**The Problem**

After SWAP bit to Swap bit application update, for example RP4.2-TY11 to UUT 0.0.0.7, We saw an inconsistency of seeing the **UsbChangeEvent(**) being called.

The Main page generates the **UsbChangeEvent(IntPtr hwnd, int msg, IntPtr wParam, IntPtr lParam, ref bool handled)** when a USB device is attached.

We observed very inconsistent behavior when we had the logs, meanings these prints were not seen as expected. This caused many reconnection issues.

This effected the connection state of the device.

**The Expected USB event behavior**

The UsbChangeEvent() should be called once when the USB device detaches, and UsbChangeEvent() should be called twice when attached.

Which should print the message as shown below

**We Connected the USB device here**

The thread 0x9894 has exited with code 0 (0x0).

Main - Device Changed | 1511000 | 537 | 7 | 0

[2022-07-19T22:28:02] DeviceManager: UsbChangeEvent: hwnd=1511000, msg:537, wParam=7, lParam=0, handle=False Connect event

Main - Device Changed | 1511000 | 537 | 7 | 0

[2022-07-19T22:28:02] DeviceManager: UsbChangeEvent: hwnd=1511000, msg:537, wParam=7, lParam=0, handle=False Connect event

**We disconnected the USB device here**

Main - Device Changed | 1511000 | 537 | 7 | 0

[2022-07-19T22:28:05] DeviceManager: UsbChangeEvent: hwnd=1511000, msg:537, wParam=7, lParam=0, handle=False Disconnect event

The Main page generates the **UsbChangeEvent(IntPtr hwnd, int msg, IntPtr wParam, IntPtr lParam, ref bool handled)** and prints"Main - Device Changed | 331564 | 537 | 7 | 0" .

This method is executed in the Main Thread as you can see Thread view, call stack.

Graphical user interface, text, application

Description automatically generated

**How USB Event reception effects reliable connection of the device after update**

The following step are followed for updating the firmware

**EraseChip() Message0xF2(242) reply 0xF3(243)**

**SendFlash() message 0xF2(242) reply 0xF5**

**RunUpdateApp() message 0xF4 (244) expected reply 0xF5 (245) This never is replied**

When we had the after calling KinetisUpdater.UpdateTask() execute **RunUpdateApp()** then cal**l** UpdateCompleteEvent() which in turn calls Dispatcher.BeginInvoke(UpdatedCompletedUI(status);) Then we can see that from the log that "Main - Device Changed | 1511000 | 537 | 7 | 0" is printed only twice(see the log below). Apparently "Main - Device Changed | 1511000 | 537 | 7 | 0" corresponding to the device disconnect is not called. Which means UsbEvent() is called only twice and the UsbEvent() call corresponding to the device disconnect is not called.

When the UsvEvent() is called to check if the connection has changed it checks ***bool connectionState = CheckDeviceAttached().***

The ***CheckDeviceAttached()*** tests if  ***connectionInfo.deviceId == (string)child.PnPDevice.GetPropertyValue("DeviceID").*** Where the ***connectionInfo.deviceid*** has the device ID of the previous connected device. Since we did a swapbit application the device ID of the newly attached device ID is same as the previous device ID. So the UsbEvent() fails to recognize the device connection has changed.

The HID receive Thread of the previous connection definitely kills it self since the old swapbit

The device officially disconnects from windows, as you can see from log output "HidDevice: Killing rx thread. Too many errors" since the device's old application shuts down and new one reboots.

Since the UsbEvent() is not called while the device is detached ( during the finite time the device is rebooting it self). We have no way of knowing of the device has changed to trigger a **CheckForNewDeviceConnection()** in the UsbEvent() it self. And the 2 subsequent called UsBEvent()s complete it self with

"Connection status did not change [True]"

[2022-07-26T00:13:02] KinetisUpdater 0: Setting flashing in progress to true

[2022-07-26T00:13:02] KinetisUpdater 0: Starting update step [FLASH\_SWAP\_BIT\_APPLICATION] using file [C:\Projects\PC\_Software\UWFU\_GitLab\ufwu\_application\UniversalFreescaleUpdater\bin\Debug\_Dev\UpdateFiles\RP5-GM31\0.0.0.7\A-109\_TESTER\_UUT\_V7.pufx]

[2022-07-26T00:13:02] KinetisUpdater 0: Attempting to erase the chip...

[2022-07-26T00:13:02] Kinetis Device: tx-F000

[2022-07-26T00:13:02] Kinetis Device: rx-F1010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

[2022-07-26T00:13:02] KinetisUpdater 0: Erasing Chip - PASS

[2022-07-26T00:13:02] KinetisUpdater 0: Attempting to flash the chip...

The thread 0x6658 has exited with code 0 (0x0).

The thread 0x3b20 has exited with code 0 (0x0).

The thread 0xb7c has exited with code 0 (0x0).

[2022-07-26T00:13:16] KinetisUpdater 0: Flashing Chip - PASS

[2022-07-26T00:13:16] KinetisUpdater 0: Flash complete, switching device mode

[2022-07-26T00:13:16] Kinetis Device: tx-F400

[2022-07-26T00:13:16] KinetisUpdater 0: Next update step [NONE]

[2022-07-26T00:13:16] KinetisUpdater 0: 14121ms to flash the device

[2022-07-26T00:13:16] HidDevice: Killing rx thread. Too many errors

[2022-07-26T00:13:16] HidDevice: Stopping receive thread

[2022-07-26T00:13:16] Kinetis Device: Requesting MCU ID records from database [9191000125D6001C0018800166324E45]

Checking for mico 9191000125D6001C0018800166324E45

Main - Device Changed | 4132712 | 537 | 7 | 0

[2022-07-26T00:13:16] Kinetis Device: UsbEvent: Starting Event [537]count: 1

Main - Device Changed | 4132712 | 537 | 7 | 0

[2022-07-26T00:13:17] Kinetis Device: UsbEvent: Connection status did not change [True]

[2022-07-26T00:13:17] Kinetis Device: UsbEvent: Completed USB eventcount: 1

[2022-07-26T00:13:17] Kinetis Device: UsbEvent: Starting Event [537]count: 2

[2022-07-26T00:13:17] Kinetis Device: UsbEvent: Connection status did not change [True]

[2022-07-26T00:13:17] Kinetis Device: UsbEvent: Completed USB eventcount: 2

SQL - deviceCheckProgramming - Took 1946ms

**How I found out that UpdatedCompletedUI()'s execution time has an effect on the UsbChangeEvent() being called**

If we call UpdateCompleteEvent() after update complete and make the UpdatedCompletedUI() to threadsleep( bout 2-3 seconds). The "Main - Device Changed | 331564 | 537 | 7 | 0" call backs will never be called in the log.

That means the UsbEvent() will never be called if thread sleep on the thread spawned by the UpdateCompleteEvent() as you can see in the log below. This is because UpdatedCompletedUI() runs on the UI thread and it blocks **UsbChangeEvent(IntPtr hwnd, int msg, IntPtr wParam, IntPtr lParam, ref bool handled) f**rom running which is also suppose to run on the UI thread(main thread).

[2022-07-26T12:14:13] KinetisUpdater 0: Setting flashing in progress to true

[2022-07-26T12:14:13] KinetisUpdater 0: Starting update step [FLASH\_SWAP\_BIT\_APPLICATION] using file [C:\Projects\PC\_Software\UWFU\_GitLab\ufwu\_application\UniversalFreescaleUpdater\bin\Debug\_Dev\UpdateFiles\RP4.2-TY11\4.1.4.47\RP4.2-TY11\_V47.pufx]

[2022-07-26T12:14:13] KinetisUpdater 0: Attempting to erase the chip...

[2022-07-26T12:14:13] Kinetis Device: tx-F000

Exception thrown: 'System.IO.IOException' in mscorlib.dll

[2022-07-26T12:14:14] Kinetis Device: rx-F1010000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

[2022-07-26T12:14:14] KinetisUpdater 0: Erasing Chip - PASS

[2022-07-26T12:14:14] KinetisUpdater 0: Attempting to flash the chip...

The thread 0x66f0 has exited with code 0 (0x0).

The thread 0x2ed4 has exited with code 0 (0x0).

The thread 0x82e4 has exited with code 0 (0x0).

The thread 0x3d48 has exited with code 0 (0x0).

[2022-07-26T12:14:26] KinetisUpdater 0: Flashing Chip - PASS

[2022-07-26T12:14:26] KinetisUpdater 0: Flash complete, switching device mode

[2022-07-26T12:14:26] Kinetis Device: tx-F400

[2022-07-26T12:14:26] KinetisUpdater 0: Next update step [NONE]

[2022-07-26T12:14:26] KinetisUpdater 0: 12632ms to flash the device

[2022-07-26T12:14:26] HidDevice: Killing rx thread. Too many errors

[2022-07-26T12:14:26] HidDevice: Stopping receive thread

[2022-07-26T12:14:26] Kinetis Device: Requesting MCU ID records from database [9191000125D6001C0018800166324E45]

Checking for mico 9191000125D6001C0018800166324E45

SQL - deviceCheckProgramming - Took 2215ms

**Why calling the ReconnectDevice() does not help us.**

Now WE called KinetisDevice.ReconnectDevice() from UpdatedCompletedUI() or from the KinetisUpdater Thread. But due to the complexities and timing issues between UsbEvent() and the ReconnectDevice() , the reconnect behavior is very unpredictable.

**Final conclusion**

we can see that "Main - Device Changed | 331564 | 537 | 7 | 0" is called only twice for Swap bit to Swap bit update.

But form bootloader to bootloader, Bootloader to swapbit the Main - Device Changed | 331564 | 537 | 7 | 0 is called 3 times, once for disconnect and two for re-connect.

But we see only two Main - Device Changed | 331564 | 537 | 7 | 0 at most being called( if we don’t thread sleep on UpdatedCompletedUI(status), non if we thread sleep ). This is because the of the KinetisUpdater.UpdateTask() calls UpdateCompleteEvent() which in turn calls Dispatcher.BeginInvoke(UpdatedCompletedUI(status);). When we call a function using **Dispatcher.BeginInvoke** that function runs on the UI thread.

The main page USB event which prints "Main - Device Changed | 331564 | 537 | 7 | 0" also runs on the UI Thread(Main thread), So this event gets blocked from being called. If we thread sleep in the UpdatedCompletedUI(status) all three USB events will be blocked.

The at least first Main - Device Changed | 331564 | 537 | 7 | 0 which is corresponding to the USB disconnect event is crucial for the UsbEvent() to determine that the device has disconnected. So that it will attempt to connect again by calling **CheckForNewDeviceConnection**(). This connection creation from **CheckForNewDeviceConnection**() may happen in the first UsbEvent() it self if the device has rebooted and reconnected or it may happen in the subsequent 2 UsbEvent() call that will follow after the device has rebooted and reconnected.

*For more details about the Dispatcher.BeginInvoke*

*See my writeup about the Dispactcher in the C# notes(Kasun's Notebook). I was able to recreate this effect in a stand alone WPF application.*