

# ModusToolbox™ DFU Host tool user guide

ModusToolbox™ tools package version 3.1.0

DFU Host version 2.0.0

## About this document

### Scope and purpose

The Device Firmware Update (DFU) Host tool is a stand-alone program included with the ModusToolbox™ software. This tool is used to communicate with PSoC™ 6, PSoC™ 4 and XMC™7000 MCUs that have already been programmed with an application that includes the DFU capability.

### Intended audience

This document helps application developers understand how to use the DFU Host tool as part of creating a ModusToolbox™ application.

### Document conventions

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

### Reference documents

Refer to the following documents for more information as needed:

- [Eclipse IDE for ModusToolbox™ user guide](#)
- [KitProg3 user guide](#)

---

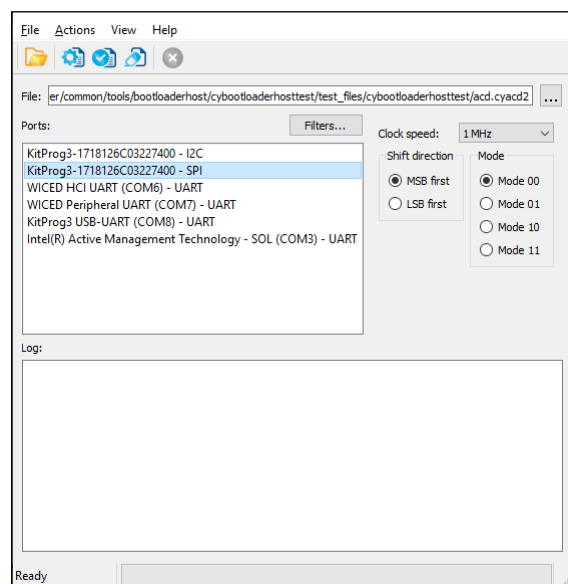
**Table of contents****Table of contents**

<b>1</b>	<b>Overview .....</b>	<b>3</b>
<b>2</b>	<b>Supported devices and file formats .....</b>	<b>4</b>
<b>3</b>	<b>Example code .....</b>	<b>5</b>
<b>4</b>	<b>Quick start.....</b>	<b>6</b>
<b>5</b>	<b>Launch the DFU Host tool .....</b>	<b>7</b>
5.1	make command .....	7
5.2	Eclipse IDE .....	7
5.3	Executable (GUI).....	7
5.4	Executable (CLI).....	7
<b>6</b>	<b>GUI description .....</b>	<b>9</b>
6.1	Menus.....	9
6.2	Main window .....	9
6.3	Port configuration .....	10
6.4	Log .....	11
6.5	Errors .....	11
<b>7</b>	<b>CLI description.....</b>	<b>12</b>
7.1	Command-line flags .....	13
7.2	CLI example .....	13
<b>8</b>	<b>Input files .....</b>	<b>15</b>
8.1	.mtbdfu file .....	16
<b>9</b>	<b>Customizing the DFU Host Tool .....</b>	<b>19</b>
<b>10</b>	<b>Troubleshooting .....</b>	<b>20</b>
<b>11</b>	<b>Version changes .....</b>	<b>21</b>

## Overview

# 1 Overview

The DFU Host tool is provided as a [graphical user interface \(GUI\)](#) and a [command-line interface \(CLI\)](#).



This tool allows you to:

- Program new application data onto the supported MCU device using supported input files. See [Supported devices and file formats](#).
- Verify the program data that is already contained on the device.
- Erase the application from the device.
- Select the \*.mtbdfu file output from a ModusToolbox™ application in order to program, write, read, erase and communicate with the device.
- Abort the current operation.

**Note:** *This operation leaves the device in whatever state it is in when the abort message is acted upon.*

The DFU Host tool supports communicating via I<sup>2</sup>C, SPI, UART, and USB-CDC. They are displayed to configure their settings and view the programming status. For I<sup>2</sup>C and SPI, supported MCU should include the KitProg3 firmware module or other bridge interface, which implements USB-UART, USB-I<sup>2</sup>C, and USB-SPI bridges. For UART, communication can be done directly from the PC simply by connecting an appropriate cable.

**Note:** *The process of initializing the device, two CPUs therein, and executing code in the SROM and supervisory flash, is referred to as “bootloading.” The process of installing and updating applications in the field is referred to as “device firmware update”. This process uses standard communication channels (UART, I<sup>2</sup>C, USB, etc.) to download new applications from a host.*

## Supported devices and file formats

## 2 Supported devices and file formats

All MCUs can be used with the all files formats.

Device	File/format			Link
	*.mtbdfu	*.hex file	*.cyacd2	
MCU				
PSoC™ 6	✓	✓	✓	<a href="https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/psoc-6-32-bit-arm-cortex-m4-mcu/">https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/psoc-6-32-bit-arm-cortex-m4-mcu/</a>
PSoC™ 4	✓	✓	✓	<a href="https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/psoc-4-32-bit-arm-cortex-m0-mcu/">https://www.infineon.com/cms/en/product/microcontroller/32-bit-psoc-arm-cortex-microcontroller/psoc-4-32-bit-arm-cortex-m0-mcu/</a>
XMC7000	✓	✓	✗	<a href="https://www.infineon.com/cms/en/product/microcontroller/32-bit-industrial-microcontroller-based-on-arm-cortex-m/32-bit-xmc7000-industrial-microcontroller-arm-cortex-m7/">https://www.infineon.com/cms/en/product/microcontroller/32-bit-industrial-microcontroller-based-on-arm-cortex-m/32-bit-xmc7000-industrial-microcontroller-arm-cortex-m7/</a>

---

## Example code

### 3 Example code

We provide source code for the DFU Host tool in the installation directory, as follows:

- **Windows/Linux:** `~/ModusToolbox/tools_<version>/dfuh-tool/sample_code/`
- **Linux:** `~/ModusToolbox/tools_<version>/dfuh-tool/share/dfuh-tool/sample_code/`
- **macOS:** `/Applications/ModusToolbox/tools_<version>/dfuh-tool/dfuh-tool.app/Contents/sample_code/`

Use this source code as a reference to build your own tool. Refer to the README.md file for instructions on how to install the required dependencies and build the DFU Host Tool.

*Note: This example requires CMake 3.12 or later.*

---

## Quick start

### 4 Quick start

This section contains a simple workflow for how to use the DFU Host tool.

1. Ensure that you have a supported MCU that has been programmed with an application that includes the DFU capability.
2. [Launch the DFU Host tool GUI](#).
3. Click **File > Open** to browse to the location of your application file (\*.cyacd2/\*.mtbdfu/\*.hex).
4. Connect a hardware port that supports the communication selected in your bootloader project.
5. Wire the hardware port pins to the corresponding pins on the target device.
6. Update the [port configuration](#). These values are set from the communication component used by the bootloader component.
7. Click **Program/Execute** to load the new application from \*.cyacd2/\*.hex file and to execute and send the commands from the .mtbdfu file.

## Launch the DFU Host tool

### 5 Launch the DFU Host tool

There are several ways to launch the DFU Host tool, and those ways depend on how you use the various tools in ModusToolbox™ software.

#### 5.1 make command

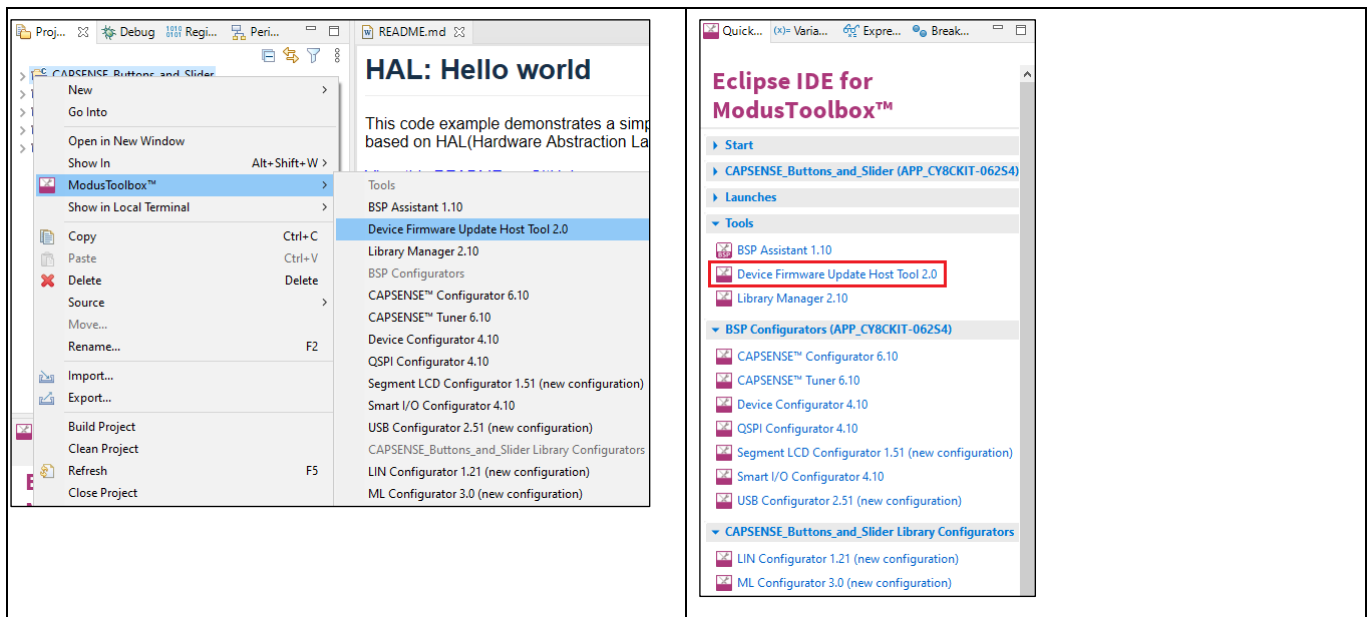
As described in the [ModusToolbox™ user guide](#) "Build System" chapter, you can run numerous make commands in the application directory, such as launching the DFU Host tool. After you have created a ModusToolbox™ application, navigate to the application directory and type the following command in the appropriate bash terminal window:

```
make open CY_OPEN_TYPE=dfuh-tool
```

This command opens the DFU Host tool GUI for the specific application in which you are working.

#### 5.2 Eclipse IDE

If the selected Eclipse IDE application is suitable to use the DFU Host tool, right-click on the appropriate project and select **ModusToolbox™ > Device Firmware Update Tool**. You can also click the DFU Host tool link in the IDE Quick Panel.



#### 5.3 Executable (GUI)

You can launch the DFU Host tool as a stand-alone tool without the Eclipse IDE. By default, it is installed here:

```
<install_dir>/ModusToolbox_<version>/tools_<version>/dfuh-tool
```

On Windows, you can launch the tool from the **Start** menu. For other operating systems, the installation directory will vary, based on how the software was installed.

#### 5.4 Executable (CLI)

Refer to [CLI Description](#) to run the *dfuh-cli* tool. You can also run the DFU Host tool GUI executable using the following command line arguments:

---

## Launch the DFU Host tool

-?, -h, --help	Displays help on command line options.
--help-all	Displays help including Qt specific options.
-v, --version	Displays version information.
--debug <filename>	Appends logging information to the specified file.



## GUI description

# 6 GUI description

## 6.1 Menus

### 6.1.1 File

- **Open...** – Opens an existing input file of the supported formats. If a file is already open, it will be closed.
- **Save Log As...** – Opens the dialog to save the log file in a specified location.
- **Clear Log** – Clears the current log.
- **Exit** – Closes the DFU Host tool.

### 6.1.2 Actions

- **Program** – Programs the application to the device.
- **Execute** – Executes the .mtbdfu file on the host, establishes a DFU session with the target and processes the DFU commands
- **Verify** – Verifies the programming of the device.
- **Erase All** – Erases the program from the device.
- **Abort** – Aborts the action.

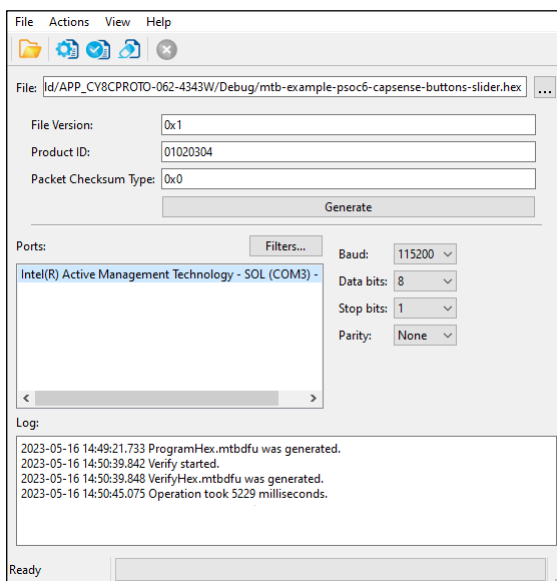
### 6.1.3 View

- **Toolbar** – Shows/hides the toolbar.

### 6.1.4 Help

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

## 6.2 Main window



## GUI description

### 6.2.1 File selection

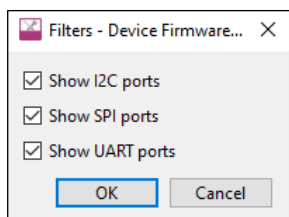
Use this section to select the input file, which contains the DFU commands/firmware image.

### 6.2.2 Ports

This is a list of all the ports attached to the computer that can be used to communicate with the bootloader. Based on which item is selected, different options are available for the Port Configuration.

### 6.2.3 Filters

The **Filters** button allows for restricting which ports are displayed in the ports list.



## 6.3 Port configuration

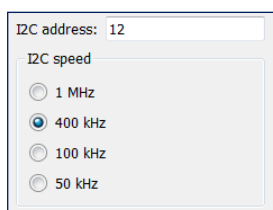
This section allows for configuring the interface-specific options for communicating with the DFU system. This is necessary to ensure both the DFU and host computer are configured the same.

Refer to the appropriate probe documentation for a list of supported modes.

*Note: Not all SPI and UART communication properties combinations are supported.*

### 6.3.1 I<sup>2</sup>C

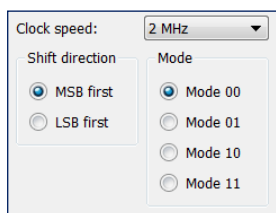
For I<sup>2</sup>C communication, there are two required parameters:



- The address of the I<sup>2</sup>C-based target DFU system with which the host is communicating. The range for valid addresses is from 8 - 120.
- The I<sup>2</sup>C SCK signal frequency.

### 6.3.2 SPI

For SPI communication, there are three required parameters:

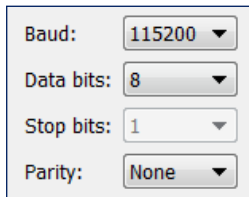


## GUI description

- The SPI SCLK signal frequency.
- The bit ordering of transferred data.
- The SPI operating mode.

### 6.3.3 UART and USB\_CDC

For UART/USB\_CDC communication, there are four required parameters:



A screenshot of a configuration window for UART/USB\_CDC communication. It contains four dropdown menus: 'Baud:' set to '115200', 'Data bits:' set to '8', 'Stop bits:' set to '1', and 'Parity:' set to 'None'.

- The baud (bit) rate at which data is transferred.
- The number of data bits per byte.
- The number of stop bits indicating the termination of a byte.
- The parity bit that is added to a byte.

## 6.4 Log

The log displays the history of what happened while the host was open:

- when operations started/completed
- information about user-initiated operations
- error messages during an operation if any.

## 6.5 Errors

Any errors for various fields display as a red X in the field containing the error, and it contains a tooltip when you hover the mouse cursor on it.

## CLI description

## 7 CLI description

In addition to the *dfuh-tool* GUI executable, there is also the *dfuh-cli* executable. The CLI allows programming, verifying, and erasing devices from a command-line prompt or from within batch files or shell scripts. The exit code for the *dfuh-cli* executable is zero if the operation is successful, or non-zero if the operation encounters an error.

To use the *dfuh-cli* executable, provide one of the following flags:

<code>--program-device &lt;cyacd2_file&gt;</code>	Programs the device with the specified file and exits.
<code>--verify-device &lt;cyacd2_file&gt;</code>	Verifies the programming of the device with the specified file and exits.
<code>--erase-device &lt;cyacd2_file&gt;</code>	Erases the specified program from the device and exits.
<code>--custom-command &lt;mtbdfu_file&gt;</code>	Sends an .mtbdfu file (JSON format) as the input to the DFU Host tool.
<code>--generate-mtbdfu &lt;mtbdfu_file&gt;</code>	Generates an .mtbdfu file from the provided input arguments.

If there is more than one device connected to the host, use the following flag to specify which device to use:

<code>--hwid &lt;string&gt;</code>	Specifies the ID of the hardware to program/verify/erase. If this option is skipped, the first appropriate device found will be used.
------------------------------------	---

The **.mtbdfu generation** flags:

<code>--product-id &lt;hex&gt;</code>	Sets the product ID as a parameter for the generated .mtbdfu file.
<code>--file-version &lt;hex&gt;</code>	Sets the file version as a parameter for the generated .mtbdfu file.
<code>--checksum-type &lt;hex&gt;</code>	Sets packet checksum type as a parameter for generated .mtbdfu file.
<code>--application-id &lt;hex&gt;</code>	(Optional) Sets the application ID as a parameter for the generated .mtbdfu file. Required only for legacy devices or devices with the DFU-MW-based bootloader.
<code>--application-start &lt;hex&gt;</code>	(Optional) Sets the application start address as a parameter for the generated .mtbdfu file. Required only for legacy devices or devices with the DFU-MW-based bootloader.
<code>--application-length &lt;hex&gt;</code>	(Optional) Sets the application size as a parameter for the generated .mtbdfu file. Required only for legacy devices or devices with the DFU MW based bootloader.

**Note:** The above flags are also used to program the .hex file because it generates an .mtbdfu file and uses the same for programming the .hex file.

**Note:** The options “--application-start” and “--application-length” can be used to set the address range for filtering the needed part of the .hex file. All extended linear address blocks before the “--application-start” address will be skipped as well as all the data at addresses after “--application-length” address.

In addition, you must provide the appropriate configuration values for one of the following protocols:

The **I<sup>2</sup>C** flags:

<code>--i2c-address &lt;int&gt;</code>	Sets the address for the I <sup>2</sup> C protocol. Valid values are between 8 (0x08), and 120 (0x78).
<code>--i2c-speed &lt;int&gt;</code>	Sets the speed for the I <sup>2</sup> C protocol in kHz. Common values are 50, 100, 400, and 1000.

## CLI description

The **SPI** flags:

<code>--spi-clockspped &lt;float&gt;</code>	Sets the clock speed for the SPI protocol in MHz.
<code>--spi-mode &lt;int&gt;</code>	Sets the mode for the SPI protocol in binary. Valid values are 00, 01, 10, and 11.
<code>--spi-lsb-first</code>	Specifies that the least-significant bit should be sent first for the SPI protocol. Otherwise, the most-significant bit will be sent first.

The **UART** flags:

<code>--uart-baudrate &lt;int&gt;</code>	Sets the baud rate for the UART protocol.
<code>--uart-databits &lt;int&gt;</code>	Sets the number of data bits for the UART protocol.
<code>--uart-paritytype &lt;string&gt;</code>	Sets the parity type for the UART protocol. Valid strings are "None", "Odd", and "Even".
<code>--uart-stopbits &lt;float&gt;</code>	Sets the stop bits for the UART protocol. Valid values are 1, 1.5, and 2.

*Note:* The DFU Host tool does not require any additional parameters to select the required interface besides the above specified parameters.

## 7.1 Command-line flags

The following flags change the overall functioning of the tool:

<code>-, -h, --help</code>	Displays information about all valid command-line arguments and exits.
<code>-v, --version</code>	Displays the version information and exits.
<code>--debug</code>	Outputs debugging information to the terminal running the CLI tool during programming, verifying or erasing.
<code>--display-hw</code>	Outputs all compatible hardware attached to the computer and exits.

## 7.2 CLI example

The following shows simple examples for using the *dfuh-cli* executable

### 7.2.1 Programming an image via \*.cyacd2 file

1. To program an image using the **.cyacd2** file under **I<sup>2</sup>C**:  

```
dfuh-cli.exe --program-device test_app.cyacd2 --i2c-address 8 --i2c-speed 100
```

### 7.2.2 Generating and programming an image via \*.mtbdfu file and \*.hex file

1. To generate an **.mtbdfu** file (without metadata):  

```
dfuh-cli.exe --generate-mtbdfu test_gen.mtbdfu --mtbdfu-data-file blinky_cm4_crc.hex --file-version 0x1 --product-id 01020304 --checksum-type 0x0
```
2. To generate an **.mtbdfu** file for **legacy** devices (with metadata):  

```
dfuh-cli.exe --generate-mtbdfu test_gen.mtbdfu --mtbdfu-data-file blinky_cm4_crc.hex --file-version 0x1 --product-id 01020304 --checksum-type 0x0 --application-id 0x1 --application-start 1005 --application-length fffc
```
3. To program the **.hex** file through the **.mtbdfu** file under **UART** (without metadata):  

```
dfuh-cli --program-device test_app.hex -huid COM5 --uart-baudrate 115200 --uart-databits 8 --uart-paritytype None --uart-stopbits 1 --file-version 0x1 --product-id 01020304 --checksum-type 0x0
```
4. To execute or send commands through the **.mtbdfu** file under **SPI**:

---

## CLI description

```
dfuh-cli --custom-command test.mtbdfu --hwid MiniProg4-0C0806F801071400 --  
spi-clocksperd 1.0 --spi-mode 0
```

## Input files

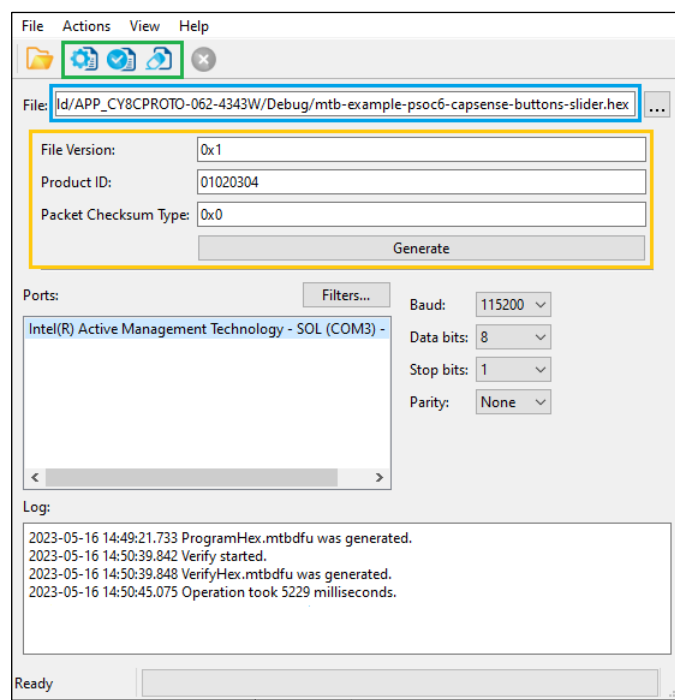
### 8 Input files

The DFU Host tool requires the .mtbdfu file (JSON format) as an input to retain the backward compatibility for PSoC™ 4 and PSoC™ 6 legacy devices. It also supports the .cyacd2 file format as an input for the basic program, verify, and erase operations.

The details for the supported input file:

1. \*.mtbdfu file: The JSON input file for the DFU Host tool. It can be used for communication with the device using different commands available in the file. It should include a set of commands with respective data to be sent to the device through the DFU Host tool, using the DFU Command-Response protocol.
2. \*.cyacd2 file: This can only be used for the firmware update use case. The file will consist of the required header information for the firmware image along with the firmware binary and address where data will be loaded. This format is only supported for PSoC™ 6 and PSoC™ 4 devices.

However, for the basic program, verify, and erase operations, the DFU Host tool can generate an .mtbdfu file using the .hex file provided as an input with the necessary header parameters required for the .mtbdfu file. The below figure shows the dynamic window of the DFU Host tool, which displays when the .hex file is selected as the input.



1. The green box on the toolbar indicates the actions available for using the .hex file as an input: program, verify, and erase the device. Modify the required parameter shown in the image and select the required option.
2. The blue box displays the .hex file path selected as an input to the host tool. Only limited operations/actions are possible with selecting the .hex file as the input. For complex operation, use the .mtbdfu file with the required set of commands.
3. The orange box indicates the “APPinfo” section of the .mtbdfu file to be generated. The window displays the default value for the same, the values must be modified per device and use case. For more details refer [.mtbdfu](#) section above. The **Generate** button generates the reference .mtbdfu file for the program use case, the file will be saved at the installation directory of the DFU Host Tool.

## Input files

### Sample JSON generated for the basic program use case

```
{
  {
    "APPInfo": {
      "File Version": "0x1",
      "Product Id": "01020304",
      "Packet Checksum Type": "0x0"
    },
    "commands": [
      {
        "dataFile": "blinky_cm4_crc.hex",
        "commandSet": [
          {
            "cmdId": "0x37",
            "dataLength": "0x10",
            "repeat": "0x20"
          },
          {
            "cmdId": "0x49",
            "dataLength": "0x08"
          }
        ],
        "repeat": "EoF",
        "flashrowlength": "0x200"
      }
    ]
  }
}
```

The above JSON sample displays the template .mtbdfu file generated from the given .hex file as the input. The DFU Host Tool will use \*.mtbdfu file for programming the firmware .hex file to the device.

The default flow of the DFU-MW-based firmware download follows the set of repetitive send data command(0x37) with 32Bytes of data and a program row command(0x49) with the leftover row data, row checksum, and the row address to write.

The JSON file uses the **APPInfo** object required for the DFU communication setup. It contains potential information, the image version, product ID of the device the same as loaded with the loader application along with the checksum type for DFU command packet.

The sample contains the command object, which contains the list of command sent to the device. This object can be used to send a single as well as multiple DFU commands in the sequence required. The object contains another sub-object **commandSet**, which can group multiple commands to provide shared data across the group and repeat the set of commands.

This set is repeated till the end of the image file, the command requires the flash row length to divide the packets into multiple sets. For more details, refer to [AN 213924](#).

## 8.1 .mtbdfu file

The file contains the following sections:

**AppInfo** – Provides information about the device and the application

- File version – Indicates the application version, can be used to track updates.
- Product ID – Unique ID of the device communicating with the host.
- Checksum type – Used for the .hex file, for DFU packet checksum.

**Command** – Consists of the details of the command to be sent to the device along with the data.



## Input files

### JSON fields available in the Command section:

Field	Description
cmdId	Command field. Specifies the DFU command number; the valid range is 0x00 - 0xFF. [0x00 - 0x49]: A standard DFU command range [0x50-0xFF]: Reserved for Future Use
commandSet	Used to denote a set of commands, required for repeating a set of commands.
dataBytes	(Optional) The actual payload to be used in a command.
dataFile	(Optional) Specifies the name of the input file from where the payload data will be read.
dataLength	(Optional) The length of the payload data in bytes.
flashRowLength	Specifies the flash row length on the device. Valid only if dataFile is defined.
outFile	(Optional) Specifies the file name to store intermediate packets generated by the tool.
repeat	Repeats the command for a specified number of times. Valid only when dataFile is defined. For example: repeat: "10" means that the command will be repeated 10 times. Repeat: "EoF" means that the command will be repeated until the end of the data file.
startOffset	Specifies the file offset from where data bytes will be read. Valid only if dataFile is defined.
sessions	(Optional) Used to define multiple DFU sessions in the .mtbdfu file.
outFile	(Optional) Specifies the filename to store intermediate packets generated by the tool.
outCli	(Optional) Output response data from the device to the command line.
msg	(Optional) Used to add comments to the .mtbdfu file. These will not be sent to the device, they will be available in the output message from the DFU Host Tool.

**Note:** The APPIInfo section is mandatory for the JSON file; it is required to start the DFU communication.

**Note:** Optional commands must be defined per use case.

## Input files

### Sample to use JSON fields in .mtbdfu file

```

{
  "APPInfo": {
    "File Version": "0x1",
    "Product Id": "01020304",
    "Packet Checksum Type": "0x0"
  },
  "sessions": [
    {
      "commands": [
        {
          "commandSet": [
            {
              "cmdId": "0x37",
              "dataLength": "0x02",
              "dataBytes": ["0xAB", "0x54"],
            },
            {
              "cmdId": "0x49",
              "dataLength": "0x12",
              "startOffset": "0x10",
              "datafile": "data.hex"
            }
          ],
          "repeat": "0x20"
        }
      ]
    },
    {
      "commands": [
        {
          "msg": "This is sample comment"
          "cmdId": "0x4C",
          "dataLength": "0x09",
          "dataBytes": ["0x00", "0x10", "0x50", "0x00", "0x00", "0x00", "0x00", "0x00", "0xfc", "0xff"],
          "outFile": "OutFile.txt"
        }
      ]
    }
  ]
}

```

DFU Session 1

DFU Session 2

**Note:** The above sample JSON is used only to demonstrate how to use the various arbitrary fields in the .mtbdfu file. This .mtbdfu file is only a reference template and does not correspond to any specific use case.

## Customizing the DFU Host Tool

### 9 Customizing the DFU Host Tool

The DFU Host tool uses a set of timeouts for read operation before disconnecting and sending the timeout error. These internal configurations are suited for the development and field upgrade operations for programming the firmware or communicating with the device.

However, these configurations might require modification while debugging the DFU applications with transports using the multiple breakpoint in the target side. Due to the breakpoint, the device will pause execution and might not be able to send a response desired by the DFU Host Tool. Recommended – modify the timeout configuration and rebuild the DFU Host Tool while debugging the application using the DFU transport.

Modify the source code available at Windows:

`~/ModusToolbox/tools_<version>/dfuh-tool/sample_code/`

as specified in the below table for respective transport.

#### Source code available at Windows for respective transport

Interface	File	Variable	Description
I <sup>2</sup> C	cychanneli2c.cpp	MAX_READS	The MAX_READS denotes the count for which the Host tool should poll/retry to read the bus for correct data. Each consecutive read have 10ms delay, as hardcoded in the same API. Modifying either both or one can be used to modify the timeout for I <sup>2</sup> C channel.
SPI	cychannelspi.cpp	MAX_READS	Similar to the I <sup>2</sup> C timeout, SPI can also be configured through the same variable “MAX_READS” as mentioned above.
UART	cychanneluart.h	READ_WRITE_TIMEOUT_MS	Modify the hard-coded value of variable “READ_WRITE_TIMEOUT_MS”. The default read/write timeout for UART is configured as 5 seconds.

**Note:** *In the above table, all the referred .cpp files will be available at ~/sample\_code/src/backend/ and the referred .h file will be available at ~/sample\_code/include/backend/*

After making the above modification in the source code of the DFU Host tool, build the DFU Host tool again. For details of building the DFU Host tool, refer to section [Example code](#).

## Troubleshooting

# 10 Troubleshooting

Problem	Workaround
On common Linux distributions, the serial UART ports (usually /dev/ttySx or /dev/ttyUSBx devices) belong to the root user and to the dialout group. Standard users are not allowed to access these devices.	<p>An easy way to allow the current user access to the Linux machine's serial ports is by adding the user to the dialout group. This can be done using the following command:</p> <pre>\$sudo usermod -a -G dialout \$USER</pre> <p><i>Note:</i> For this command to take effect, you must log out and then log back in.</p>
On Linux, attempts to set up or start DFU Host tool communication causes the tool to close without any messages.	<p>To enable DFU Host tool communication under Linux, install the udev rules for KitProg3:</p> <p>Disconnect the KitProg device.</p> <p>Execute in the terminal (root access required):</p> <pre>sh \$CYS SDK/tools/fw-loader/udev_rules/install_rules.sh</pre> <p>Reconnect the KitProg device.</p>
On common Linux distributions, the DFU Host tool forbids communication protocol selection after re-plugging KitProg during communication.	Refer to the “Installation Procedure on Ubuntu Linux (x64)” section in the CYPRESS™ Programmer 2.1 CLI User Guide.
KitProg3 UART is accessible but not able to read data on Linux kernel 4.15 and above or Mac OS X 10.13 and above.	<p>Use a third-party UART to USB bridge.</p> <p>Update the KitProg3 firmware to version 1.11.243 or above.</p>
After updating firmware and middleware for your application, SPI transfer speed is not as fast as expected.	You may be able to improve performance by modifying the <code>src/backend/cychannelspi.cpp</code> file. Remove the calls to <code>QThread::msleep(1)</code> when building your bootloader host tool.
Sending command 0x47 to a PSoC™ 6 device via SPI is not supported.	Command 0x37 can be used for transferring a large amount of data to a PSoC™ 6 device.

---

**Version changes**

## 11 Version changes

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

Version	Change Descriptions
1.0	New tool.
1.1	Added Notice List. Added command-line interface (CLI) and handling of invalid command line arguments. Added logging for firmware update process.
1.2	Removed Notice List.
1.30	Updated versioning to support patches.
1.40	Fixed minor defects.
1.50	Added PSoC™ 4 device support.
1.60	Updated versioning to support internal libraries update. Changed the device library file from xml to <i>props.json</i> .
2.0	Added the .mtbdfu file format support. Added the .hex file format support. Added the XMC7000 device support.

---

**Revision history****Revision history**

Revision	Date	Description
**	2018-11-09	New document.
*A	2019-10-16	Updated to version 1.1.
*B	2020-03-27	Updated to version 1.2.
*C	2020-09-01	Updated to version 1.30.
*D	2020-10-16	Added information about code example.
*E	2021-03-11	Updated to version 1.40.
*F	2022-05-10	Updated to version 1.50.
*G	2022-09-30	Updated to version 1.60.
*H	2023-05-19	Updated to version 2.0.
*I	2023-05-24	Added information to Troubleshooting section: "Sending command 0x47 to a PSoC™ 6 device via SPI is not supported".

#### Trademarks

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

**Edition 2023-05-24**

**Published by**

**Infineon Technologies AG**  
**81726 Munich, Germany**

**© 2023 Infineon Technologies AG.**  
**All Rights Reserved.**

**Do you have a question about this document?**

**Email:** [erratum@infineon.com](mailto:erratum@infineon.com)

**Document reference**

**002-25258 Rev. \*I**

#### Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffheitsgarantie")

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, 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.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

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.