**# Title:**

Clarify Uart() DevicePath text conversion formats

**# Status:**

Draft

**# Document:**

UEFI Specification 2.9 (Future Errata)

**# License:**

SPDX-License-Identifier: CC-BY-4.0

**# Submitter:**

* Samer El-Haj-Mahmoud, ARM
* TianoCore Community (<https://www.tianocore.org>)

**# Summary of the change**

**Problem Statement:**

Refer to See UEFI Spec Table 105. The UEFI Spec allows for Uart() device paths text to be encoded with Parity/StopBits being either Integer values (0-255) or enumerated keywords:

- Parity : "D"=0, "N"=1, "E"=2, "O"=3, "M"=4, "S"=5

- StopBits: "D"=0, "1"=1, "1.5"=2, "2"=3

The issue is that firmware may encode the DevicePath text with StopBits / Parity using either their integer values OR the keywords, allowing for four possible combinations:

1. Integer Parity + Integer StopBits, e.g. : L"Uart(115200,8,0,2)"

2. Integer Parity + Keyword StopBits, e.g. : L"Uart(115200,8,0,1.5)"

3. Keyword Parity + Integer StopBits, e.g. : L"Uart(115200,8,D,2)"

4. Keyword Parity + Keyword StopBits, e.g. : L"Uart(115200,8,D,1.5)"

This becomes an issue when converting the DevicePath from/to Text/binary, which may be non-deterministic in some cases. Consider for instance L"Uart(115200,8,D,2)". Is the "2" here using the Integer StopBit value 2 (which corresponds keyword "1.5"), or the keyword "2" (which corresponds to integer value 3)?

**Proposal:**

One proposal is to restrict implementations to either using integers or keywords for both Parity and StopBits in the same firmware implementation. This makes combinations (2) and (3) above illegal, and allows firmware to implement either (1) or (4) :

1. Integer Parity + Integer StopBits, e.g. : L"Uart(115200,8,0,2)"

4. Keyword Parity + Keyword StopBits, e.g. : L"Uart(115200,8,D,1.5)"

With this restriction, the Parity can be used as a hint to whether the device path is using keywords or integers (since there is no overlap between currently defined Parity integer and keyword values). And based on that hint, the StopBits can be interpreted using the same format (integer or keyword).

**# Benefits of the change**

Clarifies corner case in the UEFI spec

**# Impact of the change**

* **Platform FW:**
  + Firmware implementations of DevicePathToText / DevicePathFromText may need to be updated to either use keywords or integers when building/converting Uart() device path nodes, but not mix the two.
    - EDK2 may be ok: <https://github.com/tianocore/edk2/blob/master/MdePkg/Library/UefiDevicePathLib/DevicePathToText.c#L1465> and <https://github.com/tianocore/edk2/blob/master/MdePkg/Library/UefiDevicePathLib/DevicePathFromText.c#L1964>
    - UBoot (<https://github.com/u-boot/u-boot/blob/master/lib/efi_loader/efi_device_path_to_text.c#L121>) may need to be updated
* **Operating Systems:**
  + None
* **Compliance tests**
  + SCT test code (<https://github.com/tianocore/edk2-test/blob/master/uefi-sct/SctPkg/TestCase/UEFI/EFI/Protocol/DevicePathFromText/BlackBoxTest/DevicePathFromTextBBTestCoverage.c#L2722>) should be updated to allow firmware implementations to use either keywords or integers (but not mix them) when testing the DP To/From Text conversion.

**# Detailed description of the change [normative updates]**

* Insertions highlighted
* Removals in ~~red~~

10.6.1.6 Text Device Node Reference

…

Table 10-62 Device Node Table

|  |  |
| --- | --- |
| … |  |
| Type: 3 (Messaging Device Path)  SubType: 14 (UART) | **Uart**(*Baud*, *DataBits*, *Parity*, *StopBits*)  The *Baud* is a 64-bit integer and is optional. The default value is 115200.  The *DataBits* is an integer from 0 to 255 and is optional. The default value is 8.  The *Parity* is either an integer from 0-255 or else a keyword and should be **D** (0), **N** (1), **E** (2), **O** (3), **M** (4) or **S** (5). It is optional. The default value is 0.  The *StopBits* is a either an integer from 0-255 or else a keyword and should be **D** (0), **1** (1), **1.5** (2), **2** (3). It is optional. The default value is 0.  *Parity* and *StopBits* can either be two integers or two keywords. Mixing formats is prohibited. |
| … |  |