x0003 - device information

Version 4

DeviceInformation

This feature provides model and unit specific information

[0] **getDeviceInfo**() → entityCnt, unitId, transport, modelId, extendedModelId, capabilities
[1] **getFwInfo**(entityIdx) → type, fwName, rev, build, active, trPid, extraVer
[2] **getDeviceSerialNumber**() → serialNumber

Overview

Allows obtaining information that enables the host software to uniquely identify a device as well as the firmware versioning information for the different firmware entities that a device may comprise.

Functions and Events

[0] getDeviceInfo() → entityCnt, unitId, transport, modelId, extendedModelId, capabilities

Returns the number of firmware and hardware entities.

Returns information that characterises the whole device (as opposed to a particular firmware entity). A device may hold multiple firmware entities, including multiple main applications and bootloader.

The response to this request 'does not depend' on the transport protocol used to carry the request, this is very useful when the host SW wants to identify a device that supports more than one communication protocol.

Parameters

none

Returns

entityCnt

The number of entities present in the device from which this feature can obtain version information.

unitId

Four byte random array that serves as per unit identifier, guaranteed* to be unique for all units with the same modelId.

> *The probability of having more than one unit with same unitId value in a single users' setup is almost zero.

transport

A bitfield indicating which communication protocols the device supports.

modelId

Six bytes array. The model id is a dense array of the application PIDs of the different transports supported by the device. For every bit set in the 'transport' bitfield, starting from bit 0, and up to bit 15, a 16-bit transport specific ID (PID) will be appended to the modelId. + The PIDs are appended to the modelId array starting from the position bytes[7..8] and up to position bytes[11..12] this means that a device could support up to three simultaneous transports.

extendedModelId

A 8 bit value that represents a configurable attribute of the device (on the line) for a given modelId (e.g. colour of the device). Default value is 0.

> *It is TDE's responsibility to write the extendedModelId in the device non-volatile memory via a TDE specific command. This value has to match the extendedModelId in the SW so that SW displays the proper information on the screen (e.g. device colour).

capabilities

A 8 bit value that represents additional capabilities of this HIDPP feature.

```
serialNumber = 0 -> serial Number is not supported.
serialNumber = 1 -> serial Number is supported.
```

Table 1. getDeviceInfo() response packet

byte \ bit	7	6	5	4	3	2	1	0
0	entityCnt							
1	unitId_3 (MSB)							
4	unitId_0 (LSB)							
5	transport_1 (MSB) (reserved)							
6	transport_0 (LSB)							
	---	---	---	---	USB	eQuad	BTLE	BT
7	modelId_5 (MSB)							
12	modelId_0 (LSB)							
13	extendedModelId							
14	capabilities							

byte \ bit	7	6	5	4	3	2	1	0
	---	---	---	---	---	---	---	serialNumber
15	reserved							

When a bit in the transport word is set, the device supports the corresponding transport. If the transport is not supported by the device the bit is clear. A device can support a maximum of three simultaneous transports.

[1] getFwInfo(entityIdx) → type, fwName, rev, build, active, trPid, extraVer

Returns the firmware version for the given entityIdx.

Parameters

entityIdx

The index of the firmware or hardware entity for which we want to obtain the version information.

Table 2. getFwInfo() request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	entityIdx							

Returns

type

- Indicates the entity type for the requested entity index:
  - 0 - Main application (provides full functionality to a device)
  - 1 - Bootloader (provides DFU functionality)
  - 2 - Hardware
  - 3 - Touchpad (used to know touchpad FW version)
  - 4 - Optical sensor (used to know optical sensor FW version)
  - 5 - Softdevice (Nordic’s SoftDevice version)
  - 6 - RF companion MCU (used to know RF companion MCU FW version)
  - 7 - Factory application (provides full functionality to a device and handles the DFU process, but is not upgradable)
  - 8 - RGB Custom Effect (storage for a custom RGB effect)
  - 9 - Motor drive (provides an HIDPP and DFU capability for a companion MCU chip for motor drive)
  - 10..15 - Reserved

fwName

The firmware name is composed of the prefix characters (also known as the firmware class) and a firmware number between 0 and 99. The firmware name uniquely links an application binary executable with an existing product. It also uniquely links the application binary to existing documents in manufacturing documentation.

The binary executable associated to a particular firmware name constitutes the smallest unit of in the field deployed code that can be updated via DFU.

A device can have more than one unit of updateable firmware, consequently having more than one firmware name, see multiprotocol devices below.

1. **Prefix characters:** The prefix characters are letters used to identify in a human readable fashion the type of device and the type of transport used by the device. The ascii value for each letter is used/stored.
2. **Firmware number:** This number between 0 and 99 is encoded in packed BCD format. For example this means that the decimal 34 is encoded (stored in memory) as a single byte whose hexadecimal value will be 0x34. Firmware numbers are incremented each time a new product within the same class is released.

rev

Firmware revision. It is a BCD packed number that gets incremented if a new QA approved firmware is released in the field, this can happen either via manufacturing or in the field DFU.

build

Firmware build. It a is packed BCD number that gets incremented for any firmware release (internal, engineering, external, etc) except for the entity type 5 (Softdevice).
For the Softdevice, it is the raw value.

active

This entity is the active entity responding to this hid++ request. If 0 then a different entity is the active entity. Exactly one entity will return 1 in this field.

trPid

Transport PID ("product ID" - For example: USB PID, BT PID, equadId etc.)

- If the 'active' field (just above) is 1:
  This is the PID presented by the device to its host when using the transport through which it received this getFwInfo() hidpp request.
- If the 'active' field is 0:
  This field may be filled with zeros, for example if different transports could be used to access the requested entity (see "Behaviour of multi protocol devices" section below). In case only one transport can possibly be used to access the requested entity, then the PID may be provided (but it is allowed to fill this with zeros also in that case).

extraVer

Optional extra versioning information. Five bytes that link the executing binary to the source control system. When not implemented these bytes must be 0x00.

Table 3. getFwInfo() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	type							

byte \ bit	7	6	5	4	3	2	1	0
1	fwName (Prefix character 0)							
2	fwName (Prefix character 1)							
3	fwName (Prefix character 2)							
4	fwName (Firmware number)							
5	revision							
6	build (MSB)							
7	build (LSB)							
8	---	---	---	---	---	---	---	active
9	trPid (MSB)							
10	trPid (LSB)							
11	extraver_0							
	…							
15	extraver_4							

[2] getDeviceSerialNumber() → serialNumber

Returns the serial number of the device.

Parameters

none

Returns

serialNumber

Twelve bytes array. The serial number is serial number value in a Base34 Alphanumeric scheme (I and O will not be use and all are upper cases) in ASCII format. If the value is not set or the device doesn't support the serialNumber capability, the returned default value is 12 bytes set to 0 (binary). The serial number shall always be 12 bytes.

Table 4. getDeviceSerialNumber() response packet

byte \ bit	7	6	5	4	3	2	1	0

byte \ bit	7	6	5	4	3	2	1	0
0	serialNumber_11 (MSB)							
11	serialNumber_0 (LSB)							
12-15	reserved							

Behaviour of multi protocol devices:

Active link

A multi protocol device will mostly have a single link enabled at any given point in time. This transport is referred to as the active link. The current active link is the transport used by the device to send input reports as well as all hidpp traffic.

A multi protocol device can switch from one transport to a new one, in which case the new transport becomes the active link.

In the unlikely case that more than a single link is enabled at the same time, then the link used to convey the relevant hidpp transaction is the active link for the current transaction.

PIDs

Each implemented connectivity protocol (Bluetooth, BLE, eQuad, USB) has its own ID (BT PID, eQuad ID, USB PID,...).

Response to the [0] GetDeviceInfo() request

The response to the GetDeviceInfo request is always the same and independent of the transport used to perform the request.

Response to the [1] GetFwInfo() request

A multi protocol device answers to the 0x0003 GetFwInfo() request with a response that depends on the active link. In other words, the response will depend on which transport the device received the request through.

- for the trPid:
  - if the request is for the active entity, this is the PID presented by the device to its host when using the transport through which it received this getFwInfo() hidpp request.
  - if the request is for an inactive entity, because it is impossible to foresee which one of the multiple transports will be used when this entity will be active, this field shall be filled with zeros.
- for the FW name, version and build, there are two cases explained below.

Caching of the device hidpp enumeration by the host SW:

The host software can cache the hidpp enumeration (present hidpp features, indexes and versions) to avoid re-enumerating the hidpp features of a device every time it reconnects.

On device reconnection, the host SW can use the information returned by GetDeviceInfo() / GetFwInfo() to decide to invalidate the cache contents, if the FW name, version or build for active application have changed compared to the previously cached information.

ChangeLog

- Version 4: Add Serial Number support.
- Version 3: Add new FW type field for motor drive
- Version 2: Add extended model ID in information API in order to get configurable attribute of the device such as color for a specific modelId.
- Version 1: Extends function [0]getDeviceInfo() to return unitId, transport bitfield and modelId values. Specifies the behaviour of multiprotocol devices. Version 0 of getDeviceInfo() only returns the entity count for a given device, version 0 did not support full multiprotocol device specification.
- Version 0: Initial version