

## Appendix A: IOCTL codes used by the JUSB framework

IOCTL codes are used within the *DeviceIoControl* user mode API function to retrieve information from a device. These IOCTL codes allow programmers to access kernel mode functionality from within the user mode,. The following list presents all jUSB driver IOCTL codes, The supplied input structure and its corresponding output structure. In some cases, the input structure is the same as the output structure but this is not always the case.

### I: JUSB IOCTL codes

The following IOCTL codes are used in the jUSB driver and defined in `ioctl.h` file.

#### I a) IOCTL\_JUSB\_GET\_DEVICE\_DESCRIPTOR

Returns the device descriptor in the output buffer. The output buffer must have the size of the `USB_DEVICE_DESCRIPTOR` structure. The input buffer is not being treated, so we can put this parameter to `NULL`. The members of this structure are described in the DDK or on the online MSDN documentation [19].

```
typedef struct _USB_DEVICE_DESCRIPTOR {
    UCHAR bLength ;
    UCHAR bDescriptorType ;
    USHORT bcdUSB ;
    UCHAR bDeviceClass ;
    UCHAR bDeviceSubClass ;
    UCHAR bDeviceProtocol ;
    UCHAR bMaxPacketSize0 ;
    USHORT idVendor ;
    USHORT idProduct ;
    USHORT bcdDevice ;
    UCHAR iManufacturer ;
    UCHAR iProduct ;
    UCHAR iSerialNumber ;
    UCHAR bNumConfigurations ;
} USB_DEVICE_DESCRIPTOR, *PUSB_DEVICE_DESCRIPTOR ;
```

**Table 50: USB\_DEVICE\_DESCRIPTOR structure**

#### I b) IOCTL\_JUSB\_GET\_CONFIGURATION\_DESCRIPTOR

Returns the *ith* configuration descriptor of the device. Two steps are necessary to successfully execute this IOCTL code. The input and output buffer belongs to the *DeviceIoControl* WinAPI function.

##### 1. Step

Input buffer: Index of type `USHORT`

Output buffer: `NULL`

`nReturnedBytes`: Contains the length of the *ith* configuration descriptor

##### 2. Step

Input buffer: Index of type `USHORT`

Output buffer: The size of `nReturnedBytes`, type of `UCHAR`

`nReturnedBytes`: Contains the length of the *ith* configuration descriptor

#### I c) IOCTL\_JUSB\_GET\_STRING\_DESCRIPTOR

Retrieves the string descriptor from a given index and language. To dynamically allocate memory for the string descriptor we will need two *DeviceIoControl* calls. The first call returns the `USB_STRING_DESCRIPTOR` structure without the driver key name, but tells in the `bLength` member, how many bytes the string descriptor needs. In a second call we provide a buffer big enough to hold the

entire length of the string descriptor and the `STRING_REQUEST` structure. To tell the driver which string descriptor we look for a `STRING_REQUEST` structure is always put at the beginning of the input and output buffer to send the input parameters to the jUSB driver.

```
typedef struct _STRING_REQUEST{
    UCHAR ucDescriptorIndex;
    USHORT usLangId;
} STRING_REQUEST, *PSTRING_REQUEST;

typedef struct _USB_STRING_DESCRIPTOR {
    UCHAR bLength ;
    UCHAR bDescriptorType ;
    WCHAR bString[1] ;
} USB_STRING_DESCRIPTOR, *PUSB_STRING_DESCRIPTOR ;
```

**Table 51: STRING\_REQUEST and USB\_STRING\_DESCRIPTOR structures**

#### I d) IOCTL\_JUSB\_GET\_STATUS

Returns the status for the specified recipient (`bmRequestType`) which is always two bytes (further information can be found in the USB specification chapter 9.4.5). The input buffer consists of two bytes of type `USHORT`. The first byte contains the `bmRequestType` and the second byte the `wIndex` field. The output buffer will contain the data that is returned by the `GET_STATUS` request.

#### I e) IOCTL\_JUSB\_INTERRUPT\_TRANSFER

This IOCTL code invokes the jUSB driver to do a synchronously interrupt request to the driver. The input buffer must be as big as the number of bytes we want to read. To tell the driver of which endpoint we want to read, we set the first byte of the input buffer with the endpoint address. When successfully complete the request the output buffer contains the bytes read. For more information look at the source, which is in the `DoInterruptTransfer` function in the `Control.c` file.

### II: Other IOCTLs

The following IOCTL:

- `IOCTL_USB_GET_NODE_CONNECTION_NAME`
- `IOCTL_USB_GET_NODE_CONNECTION_INFORMATION`
- `IOCTL_USB_GET_NODE_CONNECTION_DRIVERKEY_NAME`
- `IOCTL_USB_GET_NODE_INFORMATION`
- `IOCTL_USB_GET_DESCRIPTOR_FROM_NODE_CONNECTION`
- `IOCTL_USB_GET_ROOT_HUB_NAME`

are applied in the `usbview` example (DDK) or very clearly arranged in an example by John Hide which is online available on:

[www.intel.com/intelpress/usb/examples/DUSBVC.PDF](http://www.intel.com/intelpress/usb/examples/DUSBVC.PDF)

We do not explain this IOCTL codes any further.