

Intel VROC UEFI Physical Drive Location

Intel Confidential

VROC Rev 8.0
Revision: 1.0.0
November 2022

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

**Other names and brands may be claimed as the property of others.*

Copyright © 2021, Intel Corporation. All rights reserved.

Contents

Contents **3**

Revision History **4**

1 Overview **5**

2 Protocol implementation **6**

2.1 Variable Name 6

2.2 Vendor GUID 6

2.3 Variable data 6

2.4 Examples 9

2.4.1 NVMe 9

2.4.2 SATA 12

3 Physical Drive Location implementation **14**

3.1 NVMe drive service label 14

3.2 SATA drive service label 14

Revision History

Revision	Description	Date
0.1	Initial version	September 2021
0.7	Incorporate feedback from stakeholder review.	October 2021
0.7.1	Incorporate feedback from stakeholder review. - LocationOrdinalValue - ServiceLabel format fix	October 2021
0.7.2	Incorporate feedback from stakeholder review. Extend the example section for NVMe to describe the AIC drive case.	October 2021
0.7.3	Update to the Variable data section: - Separate structs for VROC_PORT_ADDRESS, - Separate structs for VROC_DRIVE_ADDRESS. Update the exmple for SATA and NVMe.	November 2021
0.7.4	Update the “Physical Drive Location implementation” section with details related to controller physical location.	December 2021
0.7.5	Specify the format of ServiceLabel and OemSkuLabel - the last char shall be 0x00.	December 2021
0.7.6	Update examples for SATA and NVMe	May 2022
1.0.0	PV milestone	November 2022

Table 1 Revision History

1 Overview

This document oulines the architecture that will enable Intel VROC to report the physical location of the drives directly attached to the platform. When implemented the Intel VROC will be able to report the physical location of the platform drives in the UEFI HII user interface.

For platforms that do not have this feature implemented, Intel VROC reports the location of the drives based on how the Intel VMD controller discovers and reports the drives. For example, in the Intel VROC HII, the drives are reported, to the user, as "Port 5:0, Slot 13, CPU0, VMD4, BDF 81:00.0". This information may not be very user-friendly.

The challenge is that there is no way for Intel VROC to know how a particular manufacturer has designed and layed out the drive connections.

To make this information more user-friendly, the architecture outlined in the document provides an interface for the manufacturer to provide that physical location in formation to Intel VROC. The information that the user will see in the Intel VROC HII will match the information provided via this interface.

2 Protocol implementation

The interface used to provide the information to Intel VROC is a UEFI variable. The UEFI variable will hold a predefined structure with all the information required for the manufacturer to implement the physical drive location functionality.

2.1 Variable Name

```
#define VROC_CONF_VAR_NAME L“VrocPhysicalLocation”
```

2.2 Vendor GUID

```
#define VROC_CONF_GUID \
{0x193dfefa, 0xa445, 0x4302, 0x99, 0xd8, 0xef, 0x3a, 0xad, 0x1a, 0x04, 0xc6}
```

Note: The Intel VROC driver already uses this GUID to expose other UEFI variables.

2.3 Variable data

```
#define VROC_PHYSICAL_LOCATION_VERSION      1

/* SATA controllers (SATA, sSATA, tSATA) */
#define VROC_SATA_CTRL_LIMIT        3
#define VROC_SATA_DRIVE_LIMIT       8
#define VROC_SATA_PORTS_LIMIT       VROC_SATA_DRIVE_LIMIT

#define VROC_NVME_DRIVE_LIMIT       96
#define VROC_NVME_PORTS_LIMIT       VROC_NVME_DRIVE_LIMIT

/*
 * Should fit "Port=1A:Box=3:Bay=4".
 * Shall end with the NULL(0x00) character.
 */
#define VROC_SERVICE_LABEL_STR_LEN  31
/*
 * Should fit the OEM Part Number used for VROC license type
 * Shall end with the NULL(0x00) character.
 */
#define VROC_SKU_LABEL_STR_LEN      17

/* VROC controller location ordinal values */
#define VROC_CTRL_LOCATION_ORDINAL_NVME     0
#define VROC_CTRL_LOCATION_ORDINAL_SATA     1
#define VROC_CTRL_LOCATION_ORDINAL_SSATA    2
#define VROC_CTRL_LOCATION_ORDINAL_TSATA    3

#pragma pack(1)
/*
 * Defined in Redfish schema for LocationType (Resource.v1_13_0.json).
```

```
 */
typedef enum {
       SLOT,
       BAY,
       CONNECTOR,
       SOCKET,
       BACKPLANE,
       EMBEDDED
} VROC_LOCATION_TYPE;

/*
 * The location of a resource.
 */
typedef struct {
    /*
     * The label of the part location, for example "Port=1A, Box=3, Bay=4".
     * Shall end with the NULL(0x00) character.
     */
    CHAR8    ServiceLabel[VROC_SERVICE_LABEL_STR_LEN];
    /*
     * The types of locations for a part within an enclosure.
     */
    VROC_LOCATION_TYPE    LocationType;
    /*
     * The number that represents the location of the part.
     * For example, if LocationType is `Slot` and this unit is in slot 2, the
     * LocationOrdinalValue is `2`.
     */
    UINT32                        LocationOrdinalValue;
} VROC_PHYSICAL_LOCATION_INFO;

/*
 * Controller types supported by Intel® VROC.
 */
typedef enum {
    /*
     * NONE implies that the drive or port slot in the table is not used.
     */
    NONE,
    NVME,
    SATA
} VROC_CTRL_TYPE;

/*
 * The address below are used to match VROC ports with the OEM's provided location
 * information.
 */
typedef struct {
    VROC_CTRL_TYPE DriveType; // Indicates if the port is used.
    UINT8 CpuId;
    UINT8 RootPort;
```

```
} VROC_NVME_PORT_ADDRESS;
typedef struct {
    VROC_CTRL_TYPE DriveType; // Indicates if the port is used.
    UINT8 SataId;
    UINT8 SataPort;
} VROC_SATA_PORT_ADDRESS;
/*
 * The union is convenient to abstract interface in the VROC_PORT struct.
 */
typedef union {
    VROC_NVME_PORT_ADDRESS NvmePort;
    VROC_SATA_PORT_ADDRESS SataPort;
} VROC_PORT_ADDRESS;

/*
 * The address below are used to match VROC drives with the OEM's provided location
 * information.
 */
typedef struct {
    VROC_NVME_PORT_ADDRESS Port;
    /*
     * It is used to describe the drive PCIe port. For example, if the drive
     * PCIe port equals "1C," the offset will have a value of 2. The letter
     * at the end of the port address should be mapped to a number
     * value (0-based).
     */
    UINT8 RootPortOffset;
} VROC_NVME_DRIVE_ADDRESS;
/*
 * The union is convenient to abstract interface in the VROC_DRIVE struct.
 */
typedef union {
    VROC_SATA_PORT_ADDRESS Sata;
    VROC_NVME_DRIVE_ADDRESS Nvme;
} VROC_DRIVE_ADDRESS;

/*
 * For NVMe, Port is equal VMD domain. For SATA, it is a single SATA controller
 * port.
 */
typedef struct {
    VROC_PORT_ADDRESS PortAddress;
    VROC_PHYSICAL_LOCATION_INFO PortPhysicalLocation;
} VROC_PORT;

/*
 * Used for NVMe and SATA drives.
 */
typedef struct {
    VROC_DRIVE_ADDRESS DriveAddress;
    /*
```

```
     * Indicates if the drive was available on the boot time.
     * Note: the UEFI EDK2 environment defines BOOLEAN as uint8_t
     */
    BOOLEAN DrivePresent;
    VROC_PHYSICAL_LOCATION_INFO DrivePhysicalLocation;
} VROC_DRIVE;

typedef struct {
    VROC_PORT SataPorts[VROC_SATA_PORTS_LIMIT];
    VROC_DRIVE SataDrives[VROC_SATA_DRIVE_LIMIT];
    VROC_PHYSICAL_LOCATION_INFO CtrlPhysicalLocation;
} VROC_SATA_CTRL;

typedef struct {
    VROC_PORT NvmePorts[VROC_NVME_PORTS_LIMIT];
    VROC_DRIVE NvmeDrives[VROC_NVME_DRIVE_LIMIT];
    VROC_PHYSICAL_LOCATION_INFO CtrlPhysicalLocation;
} VROC_NVME_CTRL;

typedef union {
    struct {
        UINT32 Version;
        VROC_NVME_CTRL NvmeCtrl;
        VROC_SATA_CTRL SataCtrls[VROC_SATA_CTRL_LIMIT];
        /*
         * The OEM "Part Number" - defined for the VROC license type.
         * Shall end with the NULL(0x00) character.
         */
        CHAR8 OemSkuLabel[VROC_SKU_LABEL_STR_LEN]
    } v1;
} VROC_PHYSICAL_LOCATION;
#pragma pack()
```

2.4 Examples

2.4.1 NVMe

Below is an example with the data in the JSON format to illustrate how OEMs/ODMs can map Intel VMD addressing to the required physical location format for a scenario with the two drives:

- drive connected to the PCIe port "0C" (0-based) on CPU 0 - connected to the drive enclosure (Box=1, Bay=4),
- drive connected to the PCIe port "2A" (0-based) on CPU 0 - add-in-card SSD.
- drive connected to the PCH and port 9 - connected to the internal M.2 slot (e.g. “Embedded:Port=9A”)

```
{
      "NvmeCtrl": {
        "NvmePorts": [
```

```
            {
              "PortAddress": {
                   "NvmePort": {
                     "CpuId": 0,
                     "DriverType": 1, // Nvme
                     "RootPort": 0 // 0-based, related to “0C”
                   }
              },
              "PortPhysicalLocationInfo": {
                   "LocationOrdinalValue": 1,
                   "LocationType": 2, // Connector
                   "ServiceLabel": "Port=0"
              }
            },
            {
              "PortAddress": {
                   "NvmePort": {
                     "CpuId": 0,
                     "DriverType": 1, // Nvme
                     "RootPort": 2 // 0-based, related to “2A”
                   }
              },
              "PortPhysicalLocationInfo": {
                   "LocationOrdinalValue": 2,
                   "LocationType": 2, // Connector
                   "ServiceLabel": "Port=2"
              }
            },
            {
              "PortAddress": {
                   "NvmePort": {
                     "CpuId": 15, // value used for the PCH
                     "DriverType": 1, // Nvme
                     "RootPort": 0 // value used for the PCH
                   }
              },
              "PortPhysicalLocationInfo": {
                   "LocationOrdinalValue": 3,
                   "LocationType": 2, // Connector
                   "ServiceLabel": "Embedded"
              }
            }
        ],
        "NvmeDrives": [
            {
              "DriveAddress": {
                   "Nvme": {
                     "Port": {
                          "CpuId": 0,
                          "DriverType": 1, // Nvme
                          "RootPort": 0 // 0-based, related to “0C”
```

```
                    },
                    "RootPortOffset": 2 // related to “0C”
                }
            },
            "DrivePresent": true,
            "DrivePhysicalLocation": {
                "LocationOrdinalValue": 3,
                "LocationType": 1, // Bay
                "ServiceLabel": "Port=0C:Box=1:Bay=3"
            }
        },
        {
            "DriveAddress": {
                "Nvme": {
                    "Port": {
                        "CpuId": 0,
                        "DriverType": 1, // Nvme
                        "RootPort": 2 // 0-based, related to “2A”
                    },
                    "RootPortOffset": 0 // related to “2A”
                }
            },
            "DrivePresent": true,
            "DrivePhysicalLocation": {
                "LocationOrdinalValue": 2,
                "LocationType": 2, // Connector
                "ServiceLabel": "Port=2A:Box=1:Bay=3"
            }
        }
        {
            "DriveAddress": {
                "Nvme": {
                    "Port": {
                        "CpuId": 15, // value used for the PCH
                        "DriverType": 1, // Nvme
                        "RootPort": 0 // value used for the PCH
                    },
                    "RootPortOffset": 9 // related to port 9
                }
            },
            "DrivePresent": true,
            "DrivePhysicalLocation": {
                "LocationOrdinalValue": 3,
                "LocationType": 2, // Connector
                "ServiceLabel": "Embedded:Port=9A"
            }
        }
    ],
    "CtrlPhysicalLocation": {
        "LocationOrdinalValue": 0,
        "LocationType": 5, // Embedded
```

```
            "ServiceLabel": "Embedded=NVMe"
          }
        },
        "Version": 0
}
```

2.4.2 SATA

Below is an example with the data in the JSON format to illustrate how OEMs/ODMs can map Intel RSTe addressing to the required physical location format for a drive connected to the sSATA controller on port 4.

```
{
        "SataCtrls": [
        {
                "CtrlPhysicalLocation": {
                  "LocationOrdinalValue": 2,
                  "LocationType": 5, // Embedded
                  "ServiceLabel": "Embedded=SSATA"
                },
                "SataDrives": [
                  {
                       "DriveAddress": {
                         "Sata": {
                              "DriverType": 2, // Sata
                              "SataId": 1,
                              "SataPort": 4 // related to port 4
                         }
                       },
                       "DrivePresent": true,
                       "DrivePhysicalLocation": {
                         "LocationOrdinalValue": 4,
                         "LocationType": 1, // Bay
                         "ServiceLabel": "Port=4:Box=1:Bay=4"
                       }
                  }
                ],
                "SataPorts": [
                  {
                       "PortAddress": {
                         "SataPort": {
                              "DriverType": 2, // Sata
                              "SataId": 1,
                              "SataPort": 4 // related to port 4
                         }
                       },
                       "PortPhysicalLocationInfo": {
                         "LocationOrdinalValue": 4,
                         "LocationType": 2, // Connector
                         "ServiceLabel": "Port=4"
```

```
                    }
                }
            ]
        }
    ],
    "Version": 0
}
```

3 Physical Drive Location implementation

3.1 NVMe drive service label

Intel VROC will take the physical drive location value directly from the "NvmeDrives.DrivePhysicalLocation.ServiceLabel" field in VROC_PHYSICAL_LOCATION.

There is no additional concatenation required of the Service Label values for the connected Port and Controllers.

For the example shown in 2.4.1, the final Service Label value will be "Port=1C:Box=1:Bay=3".

3.2 SATA drive service label

Intel VROC will take the physical drive location value directly from the "SataDrives.DrivePhysicalLocation.ServiceLabel" field in VROC_PHYSICAL_LOCATION.

There is no additional concatenation required of the Service Label values for the connected Port and Controllers.

For the example shown in 2.4.2, the final Service Label value will be "Port=1:Box=1:Bay=4".

3.3 NVMe controller physical location

Intel VROC will use the following physical location values for the NVMe strorage controller resource:

- "LocationType": 5, // Embedded
- "ServiceLabel": "Embedded=NVMe"

The “LocationOrdinalValue” will not be used.