ModusToolbox™ USB Configurator user guide

ModusToolbox™ tools package version 3.8.0
USB Configurator version 3.20.0

About this document

A newer revision of this document may be available on the web here.

Scope and purpose

The USB Configurator main use is to create device descriptor configuration and generate code with the descriptor tables as part of the device firmware.

Intended audience

This document helps application developers understand how to use the USB Configurator as part of creating a ModusToolbox™ application.

Document conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, menus and sub-menus
Italics	Denotes file names and paths.
`Monospace`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
File > New	Indicates that a cascading sub-menu opens when you select a menu item

Abbreviations and definitions

The following define the abbreviations and terms used in this document:

- API – application programming interface
- BOS – binary device object store
- CDC – communication device class
- Configurator – A GUI-based tool used to configure a resource.
- Descriptor – A defined-format data structure used by USB devices to report their attributes to a USB host or in other words a piece of stored data that indicates how other data is stored.
- Endpoint – A uniquely addressable portion of a USB device that is the source or sink of information in a communication flow between the host and device.
- HID – human interface device
- USB – universal serial bus device

Reference documents

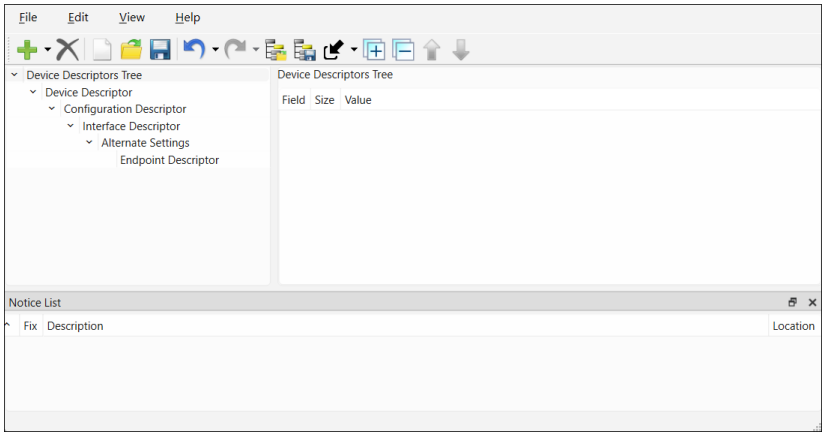
Refer to the following documents for more information as needed:

- ModusToolbox™ tools package user guide
- Eclipse IDE for ModusToolbox™ user guide
- VS Code for ModusToolbox™ user guide
- Device Configurator user guide
- USB device middleware library
- http://www.usb.org

Table of contents

1 Overview

The Universal serial bus (USB) Configurator is a stand-alone tool included with the ModusToolbox™ software to configure USB device descriptors. See the Supported descriptors section for a list of supported USB descriptors. After configuring and saving a USB device descriptor, the USB Configurator generates configuration files that store USB device descriptors and other information (see Code generation used by the USB device middleware configuration and operation.



1.1 Supported middleware

Name	Link
USB device middleware library	https://github.com/Infineon/usbdev

2 Launch the USB Configurator

There are several ways to launch the USB Configurator, and those ways depend on how you use the various tools in ModusToolbox™.

2.1 make command

As described in the ModusToolbox ™ tools package user guide "Build System" chapter, you can run numerous make commands in the application directory, such as launching the USB Configurator. After you have created a ModusToolbox™ application, navigate to the application directory and type the following command in the appropriate bash terminal window:

```
make usbdev-configurator
```

This command opens the application USB Configurator GUI for an existing USB configuration (*.*cyusbdev*) file.

If there is no *. *cyusbdev* file, then this command launches the USB Configurator with default configuration. Open an existing *.*cyusbdev* file or create a new one for the application in which you want to configure the USB Configurator.

2.2 VS Code and Eclipse

VS Code and Eclipse have tools to launch the USB Configurator from within an open application. Refer to the applicable user guide for more details:

- VS Code for ModusToolbox™ user guide
- Eclipse IDE for ModusToolbox™ user guide

2.3 Executable (GUI)

Also, you can simply launch the USB Configurator GUI by running its executable as appropriate for your operating system (for example, double-click it or select it using the Windows **Start** menu). By default, it is installed here:

<install_dir>/ModusToolbox/tools_<version>/usbdev-configurator-<version>

On Windows, launch the tool from the **Start** menu. For other operating systems, navigate to the install location and run the executable. The USB Configurator opens with an untitled configuration file (*.*cyusbdev*). Save it as a new file and provide a file name, or open another existing *.*cyusbdev* file.

2.4 Executable (CLI)

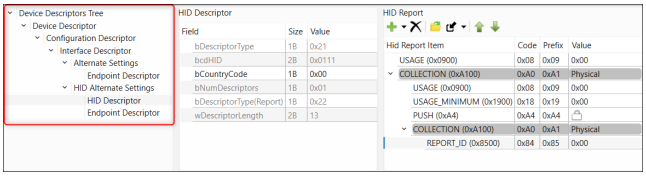
You can run the usbdev-configurator executable from the command line. There is also a usbdev-configurator-cli executable, which re-generates the source code based on the latest configuration settings from a command-line prompt or from within batch files or shell scripts. The exit code for the usbdev-configurator-cli executable is zero if the operation is successful, or non-zero if the operation encounters an error. To use the usbdev-configurator-cli executable, you must provide at least the `--config` argument with a path to the configuration file.

For more information about the command-line options, run the usbdev-configurator or usbdev-configurator-cli executable using the -h option.

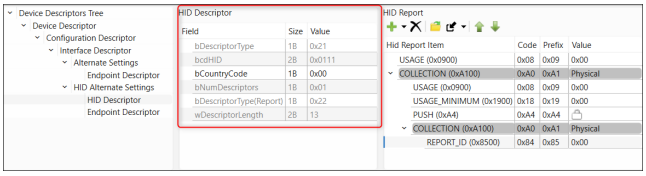
3 Quick start

This section provides a simple workflow for how to use the USB Configurator.

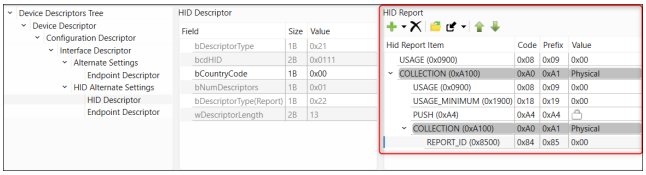
1. Launch the USB Configurator.
2. Configure the device descriptors hierarchy in the Device Descriptors Tree pane pane.



3. Configure device descriptor parameters in the Parameters pane.



4. The parameters pane contains a subpane for HID descriptor (HID Report pane).



5. Save the configuration.

   See Code generation.
6. Use the generated structures as input parameters for functions in your application.

4 Code generation

The USB Configurator generates header (.h) and source (.c) files that contain relevant firmware used by the USB Device middleware configuration and operation. The firmware contains arrays to store USB Device descriptors, structures that help the middleware to access descriptors, middleware and classes configuration structures, and a set of defines. The generated files *cycfg_usbdev.h* and *cycfg_usbdev.c* are located in the *GeneratedSource* folder next to the *. *cyusbdev* file.

Refer to the USB device middleware library for more information about this code.

5 GUI description

The USB Configurator GUI contains Menus, Toolbars, Panes, and a subpane used to configure device descriptors.

5.1 Menus

File

- **New** – Creates a new file with new configuration.
- **Open...** – Opens the configuration file.
- **Save** – Saves the existing file.
- **Save As...** – Saves the existing file under a different name.
- **Open in System Explorer** – Opens your computer's File Explorer browser to the folder that contains the *.*cyusbdev* file.
- **Load descriptor** – Loads a descriptor under the selected descriptor.
- **Save descriptor** – Saves the selected descriptor.
- **Recent files** – Shows recent files that you can open directly.
- **Exit** – Closes the configurator.

Edit

- **Undo** – Undoes the last action or sequence of actions.
- **Redo** – Redoes the last undone action or sequence of undone actions.

View

- **Notice List** – Shows/hides the Notice List. The Notice List pane displays by default.
- **Toolbar** – Shows/hides the toolbar.
- **Reset View** – Resets the view to the default.

Help

- **View Help** – Opens this document.
- **About USB Configurator** – Opens the About box for version information, with links to open https://www.infineon.com and the current session log file.

Notice List

The **Notice List** pane combines notices (errors, warnings, tasks, and infos) from many places in the configuration into a centralized list. You can double-click a notice location to show the parameter causing the error or warning. For more information about the Notice List, refer to Device Configurator user guide.

5.2 Toolbars

5.2.1 Descriptor toolbar

Provides the basic buttons from the Menus to create, open, edit, and save files.

Also, the descriptor toolbar provides buttons to configure the descriptors hierarchy:

- **Add new descriptor** – Click this button to create a new descriptor.
- **Delete selected descriptor** – Removes the selected descriptor.
- **Import descriptor** – Select a descriptor from the pull-down menu to transfer it into the configuration.

- **Expand all tree items** – Clicking this button displays all the items in the Device Descriptors Tree.
- **Collapse all tree items** – Clicking this button leaves only the Device Descriptors Tree visible.
- **Move up** – Shifts up the selected descriptor.
- **Move down** – Shifts down the selected descriptor.

5.2.2 HID Report toolbar

The HID Report toolbar provides the buttons to configure an HID descriptor report.

- **Add HID report item** – Creates a new HID report item.
- **Delete HID report item** – Removes the selected HID report item.
- **Load HID report** – Opens an HID report file into the current HID Descriptor.
- **Import HID report** – Transfers an HID report item into the configuration for the current HID Descriptor.
- **Move up HID report item** – Shifts up an IDHIFDHID report item.
- **Move down HID report item** – Shifts down an IDHIFDHID report item.

5.2.3 Array editor toolbar

The Array editor toolbar provides the buttons to edit array elements.

- **Add new element** – Creates a new element.
- **Delete element** – Removes the selected element.
- **Move up element** – Shifts up an element.
- **Move down element** – Shifts down an element.

5.3 Panes

The USB Configurator contains two panes that display information about descriptors and their parameters:

5.3.1 Device Descriptors Tree pane

This pane shows the descriptors hierarchy:

- All the descriptors have their own hierarchy that can be created when their parent is selected. However, the Device Descriptors Tree is the root descriptor.
- This pane shows the descriptors hierarchy:

To add specific descriptors, for example, CDC Interface descriptor or HID descriptor, add a special parent descriptor:

- Add CDC interface descriptor for CDC descriptor.
- Add HID alternate settings for HID descriptor.

5.3.2 Parameters pane

This pane shows configuration information for the selected descriptor.

***Note**: The parameters pane has different controls to edit different parameters (text box, combo box, or multi-line text box). Most parameters have a text box as the editing control.*

- Vendor-defined – Some string parameters or HID report items, such as **iSerialNumber**, have a combo box with a list of predefined items. The **Empty** value is selected by default. To specify a value that is not on the list, select **Vendor-defined**. The combo box will change to text box to type an appropriate value. To return to the combo box, erase the value and leave the control.
- String pool – Parameters such as iChannelNames in AC Processing Unit of Audio Interface descriptor 1.0 support a string pool and require a multi-line text box. To insert several strings, use an end-line separator.
- Read-only – There are two types of read-only parameters: predefined **bDescriptorType** or auto-calculated **bConfigurationValue**.
- Array – Parameters such as **bSubordinateInterface** in union with Communication Alternate Settings is an array. Use ";" to separate elements. Also, you can open the Array editor toolbar dialog to set the parameters using the **[. . .]** button.
- Hexadecimal/Decimal – When a value starts with **0x**, it is parsed as a hexadecimal value. Otherwise, it is parsed as a decimal.
- Map and bit fields – Some parameters, such as **bmAttributes**, are read-only by themselves. However, you can insert them using the related bit fields below them in the parameters list. The bit fields size is **0B**. Their name starts with the name of a field whole part, plus the bits for which they are responsible. For example, **bmAttributes(1-0): Transfer Type**.

6 Supported descriptors

All supported descriptors are described by USB Implementers Forum, Inc. You can find more details at http://www.usb.org.

The supported descriptors:

- Device, Configuration, IAD (Interface association descriptor), Interface, Endpoint descriptors, Endpoint Companion descriptors
- Device descriptor (Billboard), Device Qualifier descriptor
- Audio Interface descriptor 1.0 and 2.0
- CDC Interface descriptor
- HID descriptor and HID Report descriptor
- BOS descriptors:
  - USB 2.0 Extension descriptors
  - Container ID Descriptor
  - Billboard Capability Descriptor
  - Billboard Alternate Mode Capability Descriptor
  - SuperSpeed Device Capability Descriptor
  - SuperSpeedPlus Device Capability Descriptor
  - PTM Capability Descriptor
- MS OS String descriptor

***Note**: The Interface Descriptor is represented by the two items to build a tree structure: Interface Descriptor with an empty parameters pane and Alternate Settings with all Interface Descriptor parameters.*

***Note**: Class-Specific AS Encoder/Decoder Descriptors from USB Audio v2.0 are not supported.*

7 Known issues, limitations, and workarounds

Problem	Workaround
The USB Configurator supports import from the HID Descriptor tool, whose version 2.40 contains an error related to strings. Per spec HID1.11, string items have such values: • String Index 0111 10 nn • String Minimum 1000 10 nn • String Maximum 1001 10 nn However, the HID Descriptor tool generates: • String Index 0110 10 nn • String Minimum 0111 10 nn • String Maximum 1000 10 nn	Before importing, fix these items manually in the file generated with the HID descriptor tool.

8 Version changes

This section lists and describes the changes for each version of this tool.

Version	Change descriptions
1.0	New tool.
1.10	Updated the icons to be standard. Added Notice List.
2.0	Changed the user configuration storage location from the header file to the *.cyusbdev file.
	Added New, Save As, Reset View commands. Changed the Load command to Open.
	Updated the icons.
	Removed the CUSTOM item from HID Report.
	Removed the functionality to launch the USB Configurator from the Device Configurator.
2.10	Added the Undo/Redo feature.
2.20	Updated versioning to support patches.
	Added Copy feature to the Notice List.
	Added the calculation of the Endpoint Address for a new Endpoint Number.
2.30	Removed the command-line generate options: -g and –generate.
2.40	Added Device Descriptor Tree as the root element.
	Added: Device Qualifier Descriptor with Other_Speed_Configuration Descriptor, Billboard Capability Descriptor, and Billboard Alternate Mode Capability Descriptor.
	Added an Array Editor dialog for array fields.
	Updated the GUI by moving to Qt-5.15.2
	Removed: the migration of configuration to the current XML format – configuration saved in the comments in generated HEADER files (the old method).
2.50	Added SuperSpeed Device Capability Descriptor, SuperSpeedPlus Device Capability Descriptor, PTM Capability Descriptor, SuperSpeed Endpoint Companion Descriptor, SuperSpeedPlus Isochronous Endpoint Companion Descriptor.
	Updated Array Editor.
	Changed the device library file from xml to props.json.
2.51	Updated the bMaxPower parameter generation logic. Now, the generated source value is the same as in the GUI. This allows the user to freely set the required value, for example 2 mA units when the device is operating in High-speed mode and 8 mA units when operating at Gen X speed.
	Added the migration mechanism to prevent changes in the generated source for previous versions of configuration. In previous version, a value from the GUI was divided by 2 to match the USB 2.0 mod
	Changed the bMaxPower default value to 25.
2.60	Fixed minor bugs.
2.70	Bug fixes. GUI format changes. Minor back-end changes.

8 Version changes

Version	Change descriptions
2.80	Minor back-end changes.
2.90	Minor back-end changes.
3.0	Minor back-end changes.
3.10	Minor back-end changes.
3.20	Minor back-end changes.

Revision history

Revision	Date	Description
**	2018-11-20	New document.
*A	2019-02-26	Updated to version 1.10.
*B	2019-10-16	Updated to version 2.10.
*C	2020-03-27	Updated to version 2.10.
*D	2020-09-01	Updated to version 2.20.
*E	2021-03-11	Updated to version 2.30.
*F	2021-09-22	Updated to version 2.40.
*G	2022-09-28	Updated to version 2.50.
*H	2023-05-18	Updated to version 2.51.
*I	2023-05-19	Added a known issue about bmAttributes of Endpoint Descriptor bits 5..2.
*J	2024-01-21	Updated to version 2.60.
*K	2024-01-24	Minor changes.
*L	2024-09-27	Updated to version 2.70.
*M	2024-12-05	Updated to version 2.80.
*N	2025-03-21	Updated to version 2.90.
*O	2025-08-13	Updated to version 3.0.0.
*P	2025-12-11	Updated to version 3.10.0
*Q	2026-03-21	Updated to version 3.20.0

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2026-03-21

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-icv1712685607781

Important notice

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.