Document Number: MBOOTMFGTOOLUG
Rev 2, 09/2018

Kinetis MCU Manufacturing Tool User's Guide

by: NXP Semiconductors

Contents

1 Introduction

The Kinetis family and i.MX RT series of MCUs are pre-manufactured with the MCU bootloader in either the ROM or flash memory of the device, and can boot into the MCU bootloader application. A PC host or master can connect to the MCU bootloader device via the USB-HID or UART interface, and uses the bootloader's command protocol interface to program the image onto the MCU device.

The latest version of the Manufacturing tool (MfgTool) application supports the MCU bootloader and can be used in factory production environment in the same way as with other MfgTool-supported devices. The MfgTool application can detect the presence of the MCU bootloader connected to the PC and invokes the *blhost.exe* file to program the image onto the target device. The MCU bootloader and the blhost utility are documented in a greater detail in `<sdk_package>/middleware/mcu-boot/doc`. This document provides a user's guide on how to use the MfgTool for the MCU device manufacturing.

2 Directory structure

The MfgTool for MCU devices is bundled together with the MCU bootloader in `<sdk_package>/middleware/mcu-boot`. The following figure provides the mfgtools-rel directory structure on the host PC after the package installation.

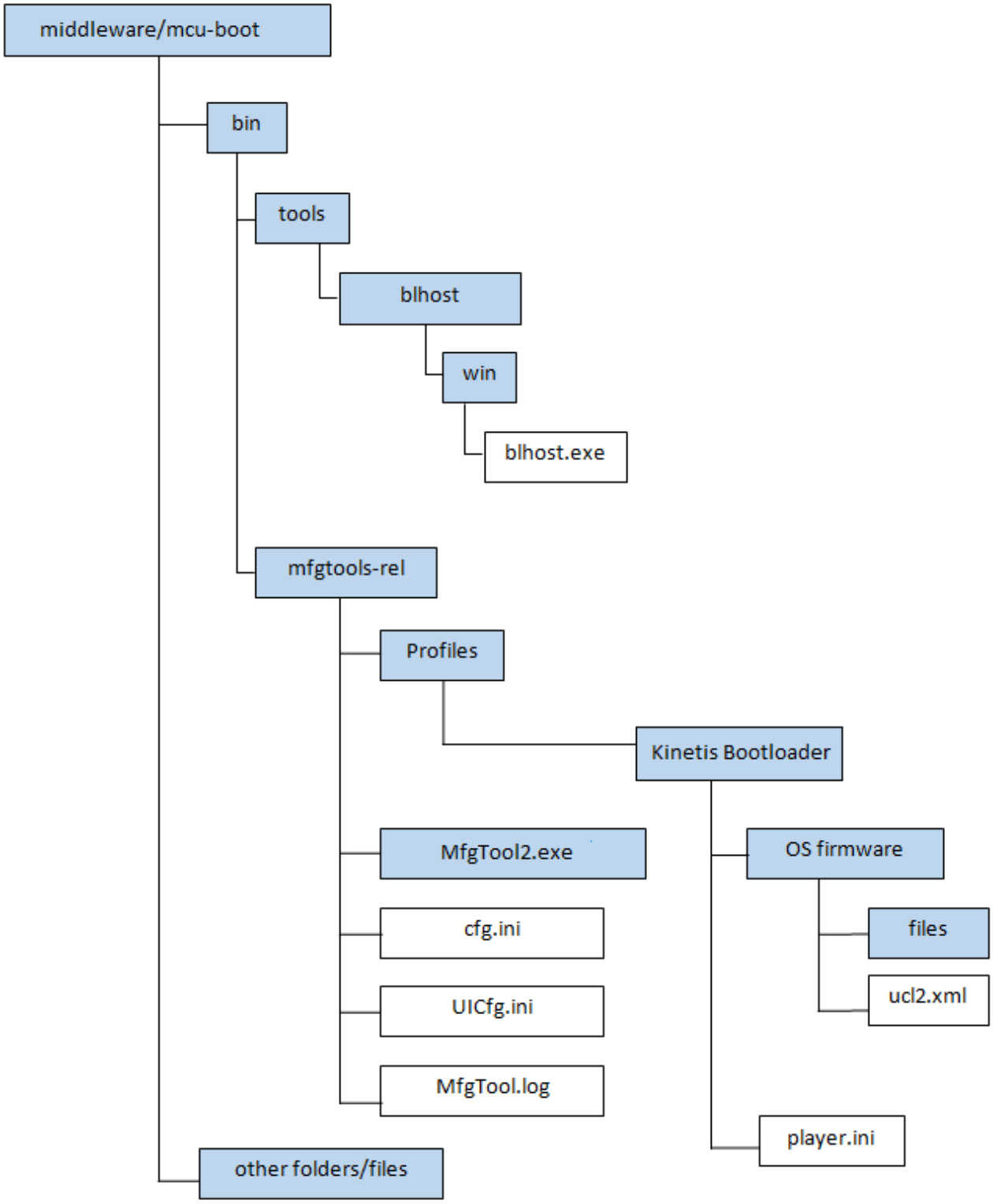

Figure 1. MCU bootloader folder hierarchy

In the package, the *mfgtools-rel* folder appears in the *middleware/mcu-boot/bin/tools* folder along with the *blhost* folder.

The blhost.exe file appears in the *blhost/win* folder and the MfgTool executable *MfgTool2.exe* appears in the *mfgtools-rel* folder.

The *Profiles* folder contains the profile for the MCU bootloader device that includes the *OS Firmware* folder and the *player.ini* file.

The *files* folder in the *OS Firmware* folder may include the application image along with the custom files to download onto the device.

The *ucl2.xml* file in the *OS Firmware* folder is the main XML that the MfgTool processes. It contains the flow of the manufacturing process for the device, which includes the identification parameters for the device and the blhost commands sequenced to query the device and program the image. The example *ucl2.xml* file is delivered with the USB device identification parameters to identify the MCU bootloader device connected to the PC host and the set of blhost commands required for updating the image on the device. The *ucl2.xml* file can be customized to suit the custom setup or the manufacturing process flow. The folder contains several example xml files for your reference.

The *player.ini* file in the 'Kinetis Bootloader' profile folder contains configurable parameters for the manufacturing tool application.

The *cfg.ini* and *UICfg.ini* files provide the customizable parameters for the look and feel of the tool's GUI.

The *MfgTool.log* text file is a useful tool to debug the failures reported in the MfgTool UI. The MfgTool logs the entire command line string it used to invoke the blhost, and also collects the output response text that the blhost puts out on stdout into the *MfgTool.log* file. The log file is the first place to look into when troubleshooting. The troubleshooting guide section at the end of this document discusses more on the information the tool logs into the *MfgTool.log* text file.

3 Initialization files

This section focuses on the parameters controlling the appearance of the MfgTool User Interface and other behavior that can be controlled via the initialization files with the *.ini* extension. The initialization files can be edited using any text editor.

- The *UICfg.ini* file is used to configure the number of ports that indicates the number of devices that can be simultaneously attached to multiple USB ports of the host PC and programmed simultaneously. The MfgTool presents one panel for each port on its GUI.

The format of the *UICfg.ini* file is as follows:

```
[UICfg]
PortMgrDlg=1
```

For example, if only one device is supported at a time, the "PortMgrDlg=1" should be set. The UI display with one dialog or panel is shown in this figure:

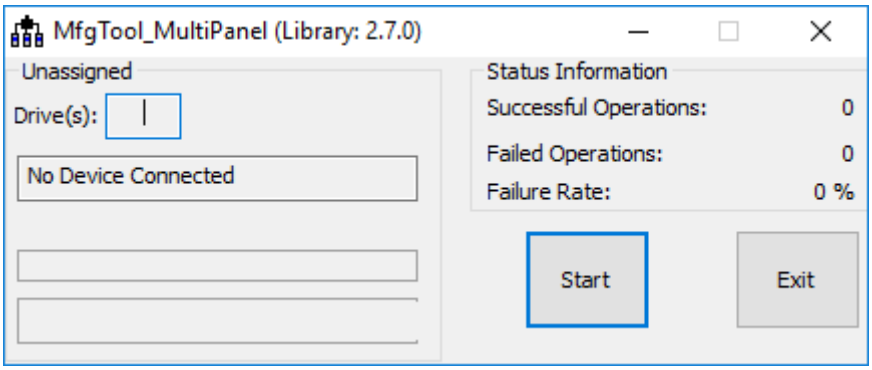

Figure 2. MfgTool UI with one panel in UICfg.ini

A maximum of up to four devices that can be simultaneously connected and updated using the MfgTool is supported. The PortMgrDlg entry in the *UICfg.ini* file can be set to any of one, two, three, or four panels, depending on the manufacturing needs. Here is an example screenshot with four ports:

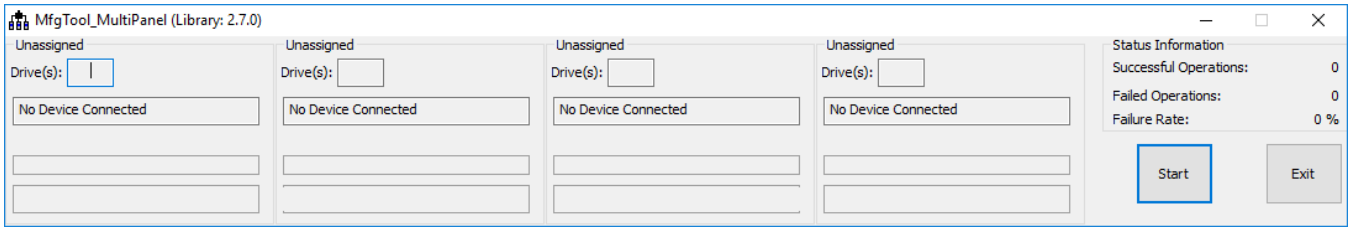

Figure 3. MfgTool UI with four panels in UICfg.ini

- The *cfg.ini* file is used to configure the target chip profile and target operation list.

The format of this file is as follows:

```
[profiles]
chip = Kinetis Bootloader

[platform]
board =

[LIST]
name = Kinetis-bootloader
```

The “profiles/chip” entry indicates the target profile name which should match the target folder name at "profiles/", and the “list/name” entry indicates the target operation list name which should match the LIST tag entry in the *ucl2.xml* file located at "profiles/Kinetis Bootloader/OS Firmware/". The “platform/board” entry is reserved and not used.

4 GUI elements

The GUI window of the MfgTool application consists of two types of panels. The panels on the left hand side (called device panels or panes) show the device(s) connected to the respective USB Hub and Port number. There may be one to four device panels depending on the value set to the PortMgrDlg entry in the *UICfg.ini* file. A single “Status Information” panel appears on the right side of the GUI window, and the two buttons at the bottom right of the UI dialog.

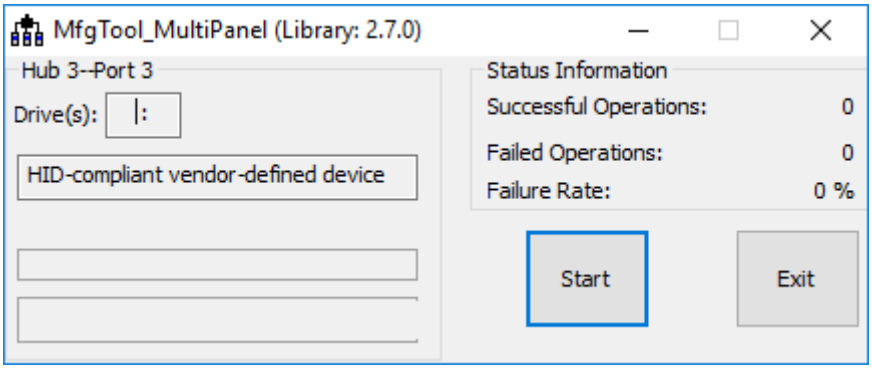

Figure 4. UI showing one device connected to the PC host in the idle state

4.1 Device panel

The device panel shows the PC hub and port numbers at the top of the pane to which the MCU bootloader device is connected.

The UI element “Drive(s)” is not applicable for MCU bootloader devices.

The next UI element shows the type of device connected in the idle state. The MCU bootloader device is connected in the USB-HID mode and the display shows ‘HID-compliant device’. When connected in the UART mode, the text shows the respective virtual COM port number along with the description of the device (as it appears in the Windows® OS device manager). While

manufacturing (non-idle state), the same element shows the command in the execution. When MfgTool is complete, the command sequences succesfully. The element can be made to display text such as ‘Done’, which is discussed later on in this document. When the manufacturing process is halted by the user using the "Stop" button, the text reverts back to the device description. The example screenshot below shows four MCUs connected to four USB ports in the HID and CDC modes in the idle state.

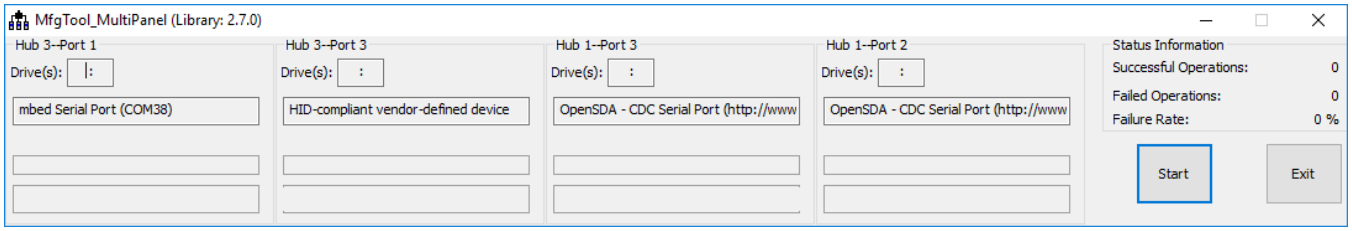

Figure 5. UI showing four MCUs connected to the PC host in the idle state

The last two UI elements of the device pane are the progress bars that show the progress while manufacturing. The first element shows the progress of the current blhost command being executed, and the second element shows the overall progress to complete the execution of all the commands listed in the ucl2.xml file. The progress bar appears in three different colors. Blue color indicates that the manufacturing is in process, green color indicates that the manufacturing completed successfully, and red color indicates that an error occurred. The error information are logged into the Mfgtool.log file for decoding purposes.

4.2 Status information panel

The Status Information panel have three data points, *Successful Operations* shows number of MCU bootloader devices successfully manufactured, *Failed Operations* shows number of devices failed and *Failure Rate* show the percentage failure.

The image below shows manufacturing in progress.

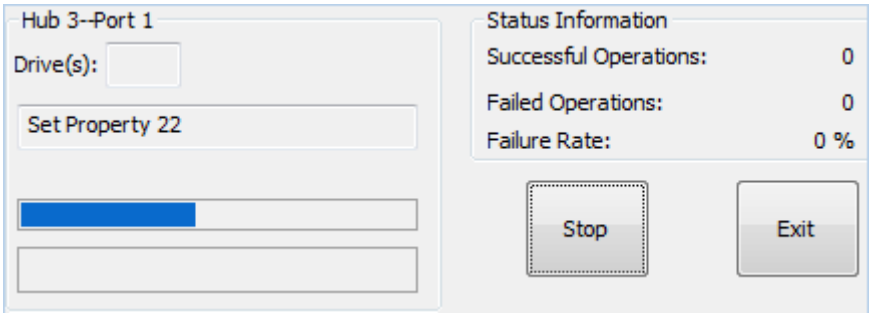

Figure 6. Manufacturing in progress

The image below shows that manufacturing completed successfully for one single device.

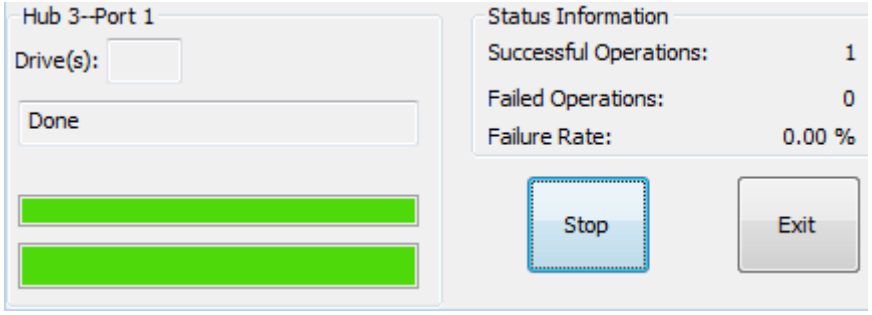

Figure 7. Manufacturing completed successfully

The image below shows an error has occurred during manufacturing of one single device.

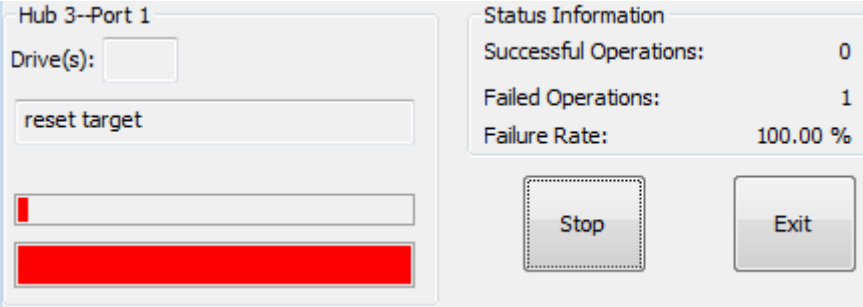

Figure 8. Error occurred during manufacturing

4.3 Start/Stop button

The "Start" button for the UI element appears in the idle state when the manufacturing process has either stopped or has not yet begun. To begin the manufacturing process, press the "Start" button. After pressing the "Start" button, the text on the button changes to “Stop” and acts like a stop button. Press the "Stop" button after the manufacturing process is complete or at any time to stop the process. Use the "Stop" button carefully to not stop while the command is being executed. Otherwise, an incomplete update may result in an unresponsive device.

4.3.1 Exit button

Press the "Exit" button to close the MfgTool application. The "Exit" button can only work in the idle state or when the "Start" button is active. The "Exit" button does not work when the device manufacturing is in progress. An error message dialog appears if the "Exit" button is pressed while manufacturing. Press the "Stop" button first to stop the manufacturing process, which activates the "Start" button. Then use the "Exit" button to close the window and shut down the MfgTool application.

4.4 Firmware update process

The chapter and subsections below describe the process of device manufacturing using the MfgTool application for the MCU bootloader devices.

4.4.1 MCU bootloader device identification

The MCU bootloader performs the active peripheral detection at device startup to connect to the host to carry out the firmware download operations. If the bootloader device is connected to the UART or the USB, the PC host operating system enumerates the device, either in the USB CDC mode (via the SDA port, subject to hardware support), or the USB-HID mode (not all MCUs have the USB support). The MfgTool can identify the device’s presence by comparing the active USB devices’ vendor and product identifiers (VID and PID) with the supported identifiers mentioned in the <CFG> tag of the *ucl2.xml* file. The MfgTool can connect and update up to four devices simultaneously. The tool supports devices connected to the USB ports either in the HID or CDC modes only. The tool does not support devices connected to the PC via non-USB ports, such as the RS232 ports.

The example of the CFG tag section in the *ucl2.xml* file contains all the possible identification parameters known at the time of the release of this version of the document:

```
<CFG>
    <STATE name="Blhost" dev="KBL-CDC" vid="1366" pid="1015"/>   <!--JLINK CDC-->
    <STATE name="Blhost" dev="KBL-CDC" vid="0d28" pid="0204"/>   <!--mBed CDC-->
    <STATE name="Blhost" dev="KBL-CDC" vid="1357" pid="0707"/>   <!--OpenSDK CDC-->
```

```
    <STATE name="Blhost" dev="KBL-CDC" vid="1357" pid="0089"/>   <!--OpenSDA CDC-->
    <STATE name="Blhost" dev="KBL-CDC" vid="1a86" pid="7523"/>   <!--CH340 CDC-->
    <STATE name="Blhost" dev="KBL-CDC" vid="067b" pid="2303"/>   <!--PL2303 CDC-->
    <STATE name="Blhost" dev="KBL-HID" vid="15A2" pid="0073"/>   <!--KBL USB-HID-->
</CFG>
```

The example lists all the possible parameters to save you the time to find them. In an actual use case, you should comment the configuration lines that they do not want to use the corresponding ports for manufacturing. All configuration lines in the CFG tag section are of the same priority. If the number of devices connected to the host is bigger than the number of panels you configured, only the first-enumerated devices are displayed in the panels and the left ones are ignored.

The below figure shows the Windows OS Device Manager showing one MCU bootloader device connected in the USB-HID mode and two devices connected in the USB CDC mode, accessible via serial COM ports 3 and 6.

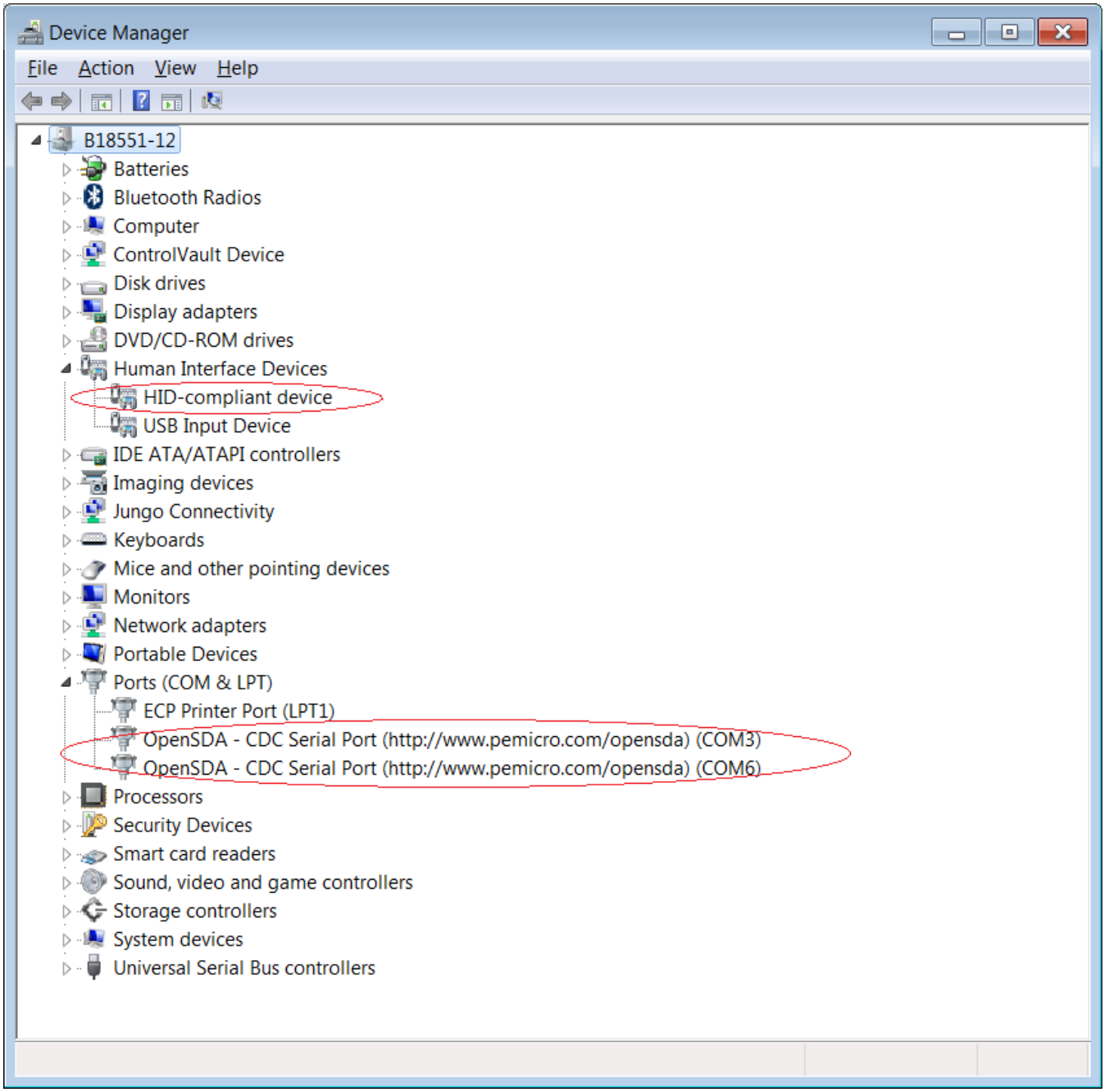

Figure 9. Windows OS Device Manager showing MCUs connected in the USB-HID and UART modes

The figure below shows the corresponding 4-panel MfgTool user interface for the above-connected devices showing one device connected in the HID mode, two devices connected in the CDC mode, and the last panel displaying as "No device connected".

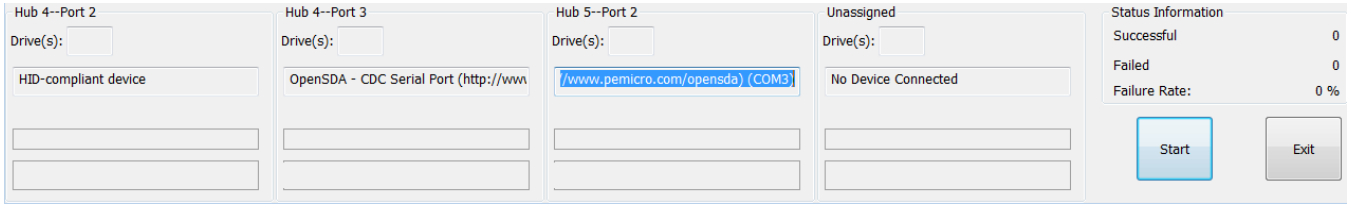

Figure 10. 4-panel UI with three devices connected

NOTE

See the *MCU Bootloader Demo Applications User's Guide* (document ID MBOOTDEMOUG) available at www.nxp.com/MCUBOOT page for instructions on how to connect the target platform MCU.

4.4.2 Sequencing MCU bootloader commands

The MCU bootloader provides a set of commands to enable the host to find information about the device and to perform operations on the device's flash memory such as read memory, write memory, erase memory, get or set properties, etc. The list of commands and properties supported by the MCU bootloader are documented in the MCU Bootloader Reference Manual available at www.nxp.com/MCUBOOT.

The blhost tool is the NXP implementation of Windows OS PC host tool that communicates with MCU bootloader device connected to host via UART or USB-HID interface. The blhost tool is command line driven, the bootloader command and its parameters are passed on the command line and by issuing a sequence of commands full firmware update on the device can be achieved. For complete documentation and usage of blhost, see the *blhost User's Guide* (document ID MCUBLHOSTUG) available at www.nxp.com/MCUBOOT.

MfgTool support for the MCU bootloader device is limited to identifying the device's presence, and using blhost as its backend to send commands to the MCU bootloader device. To complete manufacturing the device, the MfgTool expects the blhost commands and parameters to appear in the *body* for each CMD listed in the ucl2.xml file, such as:

```
<CMD state="Blhost" type="blhost" body="flash-erase-region 0xA000 0x800">Flash Erase Region</
CMD>
<CMD state="Blhost" type="blhost" body="read-memory 0xA000 0x800">Read Memory</CMD>
<CMD state="Blhost" type="blhost" body="write-memory 0xA000
 \"Profiles\\Kinetis Bootloader\\OS Firmware\\files\\simple.bin\"">Write Memory</CMD>
<CMD state="Blhost" type="blhost" body="read-memory 0xA000 0x800">Read Memory</CMD>
```

The MfgTool cannot send any command to the bootloader directly, and instead uses *blhost.exe* for that purpose. The MfgTool does not decide what command to send, so the ucl2.xml should be crafted with the commands and its parameters needed to send to the device.

Each blhost command appears with separate <CMD> tag. The *state* and *type* indicates bootstrap operation using blhost. The actual blhost command line arguments goes with the *body* tag.

Example:

```
<CMD state="Blhost" type="blhost" body="write-memory 0x0 demo.bin">Write Memory</CMD>
```

In the above example, blhost is invoked for write-memory command with 0x0 and demo.bin as two arguments; 0x0 is the start address location to write the contents of the file demo.bin.

NOTE

The default location where MfgTool locates the demo.bin is the folder where the MfgTool.exe is placed, i.e., "bin \tools\mfgtools-rel". The complete path to demo.bin should be specified in the argument if demo.bin is placed elsewhere in the directory structure. For example: `<CMD state="Blhost" type="blhost" body="write-memory 0xA000 \"Profiles\\Kinetis Bootloader\\OS Firmware\\files\\demo.bin\"">Write BIN to A000</CMD>`

The last section of the <CMD> tag is the description text that will appear on MfgTool GUI when MfgTool is invoking the command and collecting its response. In the given example, “Write Memory” appears on the GUI.

The ucl2.xml does not need to specify the arguments needed for the type of connection such as --port or –usb, which is only necessary when using blhost in standalone mode. The MfgTool automatically provides the type of connection arguments to blhost.exe depending on the type of connection.

The MfgTool collects the device’s response that blhost outputs on the stdout, logs the response to MfgTool.log file, and inspects the results, reporting any error on the UI.

4.4.3 Example ucl2.xml files

This section provides the typical *ucl2.xml* file content and shows how the different types of image files can be sent to the blhost using different commands, such as write-memory, flash-image, and receive-sb-file.

This is an example of a binary format file passed in an argument for the write-memory-command:

```
<CMD state="Blhost" type="blhost" timeout="5000" body="flash-erase-region 0xA000
0x800">Flash Erase Region</CMD>
<CMD state="Blhost" type="blhost" body="write-memory 0xA000
 \"Profiles\\Kinetis Bootloader\\OS Firmware\\files\\demo.bin\"">Write Memory</CMD>
<CMD state="Blhost" type="blhost" body="Update Completed!">Done</CMD>
```

These are the examples of the srec and hex format files passed with the flash-image command:

```
<CMD state="Blhost" type="blhost" timeout="5000" body="flash-image
 \"Profiles\\Kinetis Bootloader\\OS Firmware\\files\\simple.hex\" erase">Flash Image</CMD>
<CMD state="Blhost" type="blhost" body="Update Completed!">Done</CMD>
```

```
<CMD state="Blhost" type="blhost" timeout="5000" body="flash-image
 \"Profiles\\Kinetis Bootloader\\OS Firmware\\files\\simple.srec\" erase">Flash Image</CMD>
<CMD state="Blhost" type="blhost" body="Update Completed!">Done</CMD>
```

These is an examples of a SB (Secure Binary) file passed in an argument for the receive-sb-file command:

```
<CMD state="Blhost" type="blhost" timeout="5000" body="receive-sb-file
 \"Profiles\\Kinetis Bootloader\\OS Firmware\\files\\simple.sb\"">Receive SB File</CMD>
<CMD state="Blhost" type="blhost" body="Update Completed!">Done</CMD>
```

4.4.4 Manufacturing process

This section shows how to execute the MfgTool for the MCU device manufacturing. The key to the MCU bootloader manufacturing with the MfgTool is to have the *ucl2.xml* file completed with all needed commands that are necessary to successfully program the device in a sequence.

The typical setup during manufacturing is shown in the image below with four devices connected to four USB ports to the PC running the MfgTool application.

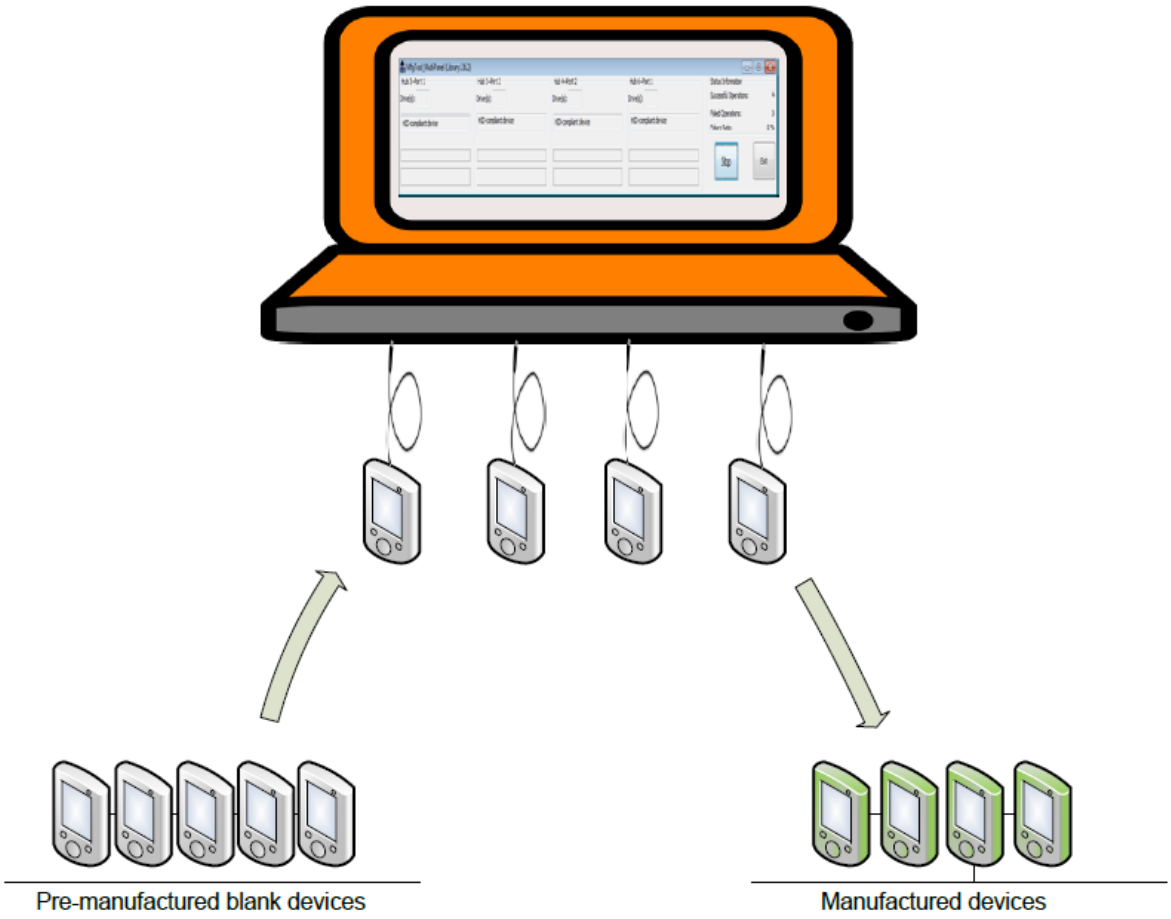

Figure 11. Typical setup at manufacturing

The manufacturing process begins when the operator launches the MfgTool application and clicks the *Start* button.

The MfgTool provides the UI to show the device update progress for each device connected to the PC. The UI also shows the description text that appears in the *ucl2.xml* file for the command that is in execution. To indicate the end of programming for a device, a dummy command can be placed at the end of the *ucl2.xml* file with the text “Done” to conveniently identify the completion of the update so that you know when to switch to the new blank device. The example dummy command to indicate the completion of the update looks like this:

```
<CMD state="Blhost" type="blhost" body="Update Complete!">Done</CMD>
```

The MfgTool is in a continuous update mode until it is stopped by the user by pressing the "Stop" button. No user interaction is necessary on the UI when the "Start" button is pressed, except for pulling off the manufactured device when complete and connecting to the new blank device.

5 Troubleshooting guide

The MfgTool logs the command and response from the device into the *MfgTool.log* file. When the device returns a failure code for a command or times out or for any other reason, the MfgTool UI indicates the occurrence of the failure. The cause can be diagnosed by inspecting the *MfgTool.log* file. To open the *MfgTool.log* file, it is recommended to stop the MfgTool using the "Stop" button, because the MfgTool continuously logs information to the file.

Example:

In this example, the "*Get Property 1*" command was sent to two devices. One device failed and one succeeded. Here is the corresponding MfgTool UI:

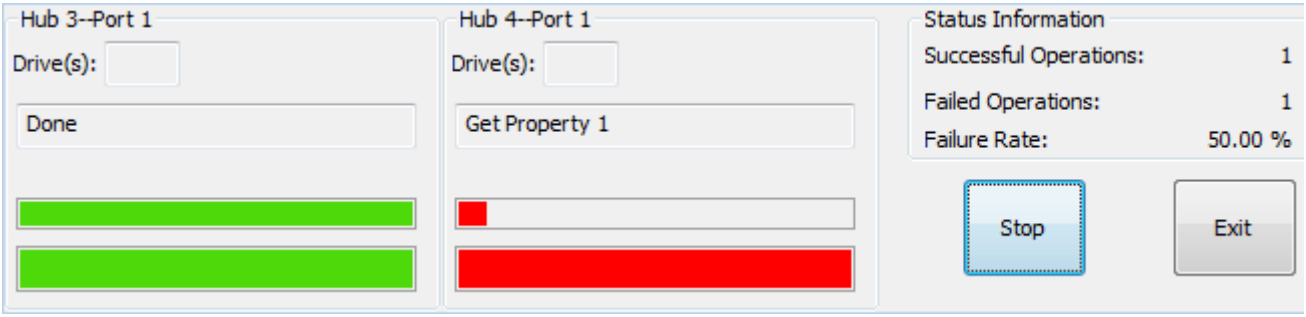

Figure 12. UI showing successful update for the first device and error for the second device

The red progress bar indicates that the device connected to the Hub 4-Port 1 failed. The corresponding result, logged to the *MfgTool.log* file looks as follows:

```
ModuleID[2] LevelID[10]: CmdOperation[0] device changed and reset to state 0
ModuleID[2] LevelID[10]: ExecuteCommand--Blhost[WndIndex:0], Body is get-property 1
ModuleID[2] LevelID[10]: CmdOperation[1] device changed and reset to state 0
ModuleID[2] LevelID[10]: ExecuteCommand--Blhost[WndIndex:1], Body is get-property 1
ModuleID[2] LevelID[10]: Get Property 1  [WndIndex:0] {
   "command" : "get-property",
   "response" : [ 1258357504 ],
   "status" : {
      "description" : "0 (0x0) Success.",
      "value" : 0
   }
}

ModuleID[2] LevelID[10]: Get Property 1  [WndIndex:1] {
   "command" : "ping",
   "response" : [],
   "status" : {
      "description" : "10500 (0x2904) No response received for ping command.",
      "value" : 10500
   }
}
```

The "**WndIndex**" shows the device index on the MfgTool UI for which the results are logged. The first two lines indicate the get-property 1 sent to two devices with WndIndex 0 and 1.

The next lines are the capture of the blhost output on the stdout. The blhost output suggests that one device did not respond to the ping sent by the blhost and returned error code 10500. The other device returned success code (0) for the Get Property 1 command and the response text shows the exact value returned by the device for the Get Property 1 command.

5.1 Examples of common failure error messages in the log

5.1.1 Cannot find ..\blhost\win\blhost.exe

The failure reports for missing blhost.exe in the folder blhost\win\. The tool searches for blhost.exe in the bin\tools\blhost\win folder. Make sure it is available in the correct folder.

5.1.2 No response received for the ping command

```
{
   "command" : "flash-erase-all-unsecure",
   "response" : [],
```

```
   "status" : {
      "description" : "10000 (0x2710) kStatus_UnknownCommand",
      "value" : 10000
   }
}
```

There could be several reasons for such error. Here are some troubleshooting steps:

- See the reference manual for the device to ensure that the device is supported by the MCU bootloader.
- Check whether the device is powered up.
- The device may boot off the image on the flash and not the MCU bootloader image from the ROM or flash. Erase the flash memory and try again to enable the device to boot into the MCU bootloader mode.
- Direct boot feature can be enabled. Erase the flash and try again to boot into the MCU bootloader.

5.1.3 UnknownCommand

```
{
   "command" : "flash-erase-all-unsecure",
   "response" : [],
   "status" : {
      "description" : "10000 (0x2710) kStatus_UnknownCommand",
      "value" : 10000
   }
}
```

The blhost.exe can execute all MCU bootloader commands. However, the command itself may not be supported by the target MCU bootloader device. See the ROM Bootloader/Flashloader chapter of the device reference manual to check whether the command is supported.

5.1.4 Command disallowed when security is enabled

```
{
   "command" : "flash-erase-all",
   "response" : [],
   "status" : {
      "description" : "10001 (0x2711) Command disallowed when security is enable
d.",
      "value" : 10001
   }
}
```

The device is in a secure state and cannot be programmed. To program a secured device, add the unlock command to the ucl2.xml file.

5.1.5 MemoryRangeInvalid

```
{
   "command" : "write-memory",
   "response" : [],
   "status" : {
      "description" : "10200 (0x27D8) kStatusMemoryRangeInvalid",
      "value" : 10200
   }
}
```

The memory range may be in the reserved region used by the bootloader. See the device reference manual for the available memory region to program the flash and fix the address range for the command accordingly.

5.1.6 FlashCommandFailure

```
{
   "command" : "write-memory",
   "response" : [],
   "status" : {
      "description" : "105 (0x69) kStatus_FlashCommandFailure",
      "value" : 105
   }
}
```

The possible cause for the failure may be the flash region not being erased before writing to it. A flash-erase-region command must be called before writing to it.

6 Revision history

The following table contains a history of changes made to this user's guide.

Table 1. Revision history

Revision number	Date	Substantive changes
0	04/2016	Initial Kinetis bootloader v2.0.0 release
1	05/2018	MCU Bootloader v2.5.0 release
2	09/2018	MCU Bootloader v2.6.0 release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support