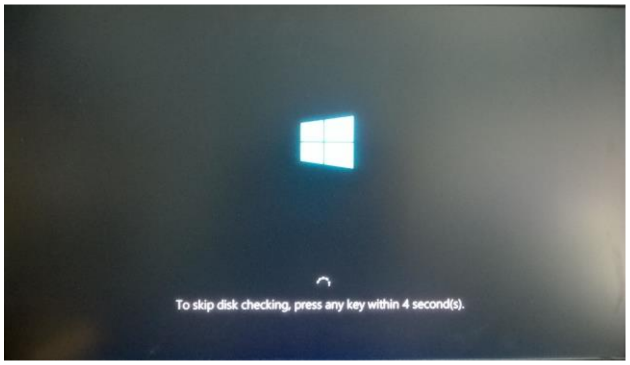
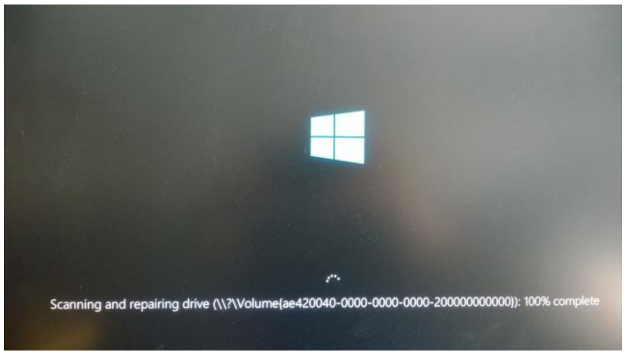

To skip disk checking, press any key within 4 second(s).

Figure 3 - Skip Checking on boot up

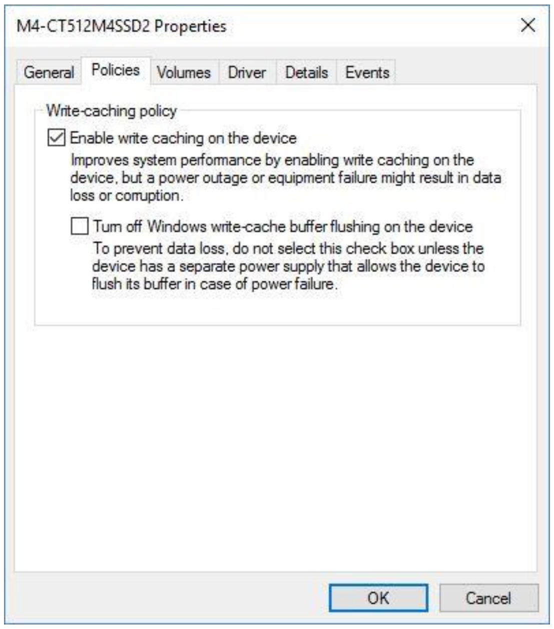


```
Scanning and repairing drive (\\?\Volume{ae420040-0000-0000-0000-200000000000}): 100% complete
```

Figure 4 - Disk Check Running

Sudden power-off during OS load on start and shutdown is not a common occurrence. The real concern is under normal operation, when the OS and main application are running. Our final test was to add the Unified Write Filter (UWF). Working with many clients since the release of Windows NT Embedded, we have developed some best practices to help mitigate disk corruption from sudden power loss:

- Choose industrial grade flash – if you are going to use a flash disk as the boot media, you might want to look for industrial grade flash devices. Also check for devices that have power sensing technology to latch up address lines as power is going down.
- For small flash disk, keep the partitions to a minimum – not possible with Windows 10 IoT Core as we already have discussed.
- Format the disk with NTFS – not much choice since Windows 10 will only boot from a NTFS partition.
- Protect the boot partition with a write filter. The Unified Write Filter (UWF) is featured in Windows 10 Enterprise and Windows 10 IoT Core build 14393.
- Page File turned off – when UWF is enabled, the page file is off by default. There is no page file in IoT Core.
- Disable disk caching – it is better to have everything flushing to the disk rather than get lost in RAM as the power is going down. This section is focused on disk corruption and not data corruption. Data corruption can still occur if data isn't written when the power goes out. Turning off disk caching will help prevent data corruption, but it might not be 100%. The dskcache.exe utility was available in XP Embedded to disable the disk cache feature. There is no such utility or capability in Windows 10 IoT Core. If you try to turn off disk caching in Windows 10, the OS will turn it on during the next reboot. This will be something for future investigation.


M4-CT512M4SSD2 Properties

General | Policies | Volumes | Driver | Details | Events

Write-caching policy

☑ Enable write caching on the device

Improves system performance by enabling write caching on the device, but a power outage or equipment failure might result in data loss or corruption.

☐ Turn off Windows write-cache buffer flushing on the device

To prevent data loss, do not select this check box unless the device has a separate power supply that allows the device to flush its buffer in case of power failure.

OK | Cancel

Figure 5 - Disk Cache Enable/Disable

So the one thing we can do for IoT Core is to enable UWF. The concept of the write filter goes all the way back to Windows NT Embedded. The idea is that the filter protects the OS partition by directing all writes to an overlay. The overlay could be RAM or disk cache implementation. The Enhanced Write Filter and File Based Write Filter were introduced in XP Embedded and carried on in Windows Embedded Standard 7. Starting with Windows Embedded 8 Standard, UWF was introduced as the future replacement for these filters, and is the only filter available in Windows 10.

We custom built images for all three platforms with the UWF filter installed (IOT_UNIFIED_WRITE_FILTER). We enabled the filter to protect C: drive, RAM overlay, and no exclusions. The same boot media for the initial sudden power-off test was used again. The power was pulled three times on each system. All three systems were able to boot again without any trouble. The combination of NTFS and UWF performed as expected, protecting the system from sudden power loss. Microsoft made a wise choice to make UWF available for IoT Core.

The results do not mean that data corruption or boot failure will not occur. We were not running an application that was writing data to the disk, and we only did a few power cycles (3 to 4) to see that there was a difference when UWF was enabled and UWF was disabled.

Summary: Architect the System

As with all our test and results, your results will vary. This paper only serves as a starting pointThere is more effort to be put into this area for IoT Core as more developers start to use it. We encourage that you pay close attention to the type of media you choose and how it performs over the life for your product.

We have stated in books and articles that architecting the system is important. One doesn't just pick a few features, build an image, and ship. Adding UWF will help add some level of protection for flash life and issues from sudden power failure. Of course, UWF has its own quirks that must be addressed, but we will discuss this at a later time.