

MBOOTFLTOOLUG

Kinetis Flash Tool User's Guide

Rev. 4 — 28 December 2020

User's Guide

1 Introduction

The Kinetis Flash Tool is a GUI application for Windows® OS which provides a user friendly way to communicate with the Kinetis device running the MCU bootloader application. Write or erase operation can be executed on the non-volatile memory of the Kinetis device, including (but not limited to) these features:

- Supports the Kinetis devices with MCU ROM bootloader, MCU flashloader, or running the flash resident bootloader.
- Supports UART with a baud rate ranging from 4800 to 256000 bit/s along with a user-defined baud rate in the supported range.
- Supports the USB-HID with the default or user-defined VID/PID.
- Supports the binary file (*.bin*), SB file (*.sb*), HEX file (*.hex*), and SRecord file (*.s19*, *.srec*).
- Supports the “Quick Update” without initial connection to the Kinetis device.
- Supports the flash-erase operation.
- Supports programming of the flash non-volatile information register (IFR).
- Supports generating the Bootloader Configuration Area (BCA) binaries and the C code file.
- Supports the CRC calculation of the user application firmware (only binary files).

This document describes the function, user interface, and use of the Kinetis Flash Tool.

2 System Requirements

Table 1. Requirements

Component	Requirement
Processor	1 GHz or faster x86/x64 processor
Memory	1 GB RAM (32-bit); 2 GB RAM (64-bit)
Operating system	Windows OS XP, Windows OS 7, Windows OS 8, Windows OS 10

3 Tool structure

The Kinetis Flash Tool is available at `middleware/mcu-boot/bin/Tools` in SDK package which can be downloaded from mcuxpresso.nxp.com. This tool consists of the files shown in the following figure.

Contents

1	Introduction.....	1
2	System Requirements.....	1
3	Tool structure.....	1
4	User interface.....	3
5	Typical usage.....	29
6	Revision history.....	47



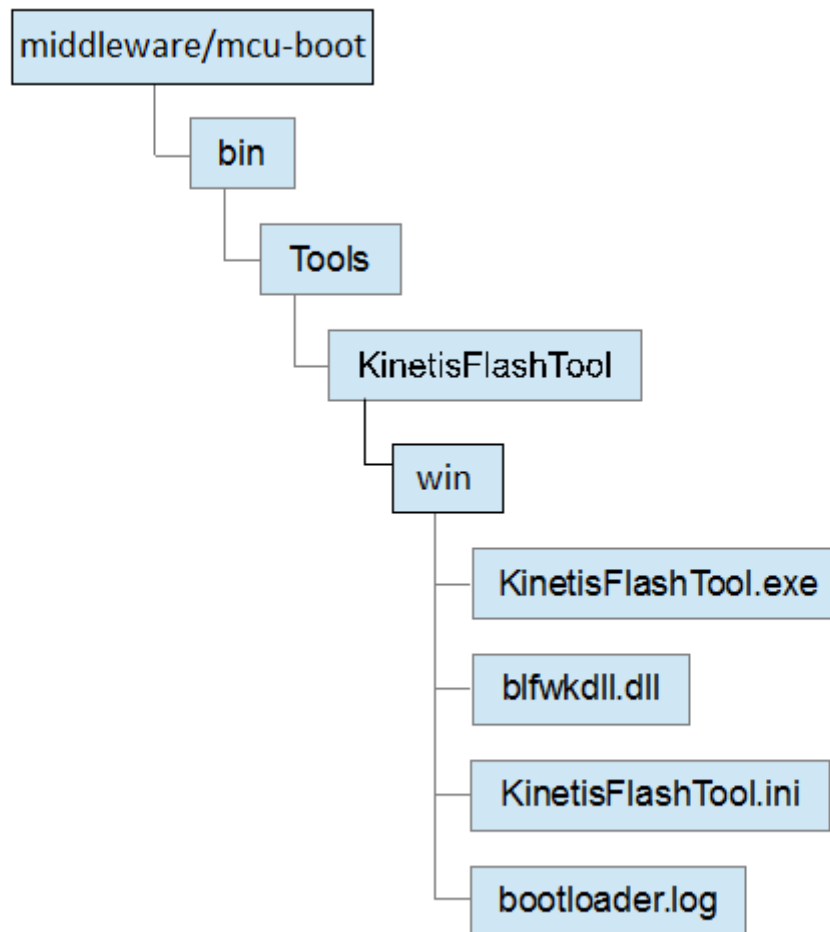


Figure 1. Kinetis Flash Tool structure

In the release package, the KinetisFlashTool folder appears in the *middleware/mcu-boot/bin/Tools* folder. It contains four files.

- The *KinetisFlashTool.exe* file is the executable file.
- The *blfwkdll.dll* file is the dynamic link library used by the Kinetis Flash Tool and is responsible for the communication with the MCU bootloader.

NOTE

The *blfwkdll.dll* file for the Kinetis Flash Tool is different from the file used by the Kinetis Updater application in the Kinetis bootloader v1.2 release package. Do not move, edit, or exchange them.

- The *KinetisFlashTool.ini* file is the initialization/configuration file. The Kinetis Flash Tool loads the history information from this file at startup and saves it when it is existing, not during the execution. See the *KinetisFlashTool.ini* file for more details.
- The *bootloader.log* file contains the detailed data transferring between the host and target devices during the lifetime of this program. This information helps in review and troubleshooting.

NOTE

1. The *KinetisFlashTool.ini* and *bootloader.log* files do not already exist in the package. They are created after the program launches for the first time. This is only required for historical data else can be deleted.
2. The text in the *bootloader.log* file is different from the text in the log window. The former offers the detailed data transferred by the Kinetis Flash Tool. The latter shows the results of the user operation. See the "Log window" section for more details about the log window.

3.1 KinetisFlashTool.ini file

This file saves the historical information used by the Kinetis Flash Tool. This section helps understanding the structure of this file. The following figure displays a typical *KinetisFlashTool.ini* file.

```
[Port]
Current=COM4
[Baudrate]
List=4800;9600;19200;38400;56000;57600;115200;128000;256000
Current=115200
[VID]
List=0x15A2;0x1234;0x4321
[PID]
List=0x0073;0x5678;0x8765
[File]
List=C:\Image\led_demo.bin;C:\Image\led_demo.sb;C:\Image\Test Image Files\led_demo.hex
Address=0x0000a000
Backdoorkey=0102030405060708
```

Figure 2. Example for KinetisFlashTool.ini

- *[Port]Current*: Saves the pre-selected COM port. If the pre-selected COM port does not exist on the host, the first device in the port combo box is selected.
- *[Baudrate]List*: Contains the baud rates that are listed in the baud rate combo box.
- *[Baudrate]Current*: Saves the pre-selected baud rate. If it does not exist in the "List", the first one is selected.
- *[VID]/[PID]List*: Contains the VID/PIDs that is listed in the VID/PID combo box.
- *[File]List*: Contains the full paths of image files that are listed in the image file combo box.
- *[File]Address*: Saves the previous image address; the firmware is downloaded to this address.
- *[File]BackdoorKey*: Saves the previous backdoor key.

The parameters in the *List* file must be separated by separator ";" (half width). Only a maximum of ten parameters in the front of the line is loaded at the startup.

Except the backdoor key, the numerical parameters must follow the rule that hex digits must have the "0x" as a prefix, octal digits must have the "0" as a prefix, and a non-numerical character is unacceptable. The backdoor key must be 16 hex digits without the "0x" prefix. Any invalid parameters are ignored and abandoned during initialization of the program. Files in the *[File]List* must exist in the file system or are ignored at startup.

4 User interface

This section describes the UI pages shown in the Kinetis Flash Tool to gather necessary information and user operation intentions.

4.1 Main window

The main window consists of the four parts shown in the following figure.

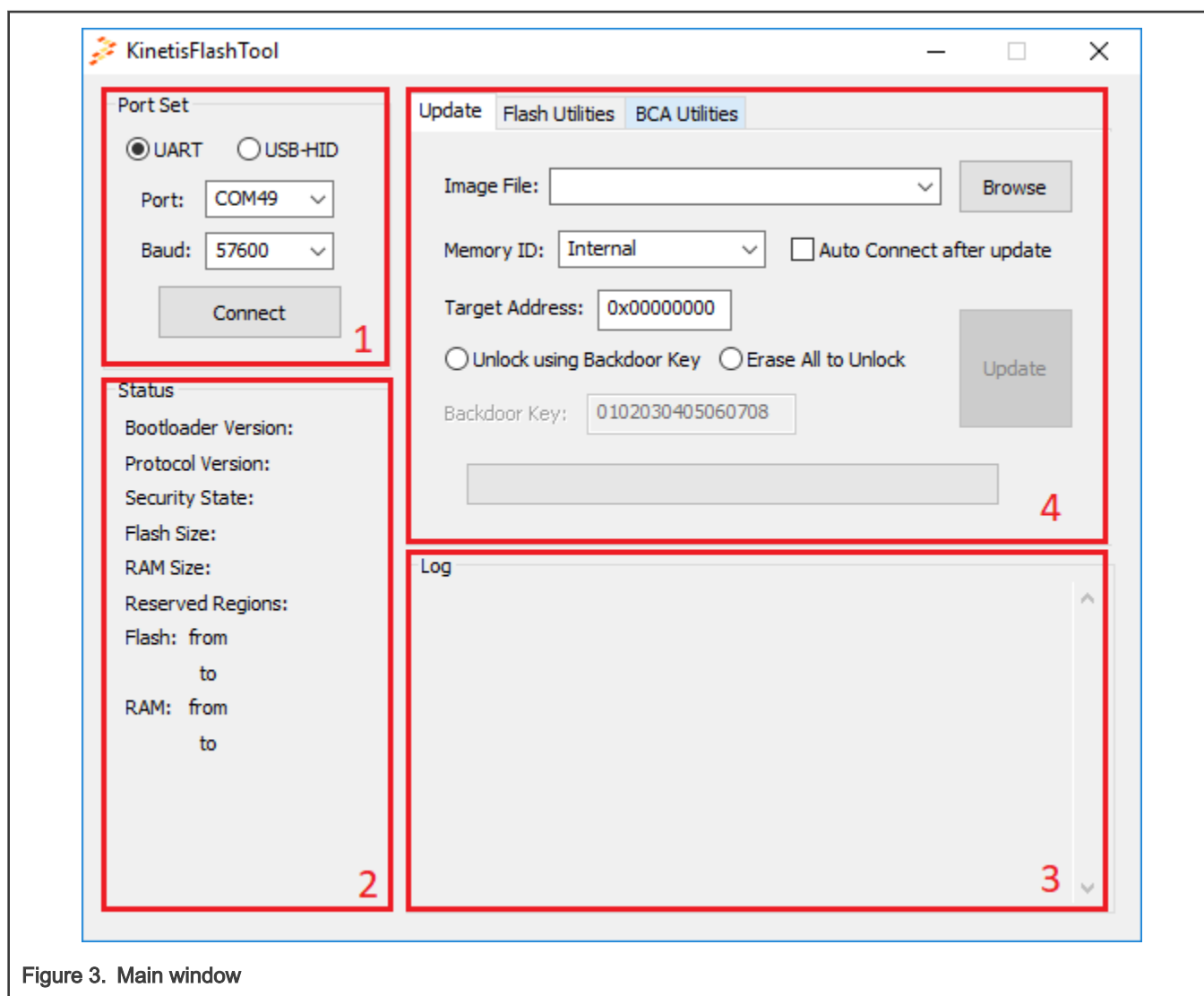
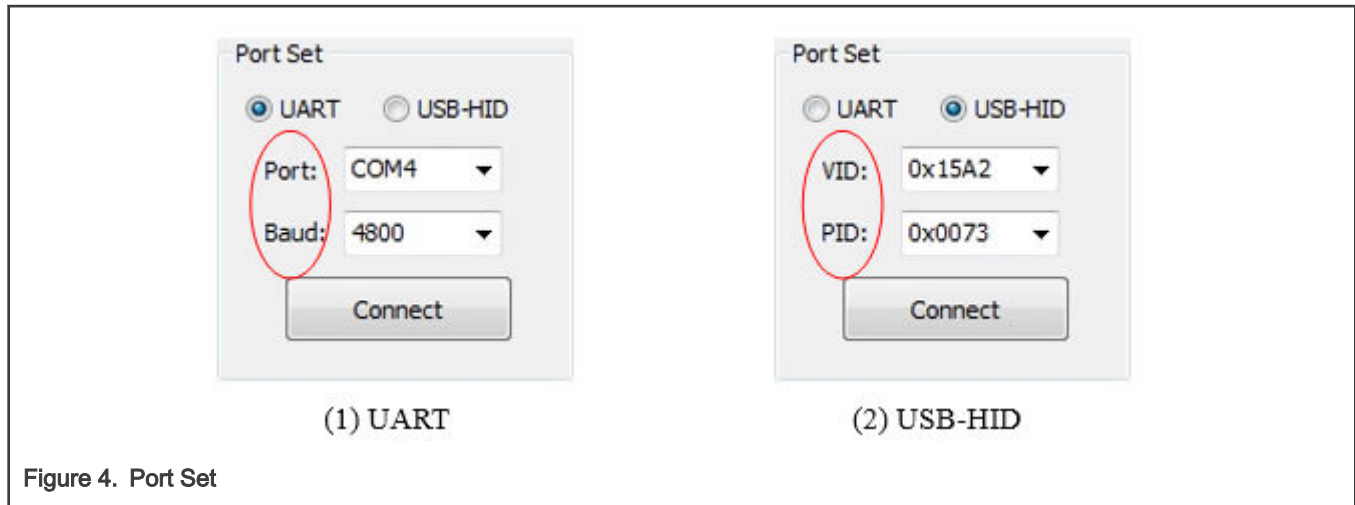


Figure 3. Main window

4.1.1 Port Set

The *Port Set* part is used to select the peripheral and corresponding configuration which is used to communicate with the target device.



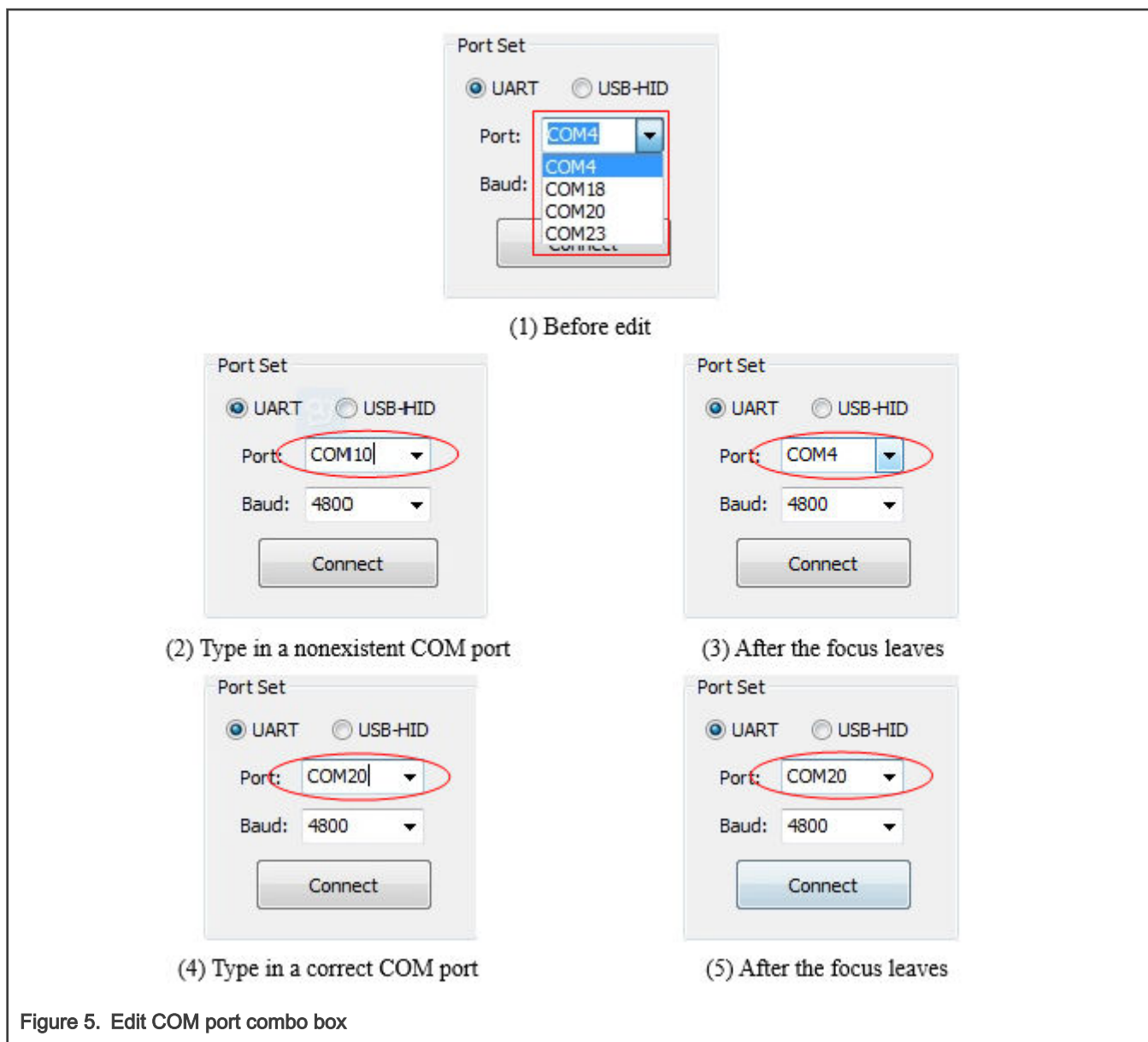
4.1.1.1 Peripheral selection radio button

Currently, only the UART and USB-HID are supported by the Kinetis Flash Tool. Select one of them to establish the communication between the host and target devices. UART is selected by default when the program starts.

4.1.1.2 COM port combo box

Displays the currently selected COM device. The drop-down list holds all COM ports on the host (PC) in ascending order. Every time the program starts, it scans the Windows OS registry to find all serial devices. Both the RS232 devices and the USB-CDC devices are supported.

A COM port can be selected from the drop-down list or typed in. If you type an illegal baud rate (non-numerical characters), the combo box selects the previous baud rate when the focus leaves the text box of this control.



4.1.1.3 Baud rate combo box

Displays the currently selected baud rate. The drop-down list displays some typical baud rates supported by the MCU bootloader in the ascending order.

NOTE

The supported baud rate range is different on different target devices. See the reference manual to ensure usage of correct and supported baud rate.

At startup, the program loads a maximum of ten baud rates at the beginning of the list. If the configuration file is absent, the program lists nine typical baud rates, as shown in the following figure.

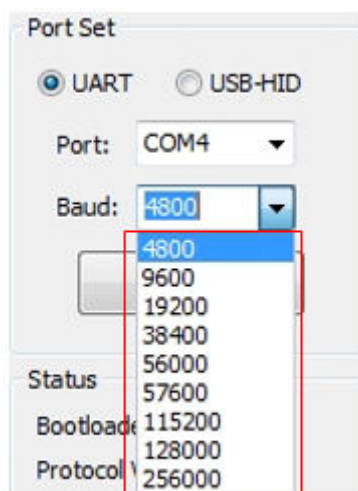


Figure 6. Baud rate list (default)

You may select one from the drop-down list or simply type your defined baud rate. If you type an illegal baud rate (non-numerical characters), the combo box selects the previous baud rate when the focus leaves the text box of this control.

If you type a valid baud rate that does not exist in the list, the baud rate is then added to the list.

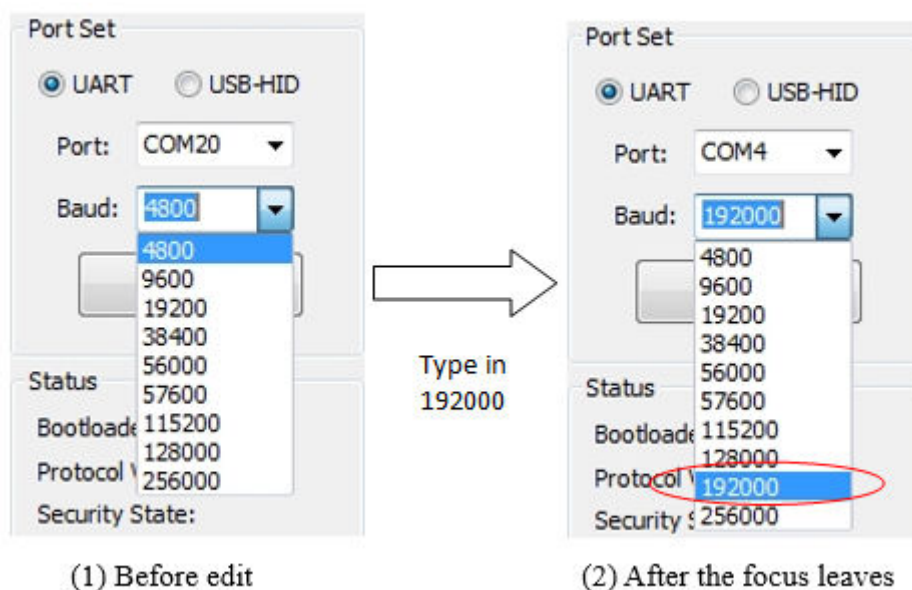


Figure 7. Insert new baud rate list

4.1.1.4 VID combo box

Displays the currently selected vendor ID. The drop-down list displays the recently used VIDs. The most recently selected VID is always at the top of the list.

Similar to the baud rate combo box, the program loads 10 VIDs at most. If the configuration file is absent, there is only one VID (MCU bootloader default VID (0x15A2)) listed in the combo box.

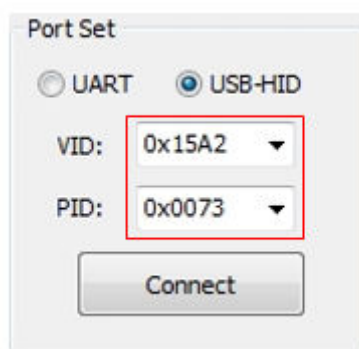


Figure 8. Default VID and PID

You may select it from the drop-down list or just type in a user-defined VID, similarly to the behavior of the baud rate combo box.

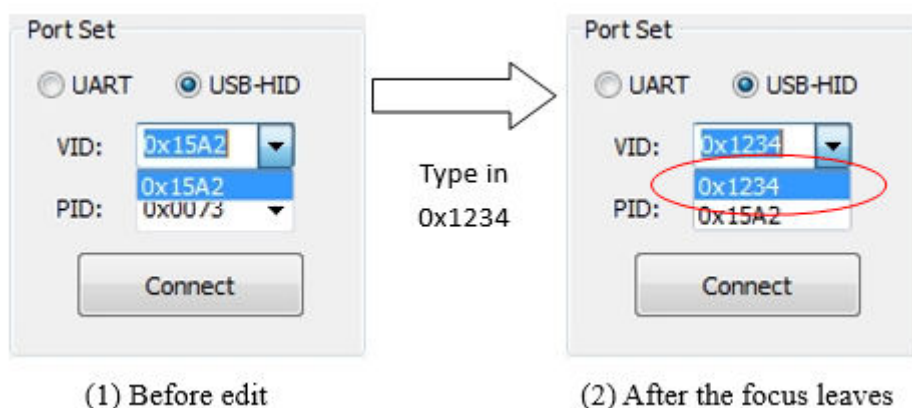


Figure 9. Insert a new user-defined VID

When a VID is selected, it comes on the top of this list.

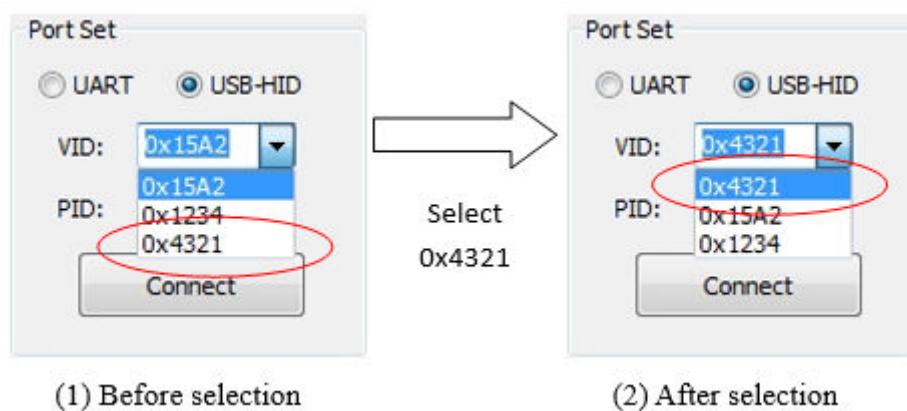


Figure 10. Selection change of VID

4.1.1.5 PID combo box

Displays the currently-selected product ID. The drop-down list holds the most recently used PIDs. It has the same behavior as the VID combo box. See the "VID combo box" section.

4.1.1.6 Connect or Reset button

The "Connect" or "Reset" button sends a get-property command to the MCU bootloader using the current peripheral configuration if there is no connected device. It sends a reset command if the communication is established.

Sending a get-property command tries to establish the communication with the MCU bootloader. When the communication is established, this sets the selected peripheral as active in the MCU bootloader and shuts down other peripherals. The text on the button changes from "Connect" to "Reset".

When the "Connect" button is clicked, the Kinetis Flash Tool keeps sending the get-property command until you click the "Reset" button, changes the peripheral configuration, or updates the firmware without checking the "Auto Connect" box. You may exchange the device during this time and the tool disconnects the removed device and connects the new device automatically.

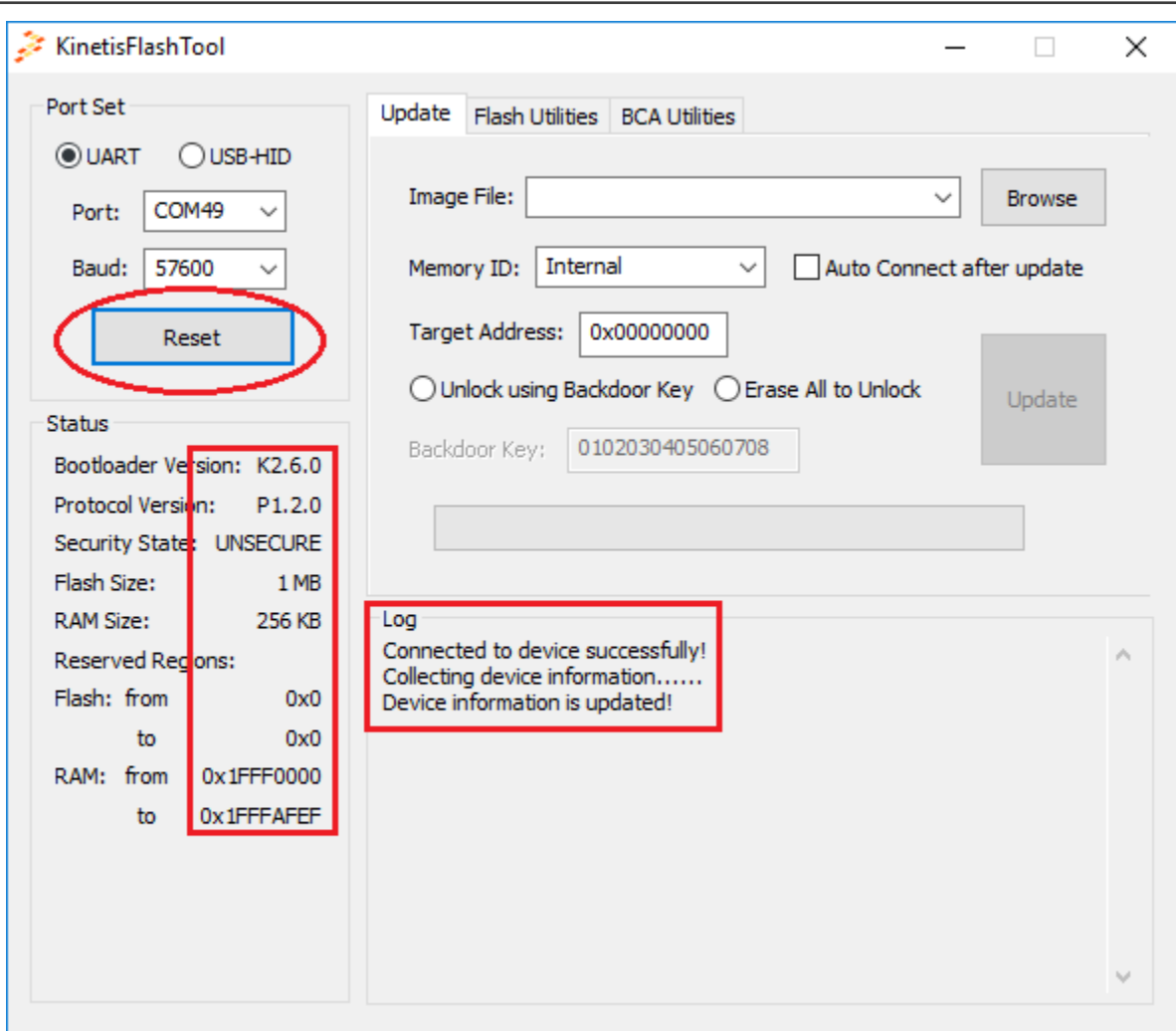


Figure 11. Connect to a device successfully

Sending a reset command disconnects the current device and puts the MCU bootloader back to the get-active-peripheral state. In this state, all peripherals are enabled again and are ready to be selected. The text on the button changes back to "Connect", once reset command is executed.

NOTE

1. Once the communication is established, any change in peripheral configuration sends a reset command automatically.
2. If an operation is being executed for example, update/erase/program IFR, connect/reset operation, get-property command, or reset command, no command is sent even if this button is pressed.

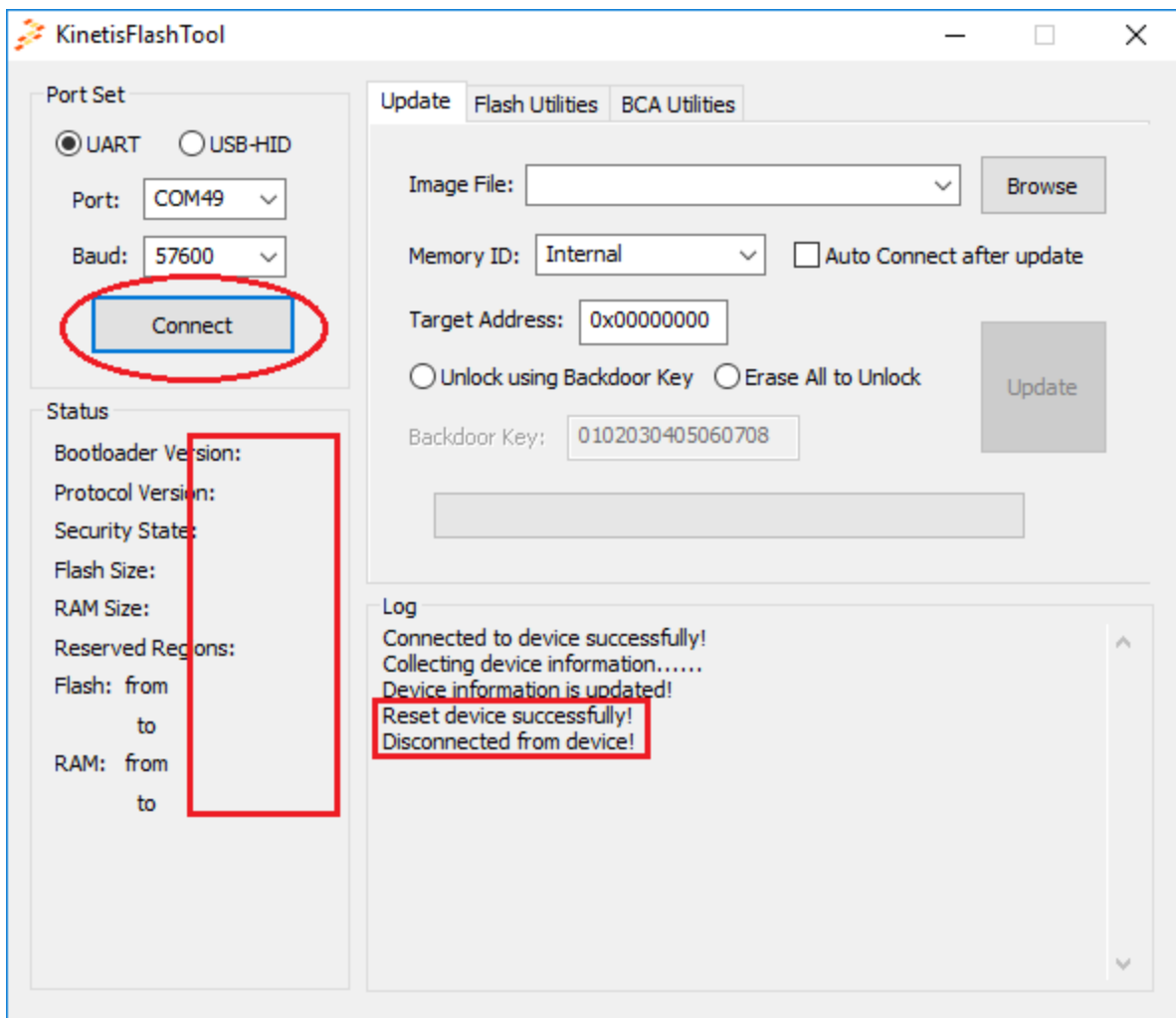


Figure 12. Reset the device

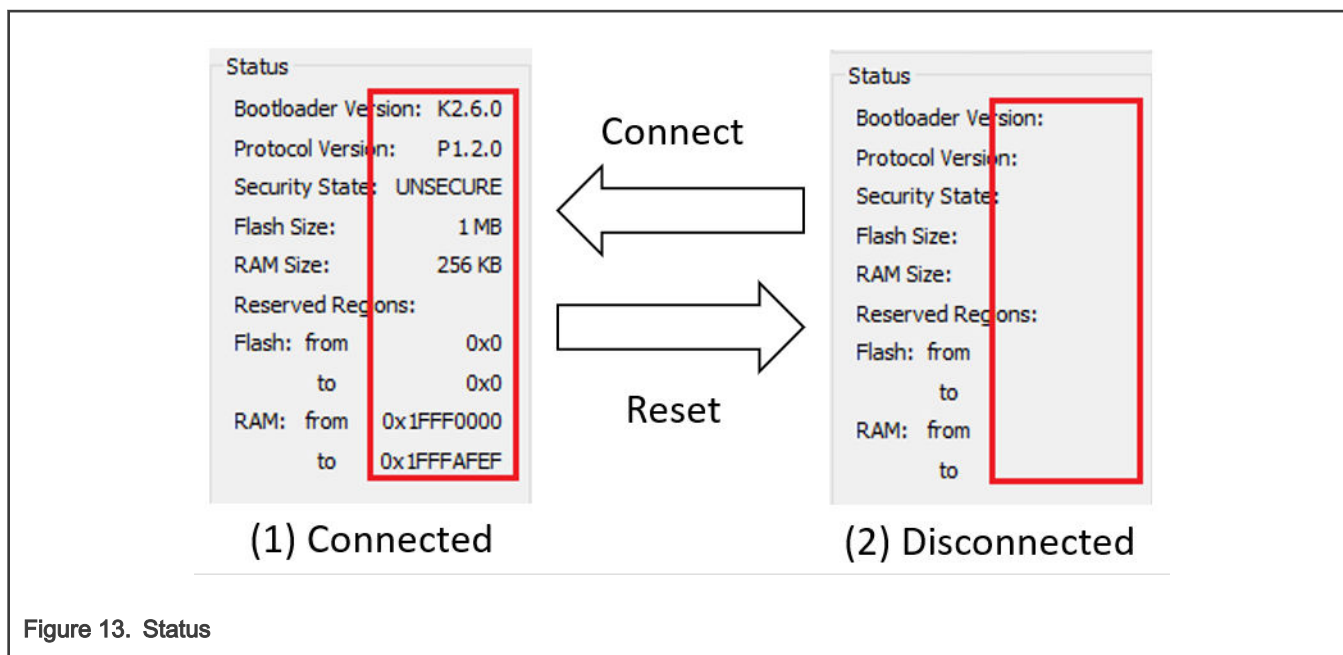
4.1.2 Status

This panel displays properties of the target device and the MCU bootloader running on the device. The information in this region is valid only when the communication is established.

Kinetis Flash Tool sends the get-property commands to collect this information once successfully connected to a device. After this process, all information is displayed in this region. Sending a reset command clears the information at the same time.

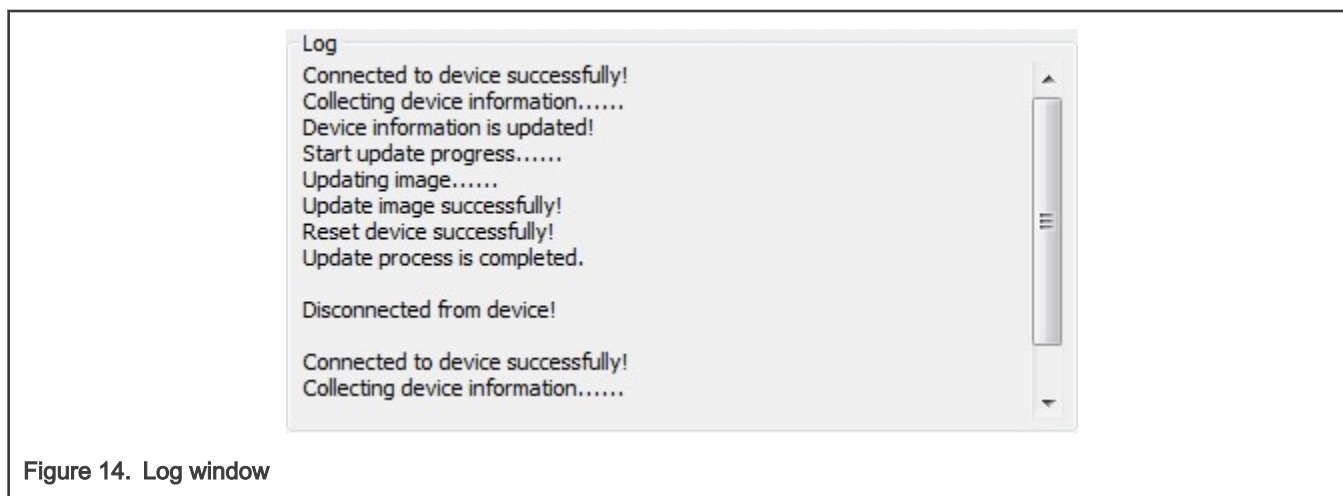
NOTE

Protocol Version is empty when connecting via USB-HID, because USB-HID uses the USB communication protocol, not MCU bootloader serial protocol.



4.1.3 Log window

Shows the results, warnings, and errors of the user operations while being used.



4.1.4 Tab page

Currently, there are three tab pages in the Kinetis Flash Tool, and each of them is responsible for one function.

- The *update* page is for writing the user application firmware to the Kinetis device non-volatile memory. See the "Update tab page" section for details.
- The *flash utilities* page is for flash erase and flash IFR programming. See the "Flash utilities tab page" section for details.
- The *BCA utilities* page is for the BCA-related operation. See the "BCA utilities tab page" section for details.

4.2 Update tab page

The update tab page is used to launch the update process that writes the application image to the device.

The screenshot shows the 'Update' tab page of the Kinetis Flash Tool. At the top, there are three tabs: 'Update', 'Flash Utilities', and 'BCA Utilities'. The 'Update' tab is selected. Below the tabs, there are several input fields and controls:

- Image File:** A text box containing the path 'r\release\ed_demo_freedom_0000.bin' with a dropdown arrow on the right. A 'Browse' button is to its right.
- Memory ID:** A dropdown menu showing 'Internal'.
- Auto Connect after update:** An unchecked checkbox.
- Target Address:** A text box containing '0x00000000'.
- Unlock options:** Two radio buttons. The first is 'Unlock using Backdoor Key' (selected). The second is 'Erase All to Unlock'.
- Backdoor Key:** A text box containing '0102030405060708'.
- Update button:** A large button on the right side of the form.

Figure 15. Update tab page

4.2.1 Image file combo box

Displays the full path to the file name of the currently selected application image. The drop-down list holds the recently-used full-path file names. The most recently selected file name is always at the head of the list.

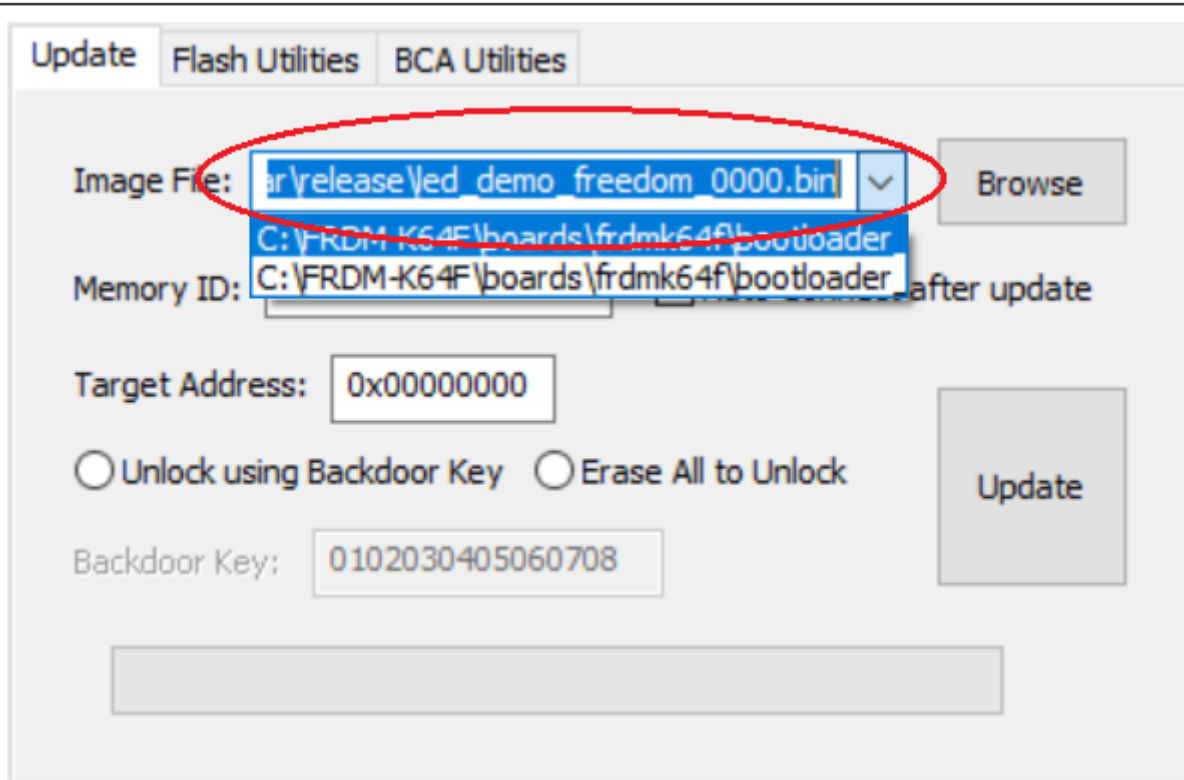


Figure 16. Image file combo box

Similar to the VID/PID selection combo box, the program loads a maximum of ten file names from the configuration file. If the configuration file is absent, the drop-down list is empty. Select a file from the drop-list of the control, type in a full-path file name, or use the "Browse" button to locate the application image. If you type in a file that does not exist on the host, the textbox of this control clears when the focus leaves.

4.2.2 Browse button

Click the "Browse" button to locate the application image that is written to the Kinetis device. Currently, the supported file types are *.bin*, *.sb*, *.hex*, *.s19*, and *.srec*.

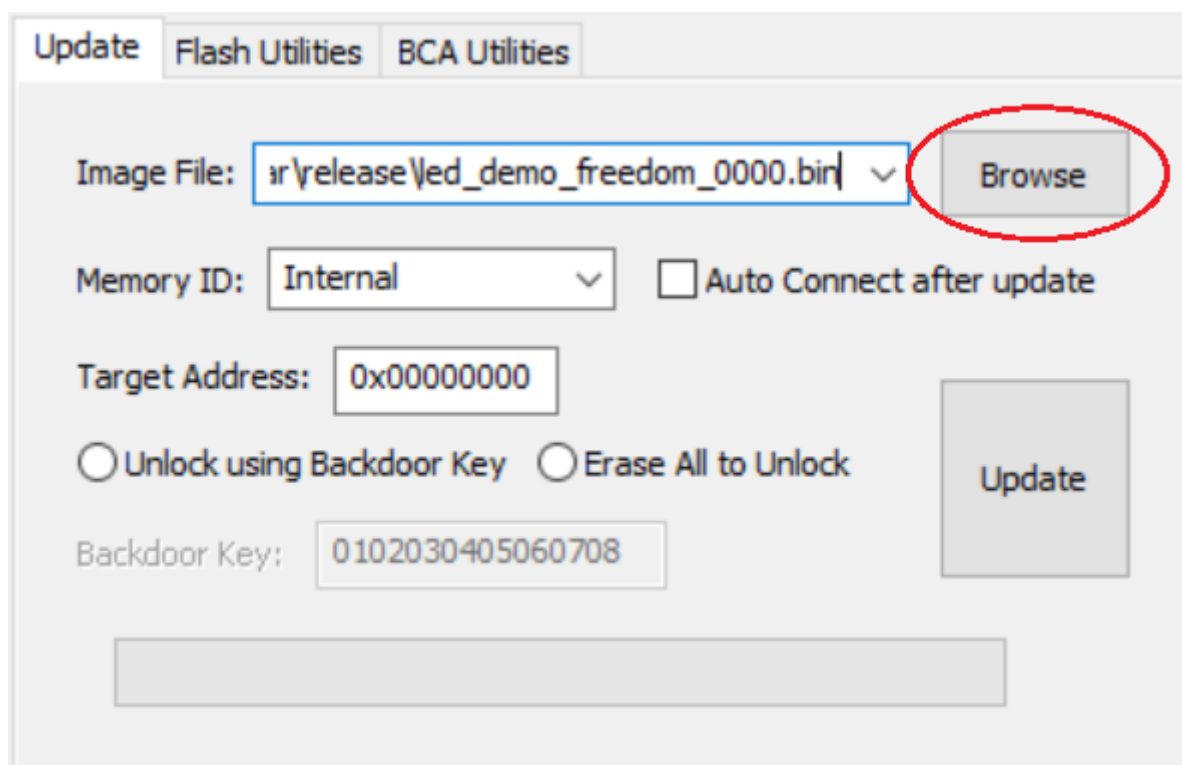


Figure 17. Browse button

4.2.3 Target address textbox

The target address textbox contains the address where the binary image is going to be placed. The address information must be completed if the selected file is binary, which does not contain the location information. For other supported file types, the textbox is not editable, and the address information is ignored.

Figure 18. Target address

The value can be any valid memory address supported by the MCU bootloader. Ensure that the memory region to be written to is not located at the reserved regions and has enough space. If you want the application to be launched automatically when reaching the timeout, the value depends on the device and implementation of the MCU bootloader.

The target address should be zero (0) for the ROM and flashloader implementations of the MCU bootloader.

For the flash-resident implementation of the MCU bootloader, the base address is determined by the macro (*BL_APP_VECTOR_TABLE_ADDRESS*) defined in the *bootloader_config.h* file, which appears in the *target/(\$device)/src* folder in the release package. *\$device* indicates the part number of the selected device.



Figure 19. Target address value in bootloader_config.h file

NOTE

The user application written to the Kinetis device must be linked so that the vector table is located at the target address.

4.2.4 Auto connect check box

The autoconnect check box determines whether to reconnect to the device after the update or not.

The screenshot shows the 'Update' tab of the Kinetis Flash Tool. It features several input fields and controls: 'Image File' with a file path, 'Memory ID' set to 'Internal', 'Target Address' set to '0x00000000', and 'Backdoor Key' set to '0102030405060708'. There are two radio buttons for unlocking: 'Unlock using Backdoor Key' and 'Erase All to Unlock'. A checkbox labeled 'Auto Connect after update' is circled in red and is currently unchecked. A large 'Update' button is positioned on the right side of the form.

Figure 20. Autoconnect check box

The Kinetis Flash Tool sends the reset command after updating the firmware, which sets the MCU bootloader back to the get-active-peripheral state and disconnects the device from the host.

If you want to run an application, ensure this box is left unchecked. The Kinetis Flash Tool stops sending the get-property command after the update progress and the MCU bootloader exits the get-active-peripheral state and launches the application after reaching the timeout. Otherwise, the Kinetis Flash Tool sends the get-property command immediately to re-establish the communication after the update progress is completed.

4.2.5 Security radio button

The security radio button offers two safety ways for you to program the Kinetis device in the secure state.

The screenshot shows the 'Update' tab of the Kinetis Flash Tool. It includes fields for 'Image File', 'Memory ID', 'Target Address', and 'Backdoor Key'. There are two radio buttons for unlocking the device: 'Unlock using Backdoor Key' and 'Erase All to Unlock'. Both are currently unselected and are circled in red. A large 'Update' button is positioned to the right of the radio buttons.

Figure 21. Security radio button

You may select one of these ways to unlock the device, or select neither option and download the firmware directly without executing the unlock operation.

NOTE

1. The MCU bootloader only supports programming of secure devices directly via the encrypted SB file. See the MCU bootloader chapter in the target device reference manual to ensure whether the SB file decryption is supported. Otherwise, the update progress fails.
2. If the device is in an unsecure state, no unsecure operation is processed regardless of the button pressed.

4.2.5.1 Erase All to unlock

Erase All sends the flash-erase-all-unsecure command to the MCU bootloader during the update progress. All space of the device's non-volatile memory is erased, and the device is set to the unsecure state. The MCU bootloader may not support the mass erase command on some devices. See the MCU bootloader chapter of the device reference manual.

NOTE

The flash-erase-all-unsecure command leaves the Kinetis device in an unsecure state until you change it to a secure state.

4.2.5.2 Unlocking using backdoor key

The unlocking using the backdoor key sends the flash-security-disable command using the backdoor key in the textbox below during the update process. If the backdoor key matches the key stored in the device, the device is set into the unsecure state until the reset.

NOTE

In the last operation of the update process, the Kinetis Flash Tool sends the reset command to set the device back into the secure state.

4.2.5.3 Backdoor key textbox

The backdoor key textbox contains 16 hex digits used to unlock the device. The textbox is enabled only when the "Unlock using Backdoor Key" radio button is selected. The key in the textbox must be 16 hex digits with no leading "0x".

4.2.6 Update button

Click the "Update" button to start the update process. The "Update" button is enabled only when a valid image file is selected.

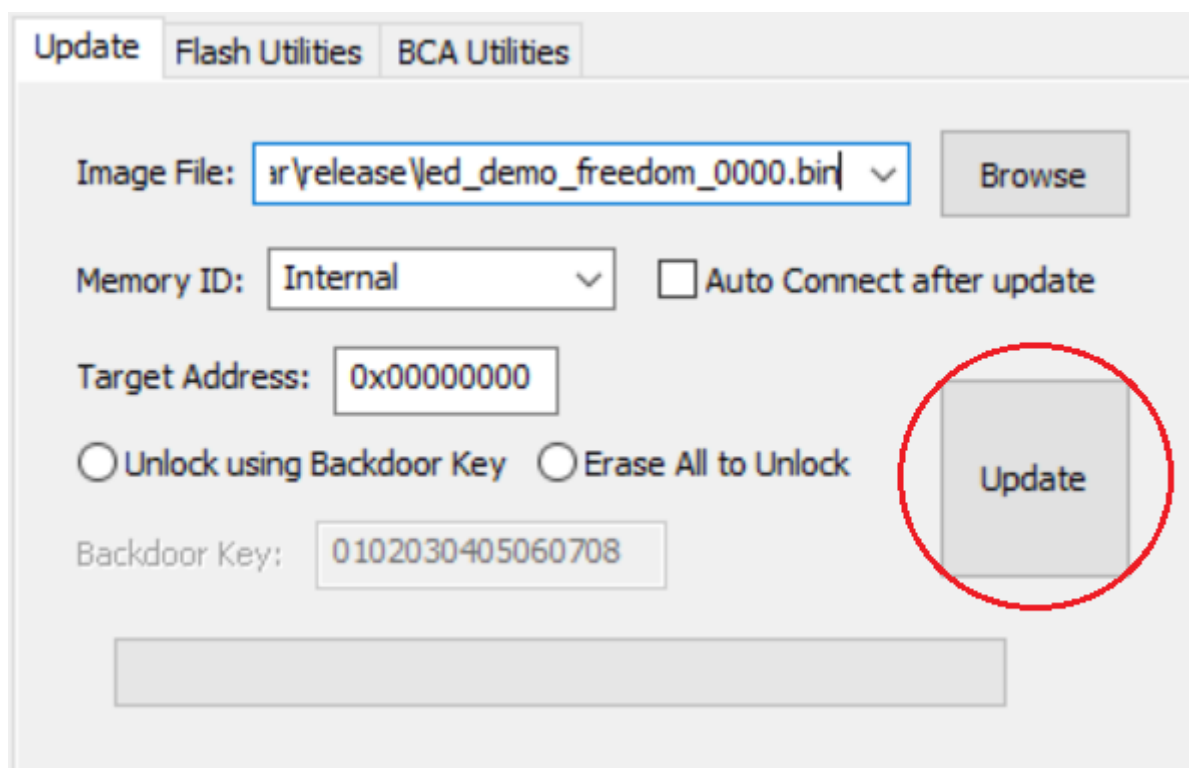


Figure 22. Update button

The update process consists of these steps:

1. Check if the communication is already established. If yes, go to Step 4. If not, go to step 2.
2. Send the get-property command to establish the connection. The Kinetis Flash Tool makes five attempts with 500 ms intervals. If the communication cannot be established, the update progress terminates.
3. When successfully connected to a device, the Kinetis Flash Tool sends the command to get the device security state.
4. If the device is in an unsecure state, or no unsecure operation is selected, go to step 6.
5. Execute the unlocking operation. For details about the unlocking operation, see the "Security radio button" section.
6. Erase the region of flash starting at the base address, with a length equal to the size of the application image file using the flash-erase-region command.

NOTE

1. The flash is erased on the sector boundaries, so the erased amount may be higher than the size of the image.
 2. The SB file should include the erase command. The Kinetis Flash Tool does not perform the erase operation for the SB file.
-
1. Write the application image to the base address of the Kinetis device flash (Memory ID is internal) using the write-memory or receive-sb-file command.
 2. Reset the device with the reset command. This causes the device to disconnect from the host.

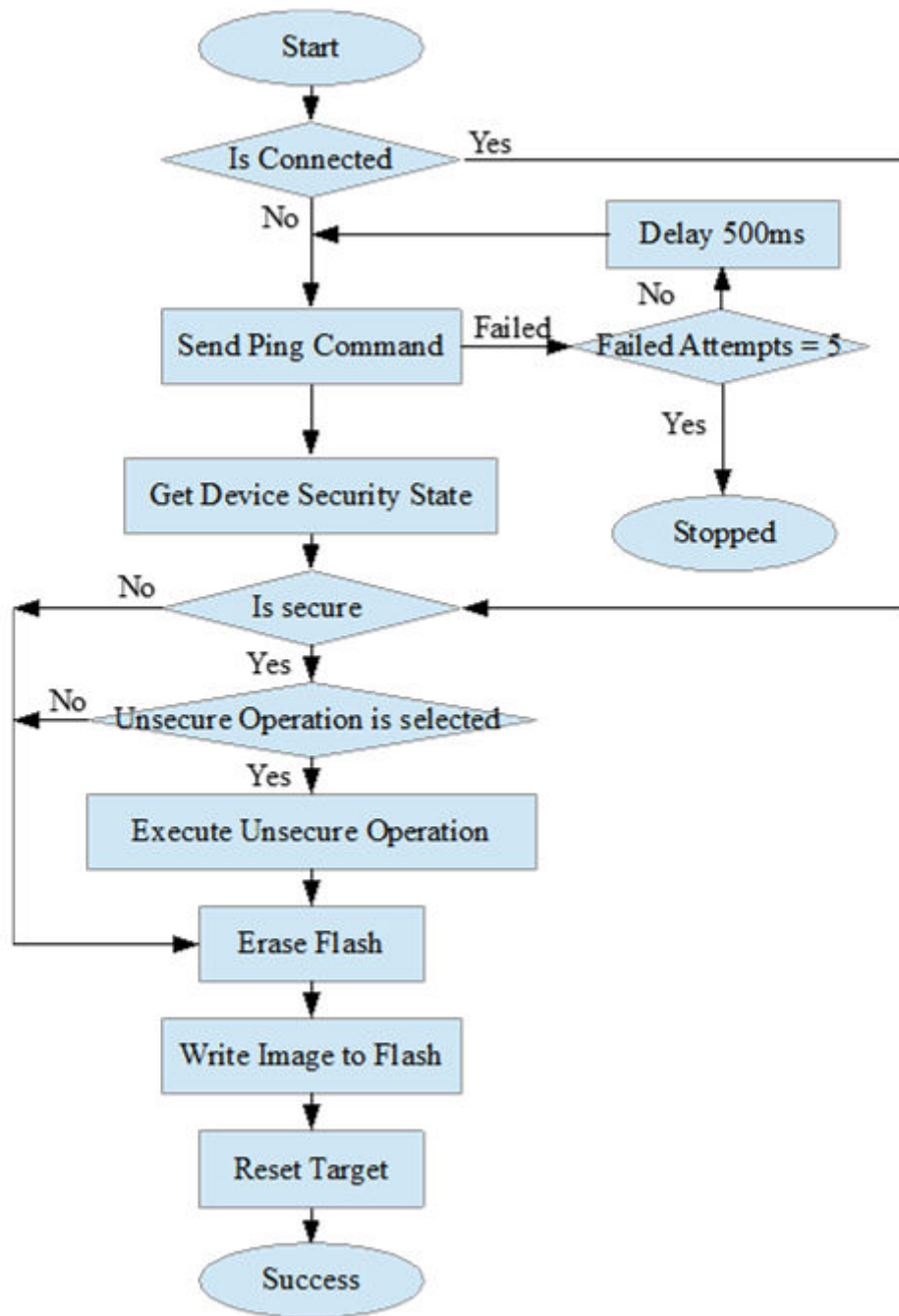


Figure 23. Update flowchart

4.2.7 Progress bar

Displays the progress of the current update process.

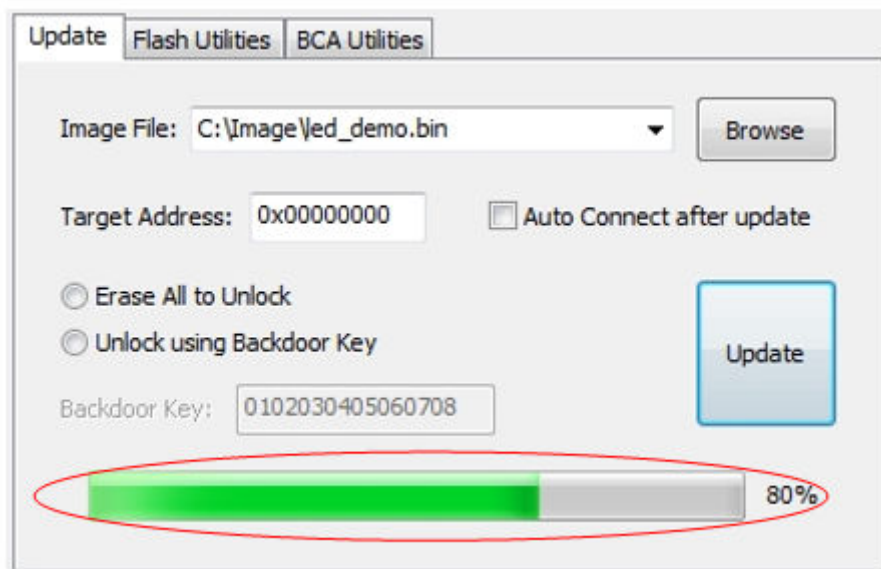


Figure 24. Progress bar

4.3 Flash utilities tab page

The flash utilities tab page is used to launch the erase flash process and program the IFR process.

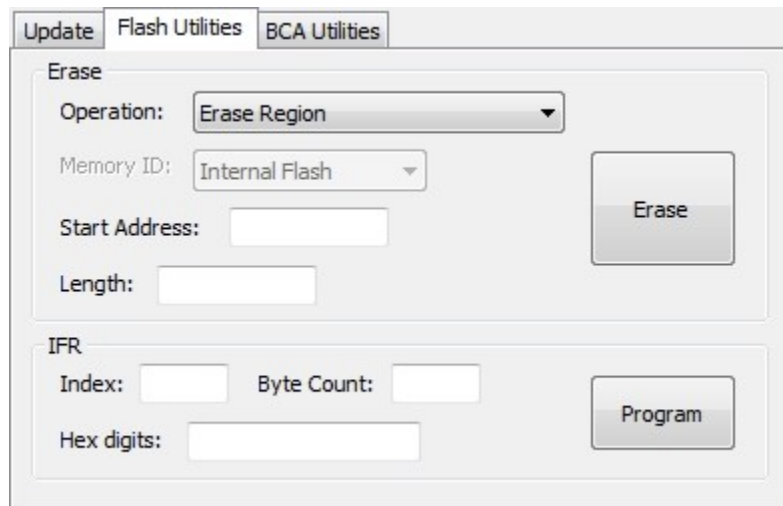


Figure 25. Flash utilities tab page

4.3.1 Erase

The erase group contains the controls used to erase the internal or external non-volatile memory regions.

4.3.1.1 Erase operation combo box

Contains three choices for use: "Erase Region", "Erase All", and "Erase All and Unsecure".

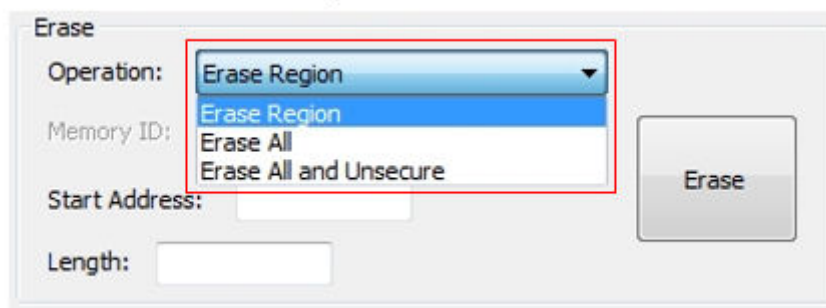


Figure 26. Erase operation combo box

The erase-region operation performs an erase of one or more sectors of the flash memory or a specified range of flash within the connected QuadSPI flash devices. The Start Address and Length are the two parameters required for this operation.

The erase-all operation performs an erase of the entire internal flash memory or of the QuadSPI memory. The memory ID is required for this operation.

The erase-all-and-unsecure operation performs a mass erase of the internal flash memory, including protected sectors. Only the internal flash memory is saved.

NOTE

Erase-all-unsecure is only supported on the internal flash, whereas erase-all is supported on both the internal flash and the QuadSPI flash.

4.3.1.2 Memory ID combo box

Contains two choices: "Internal Flash" and "QuadSPI 0 Memory". This combo box is enabled only when the erase-all operation is selected.

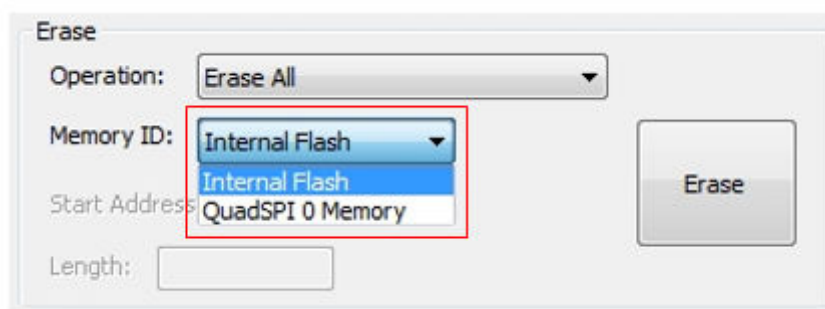


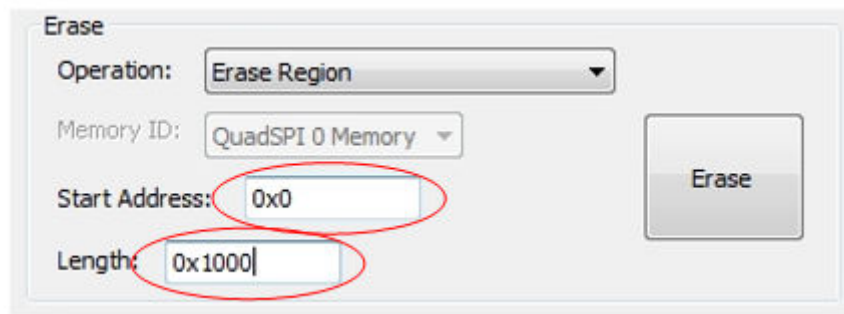
Figure 27. Memory ID

NOTE

The QuadSPI memory must be configured before erasing.

4.3.1.3 Start address textbox

Contains the start address of a specified flash region to erase. This textbox is enabled only when the erase region operation is selected.



The screenshot shows a software interface titled "Erase". It contains several controls: a dropdown menu for "Operation" set to "Erase Region", a dropdown for "Memory ID" set to "QuadSPI 0 Memory", a text box for "Start Address" containing "0x0", and a text box for "Length" containing "0x1000". A button labeled "Erase" is positioned to the right. Red circles are drawn around the "Start Address" and "Length" text boxes.

Figure 28. Start address and length

NOTE

The start address must be a valid address in the internal flash memory.

4.3.1.4 Length textbox

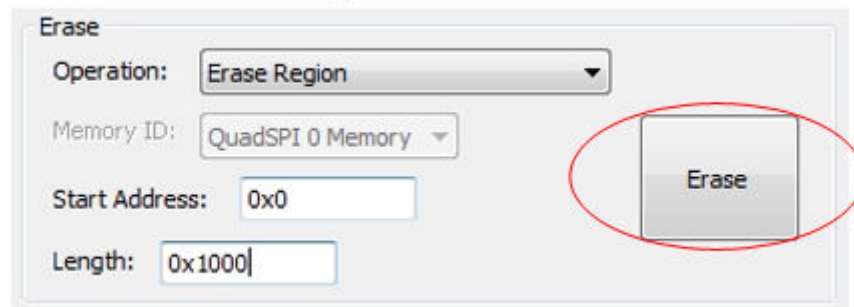
Contains the length (in bytes) of a specified flash region to erase. This textbox is enabled only when the erase region operation is selected.

NOTE

The erase region is forced to the next flash sector boundary.

4.3.1.5 Erase button

Click the "Erase" button to start the erase process.



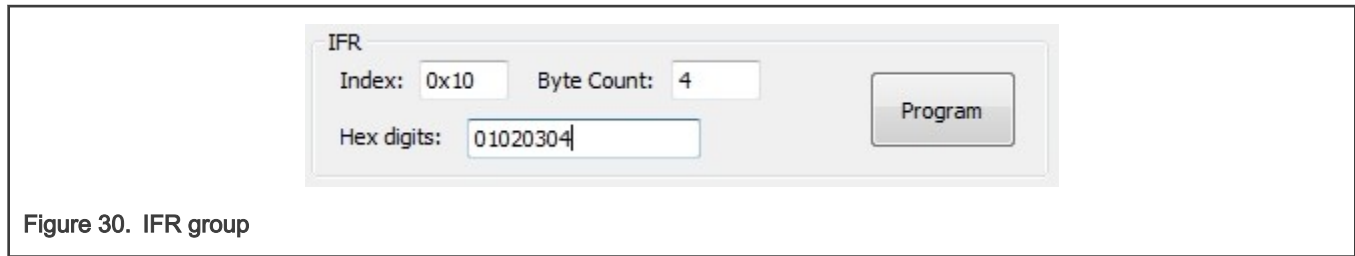
This screenshot shows the same "Erase" dialog box as Figure 28. In this view, the "Erase" button is highlighted with a red circle, while the other fields remain the same.

Figure 29. Erase button

Press the connect button to establish the communication first or the erase operation is terminated.

4.3.2 IFR

The IFR group contains the controls used to program the IFR.



4.3.2.1 Index textbox

The index textbox contains the index of the IFR field. See the flash memory module chapter of the device reference manual to receive the available IFR index.

NOTE

1. There is no erase capability for some IFR indexes.
2. For an erasable IFR, this field can be erased by the erase-all and the erase-all-and-unsecure commands.

4.3.2.2 Byte count textbox

Contains the size (in bytes) of each IFR index. This parameter must be 4-byte aligned for non-FAC fields or 8-byte aligned for FAC fields. You may also find this information in the device reference manual.

4.3.2.3 Digits textbox

Contains 8 or 16 hex digits to write to the IFR region. The digits in the textbox must be 8 or 16 hex digits long with no leading "0x".

4.3.2.4 Program button

Click the "Program" button to start the programming IFR process. Press the correct button to establish the communication first, or the programming operation is terminated.

NOTE

The Kinetis Flash Tool does not execute the erase operation during the programming process. Any attempt to reprogram the IFR field gets an error response. Therefore, execute the mass-erase command first.

4.4 BCA utilities tab page

The BCA utilities tab page offers several BCA-related features. Only the binary file format is supported.

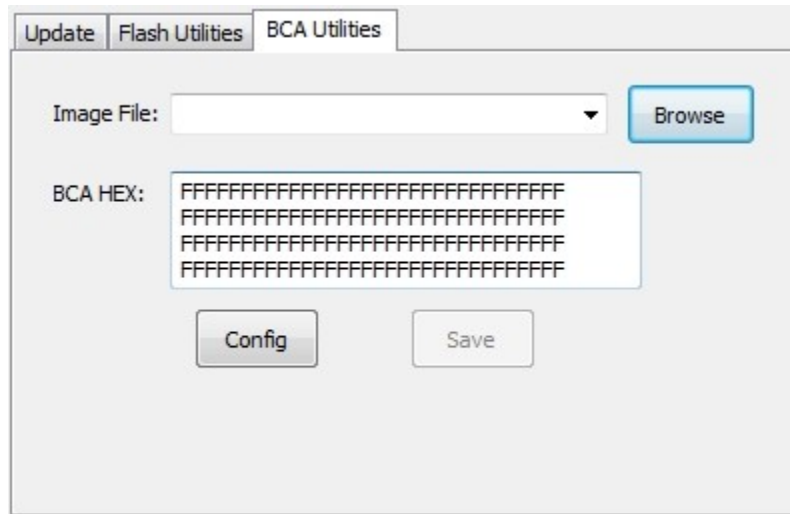


Figure 31. BCA utility tab page

4.4.1 Image file combo box

See Section 4.2.1, "Image file combo box".

4.4.2 Browse button

See Section 4.2.2, "Browse button".

4.4.3 BCA hex textbox

The BCA hex textbox contains the BCA hex digits. The digits are separated into four lines with a Windows OS-style ending. If a binary file is selected, the binaries at the BCA region are read and displayed in the BCA hex textbox. You may modify the binaries directly and click the "Save" button to write the new BCA digits back to the binary file.

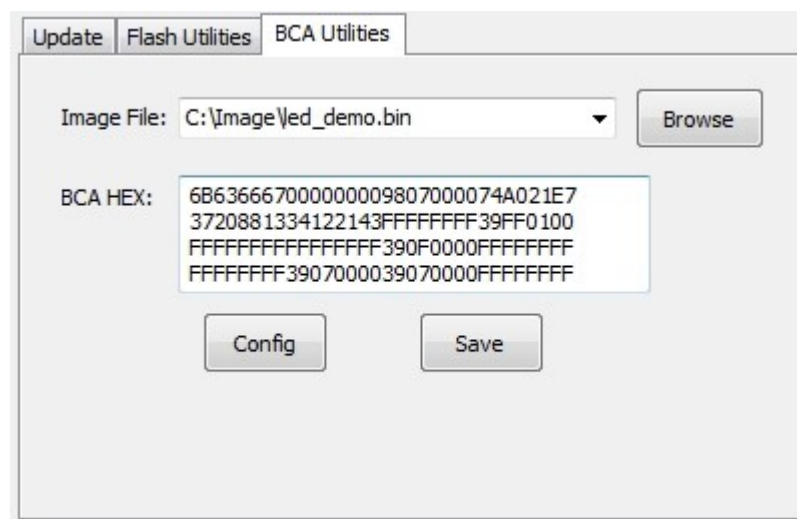


Figure 32. BCA HEX (binary is selected)

If no file or a different type of file is selected, the textbox is filled with all 0xFFs.

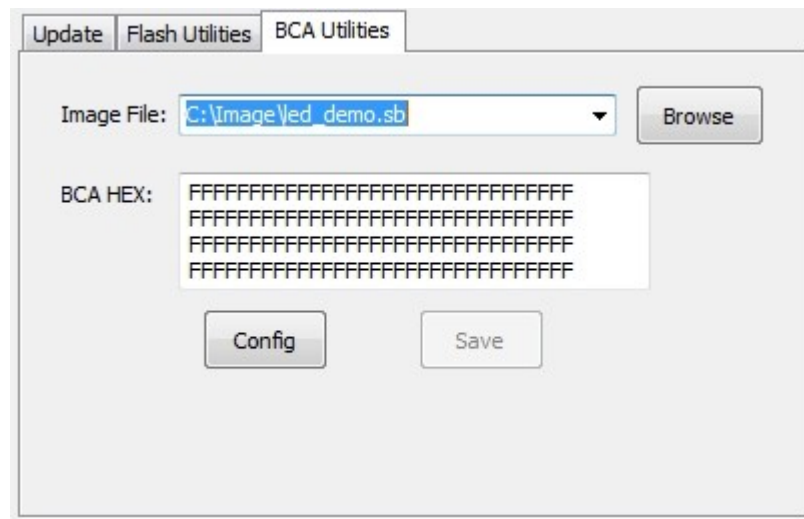


Figure 33. BCA HEX (file of other types is selected)

NOTE

1. If the binary file is smaller than 1 KB, the Kinetis Flash Tool appends 0xFFs at the end of the file to extend it to 1 KB.
2. Do not type illegal characters into the BCA hex textbox. If an illegal character is detected, this byte (two characters) is ignored.
3. Use a Windows OS-style ending. Do not use a UNIX-style ending.

4.4.4 Config button

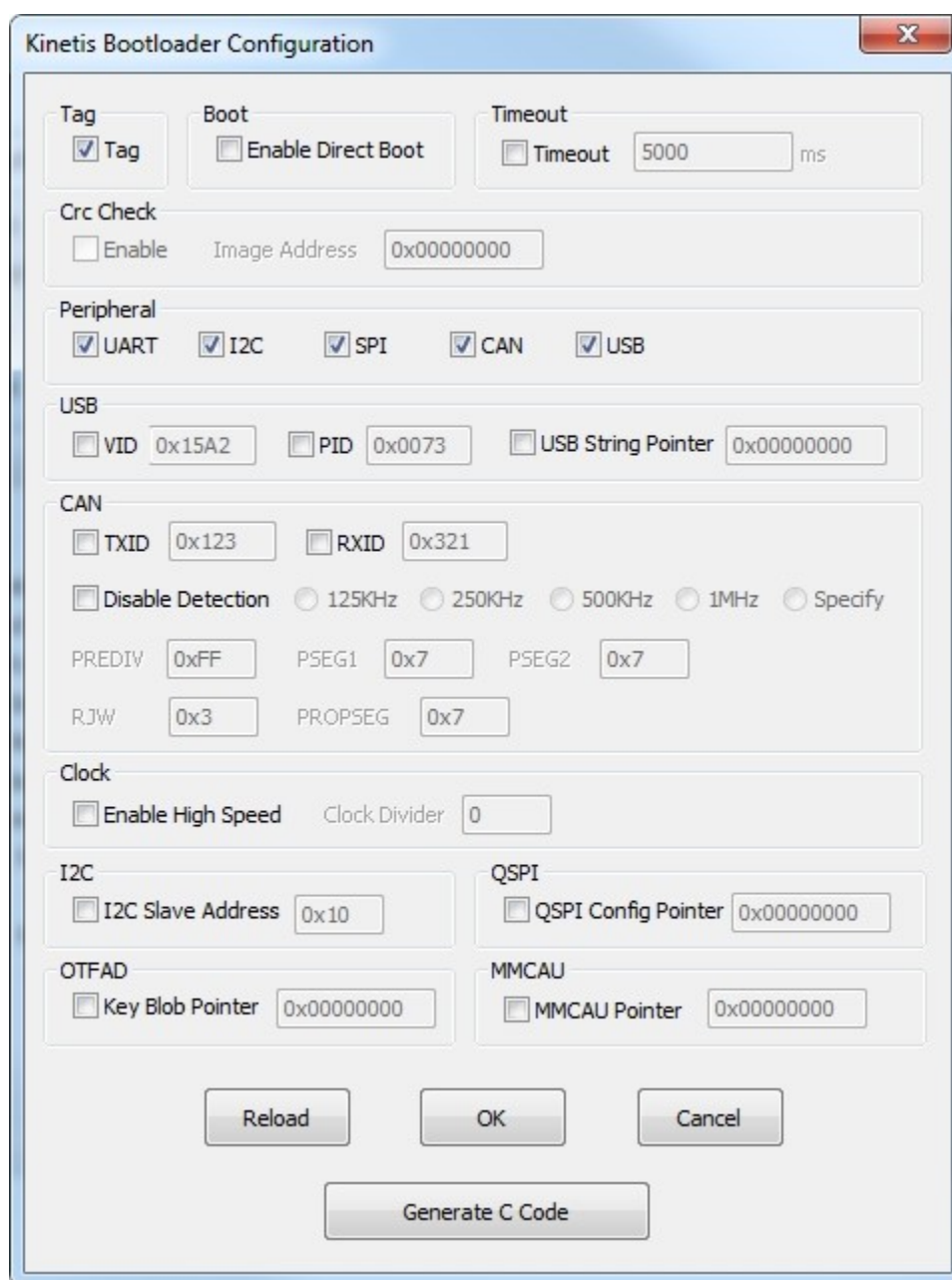
Click the "Config" button to open the UI configuration page. The binaries in the BCA hex textbox are loaded to the new UI page.

4.4.5 Save button

Click the "Save" button to write the digits in the BCA hex textbox to the binary image file. This button is enabled only when the selected file is a valid binary file.

4.5 Bootloader configuration page

The bootloader configuration page provides you the UI controls to generate BCA binaries or a BCA C structure. You may open this page by clicking the "Config" button in the BCA utilities tab page.



The image shows a 'Kinetis Bootloader Configuration' dialog box with the following sections and controls:

- Tag:** ☒ Tag
- Boot:** ☐ Enable Direct Boot
- Timeout:** ☐ Timeout 5000 ms
- Crc Check:** ☐ Enable. Image Address: 0x00000000
- Peripheral:** ☒ UART, ☒ I2C, ☒ SPI, ☒ CAN, ☒ USB
- USB:** ☐ VID 0x15A2, ☐ PID 0x0073, ☐ USB String Pointer 0x00000000
- CAN:** ☐ TXID 0x123, ☐ RXID 0x321. ☐ Disable Detection. Radio buttons: 125KHz, 250KHz, 500KHz, 1MHz, Specify. PREDIV: 0xFF, PSEG1: 0x7, PSEG2: 0x7. R.JW: 0x3, PROPSEG: 0x7.
- Clock:** ☐ Enable High Speed. Clock Divider: 0
- I2C:** ☐ I2C Slave Address 0x10
- QSPI:** ☐ QSPI Config Pointer 0x00000000
- OTFAD:** ☐ Key Blob Pointer 0x00000000
- MMCAU:** ☐ MMCAU Pointer 0x00000000
- Buttons:** Reload, OK, Cancel, Generate C Code

Figure 34. Bootloader configuration page

4.5.1 Check boxes

Check the boxes to enable the corresponding config and the associated textbox (if it has one). If a config is not enabled, the relevant bits in the BCA are filled with 1s when generating BCA binaries or a C structure.

NOTE

The CRC-enabled check box is disabled if no valid binary file is selected. The CRC calculation needs the final binary image.

4.5.2 Textboxes

The textboxes contain the digits of the corresponding config. The textbox is enabled only when the associated check box is checked.

NOTE

1. Do not enter overflowing digits, otherwise the overflowed bits are ignored in the generating process.
2. The value of the image address at the CRC check group is the start address where the image is going to be placed to.

4.5.3 Reload button

Click the "Reload" button to abandon the changes and reset the controls to the state that the bootloader configuration page just opened.

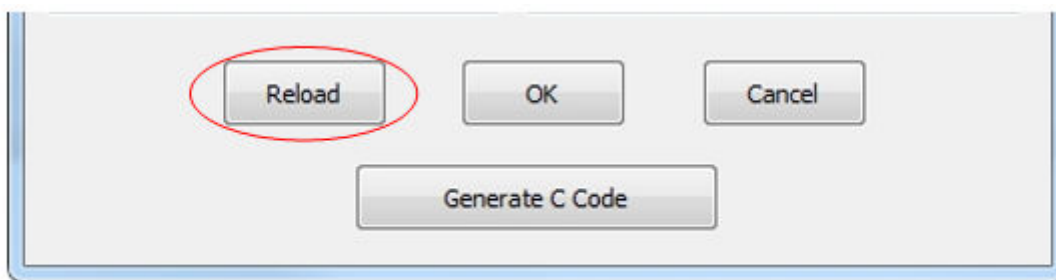


Figure 35. Reload button

4.5.4 OK button

Click the "OK" button to generate new BCA binaries according to the changes you made. The new binaries are updated to the BCA hex textbox.

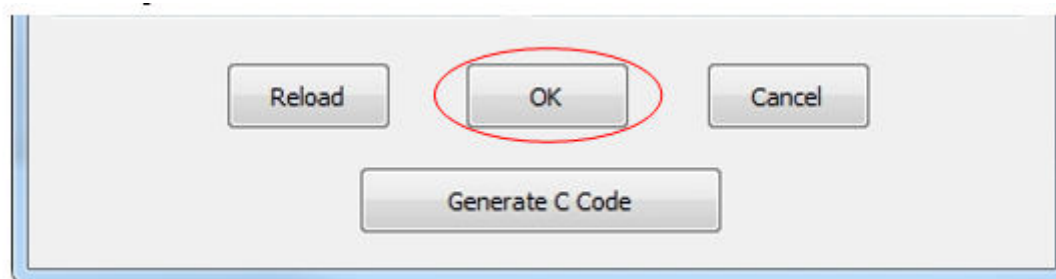


Figure 36. Reload button

NOTE

The new BCA binaries are not saved to the image file by clicking the "OK" button. Instead, click the "Save" button in the BCA utilities tab page to execute the save progress.

4.5.5 Cancel button

Click the 'Cancel' button to abandon the changes that you made and go back to the BCA utilities tab page.

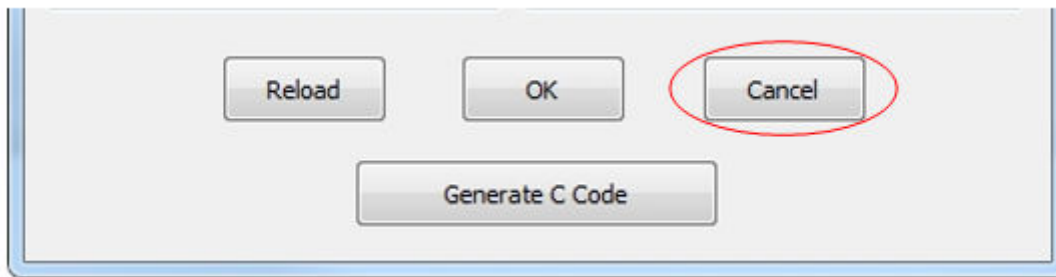


Figure 37. Cancel button

4.5.6 Generate C code button

Click the "Generate C code" button to begin the process of generating the C code files.

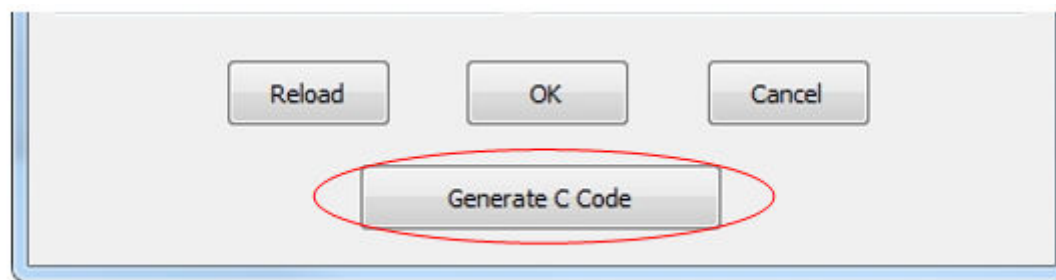


Figure 38. Generate C code button

This file contains the definition of the MCU bootloader configuration structure and a variable filled with configuration data. You may use this code for your application projects. For step-by-step instructions and usage, see the "Integrate config file to user project" section.

NOTE

Three parameters of CRC are filled with all 0xFFs when generating, because the CRC calculation needs the final binary image.

5 Typical usage

This section describes the step-by-step usage of the Kinetis Flash Tool.

5.1 Quick update

1. Launch the Kinetis Flash Tool.

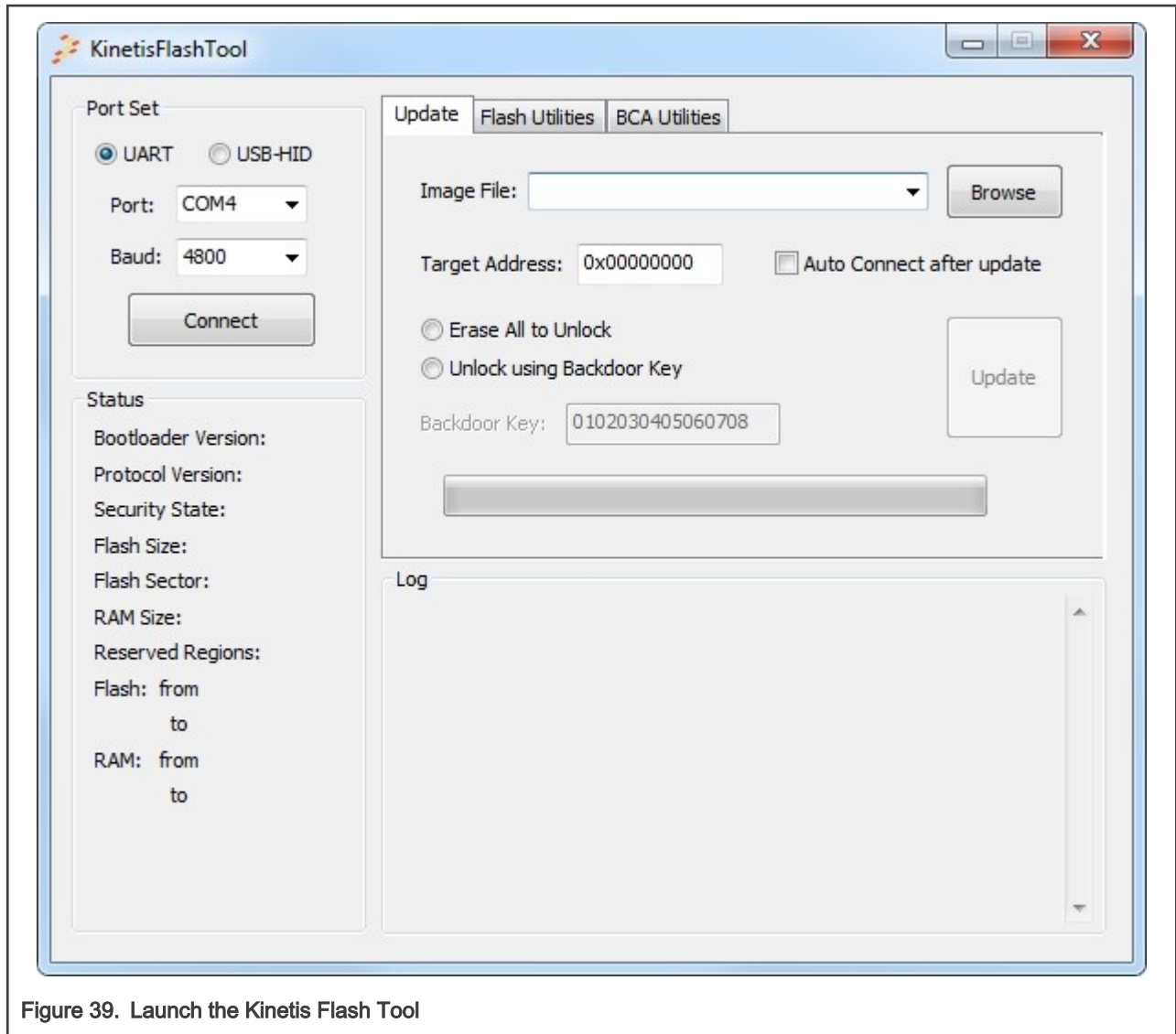


Figure 39. Launch the Kinetis Flash Tool

2. Select the peripheral. Currently, only UART and USB-HID are supported.
3. Configure the peripheral. The COM port and the baud rate must be specified for the UART, while the VID and PID must be configured for the USB-HID.
4. Select the image file. If the image file is not in the combo box list, type the full path file name into the textbox or use the "Browse" button to locate the image file.
5. If the selected image file is binary, the target address must be specified.
6. Select the unlock operation. If the device is not in a secure state, no unlock operation is executed. Select an operation if you are not sure whether the device is secure.
7. Click the "Update" button to start the process.
8. After a few seconds, the application runs after reaching the timeout.

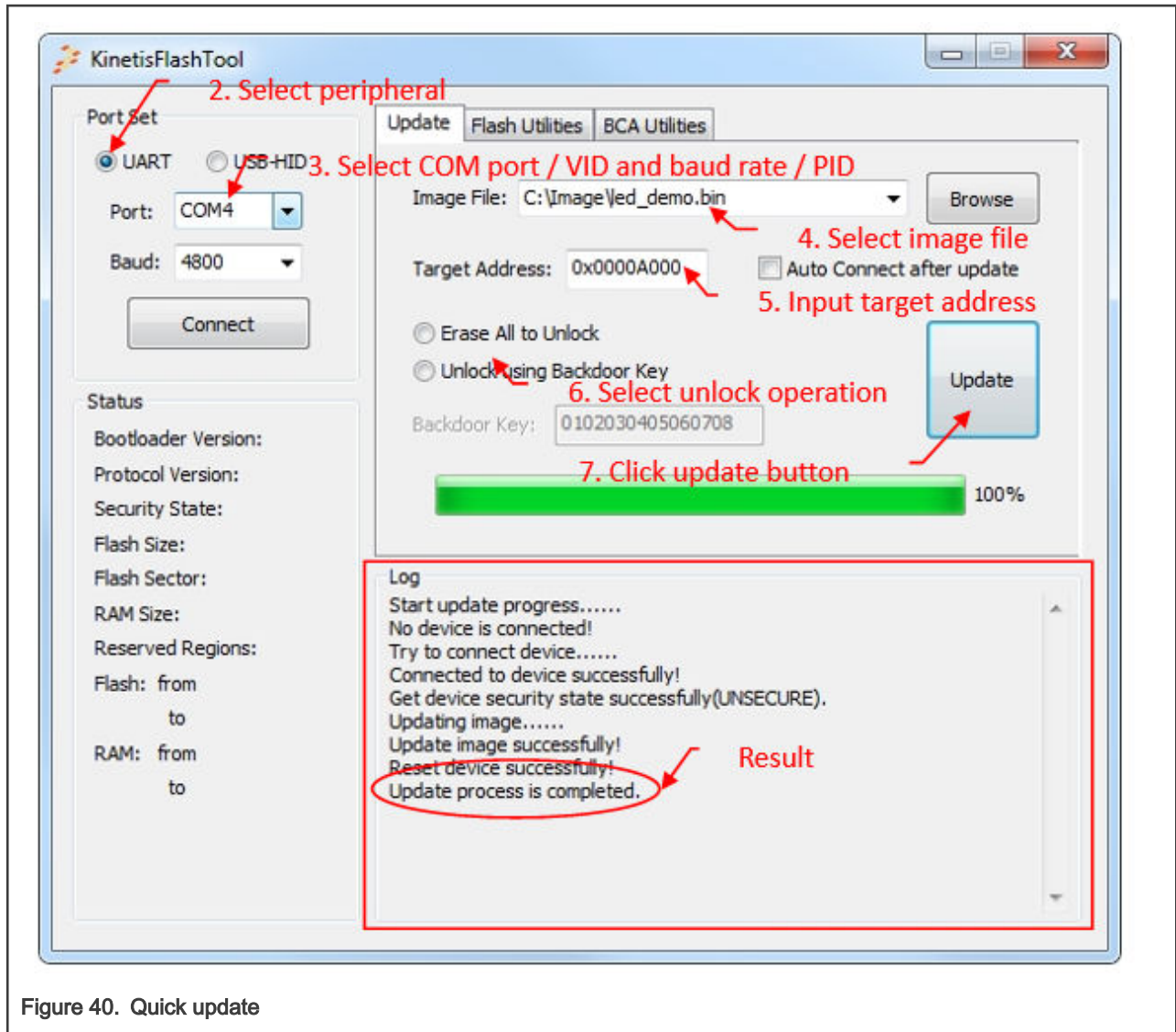


Figure 40. Quick update

5.2 Normal update

1. Follow steps 1 to 3 of the quick update.
2. Click the "Connect" button to connect to the target device.

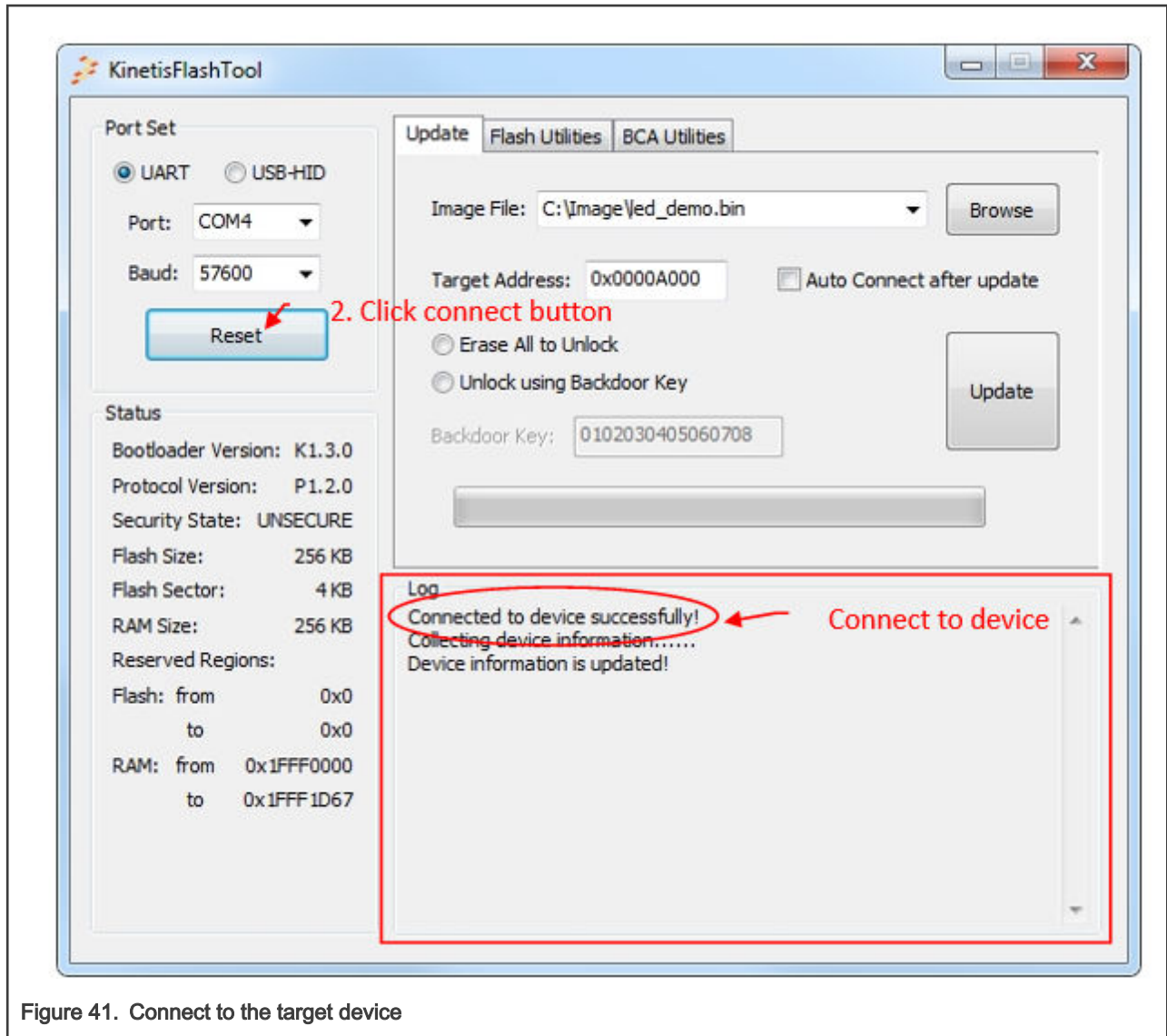


Figure 41. Connect to the target device

3. Select the image file. If the image file is not in the combo box list, type the full path name into the textbox of this control, or use the "Browse" button to specify the image file.
4. If the selected image file is binary, the Memory ID and target address must be specified.
5. Select the unlock operation. You may get the security information from the status region.
6. Click the "Update" button to start the process.
7. After a few seconds, the application runs after reaching the timeout.

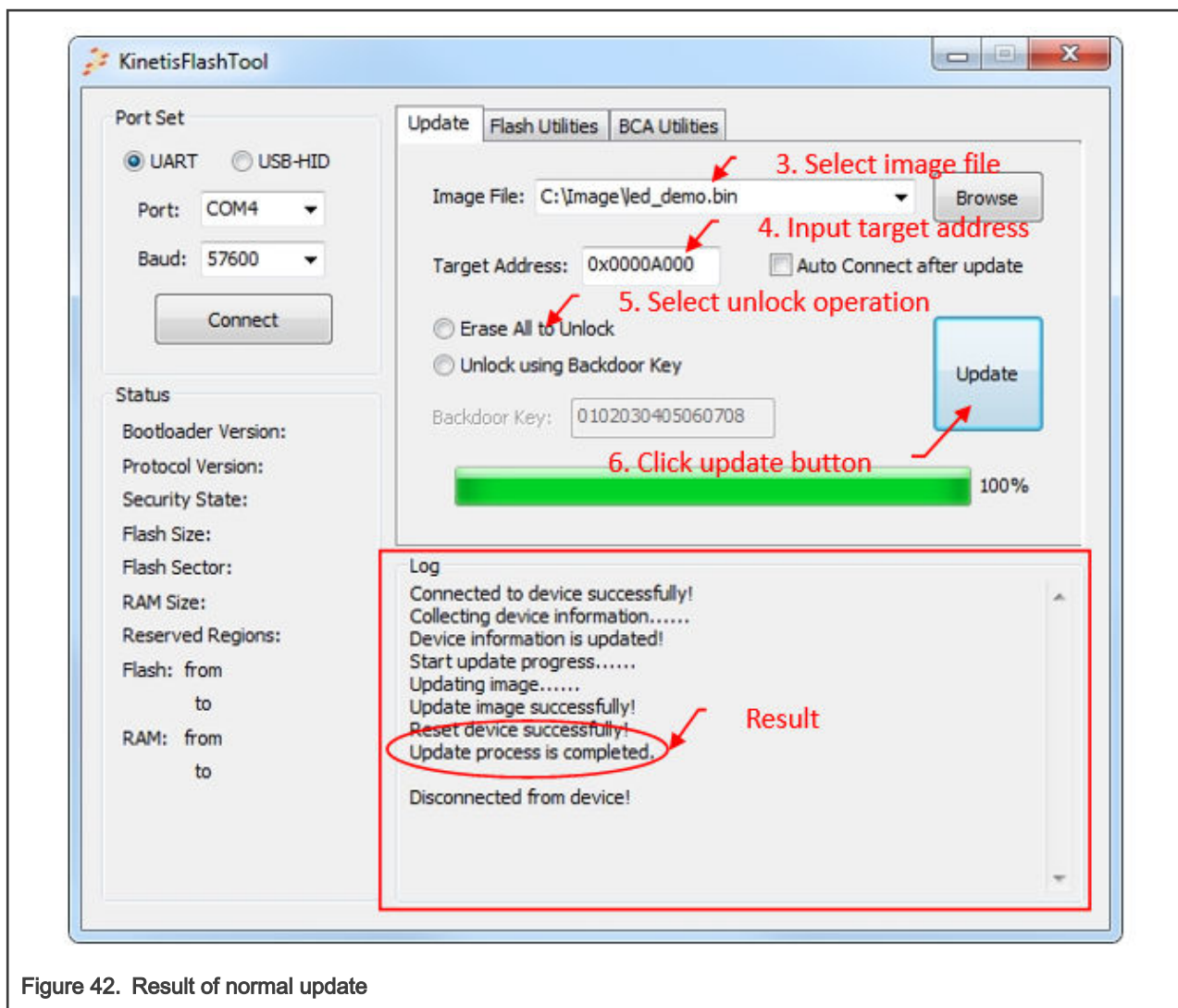


Figure 42. Result of normal update

5.3 Continuous update

An extra step is needed before clicking the "Update" button for a quick update or a normal update. This is an example of a normal update:

1. Follow the steps 1 to 5 of the normal update.
2. Select the "Auto Connect after update" check box.
3. Click the "Update" button to start the process.
4. The Kinetis Flash Tool tries to establish the communication when the update process is finished.
5. You may change the device or the board after the update.
6. When the communication is re-established, you may perform the second update.
7. Click the "Update" button for the second update.

NOTE

The port set must not be changed. Otherwise, the Kinetis Flash Tool stops the sent commands to try to establish communication until you press the connect button.

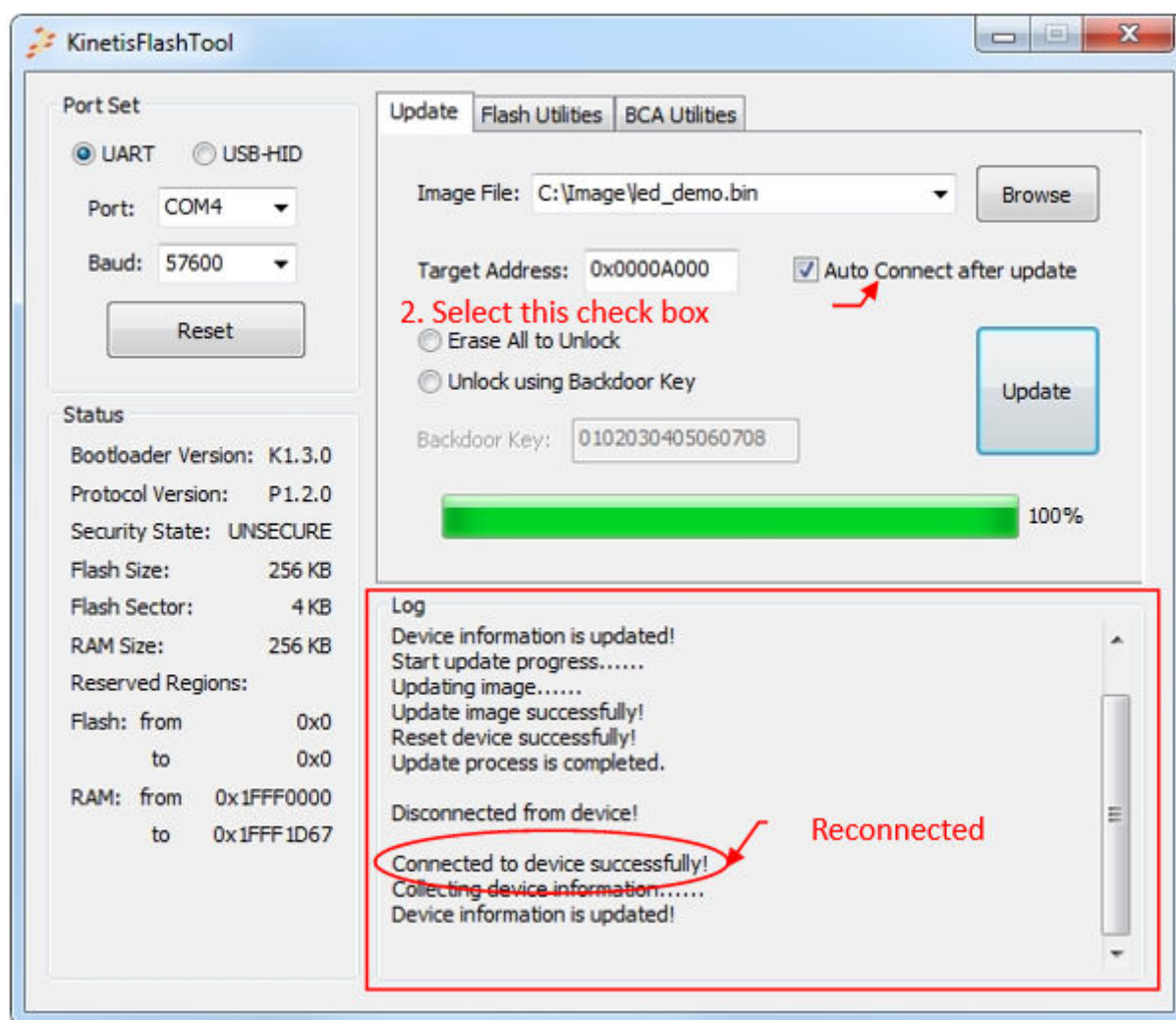


Figure 43. Continuous update

5.4 Flash erase

5.4.1 Flash erase region

1. Follow steps 1 and 2 in the normal update to connect the target device.
2. Select the flash utilities tab page.
3. Select the erase region operation.
4. Fill the start address and length.
5. Click the "Erase" button.

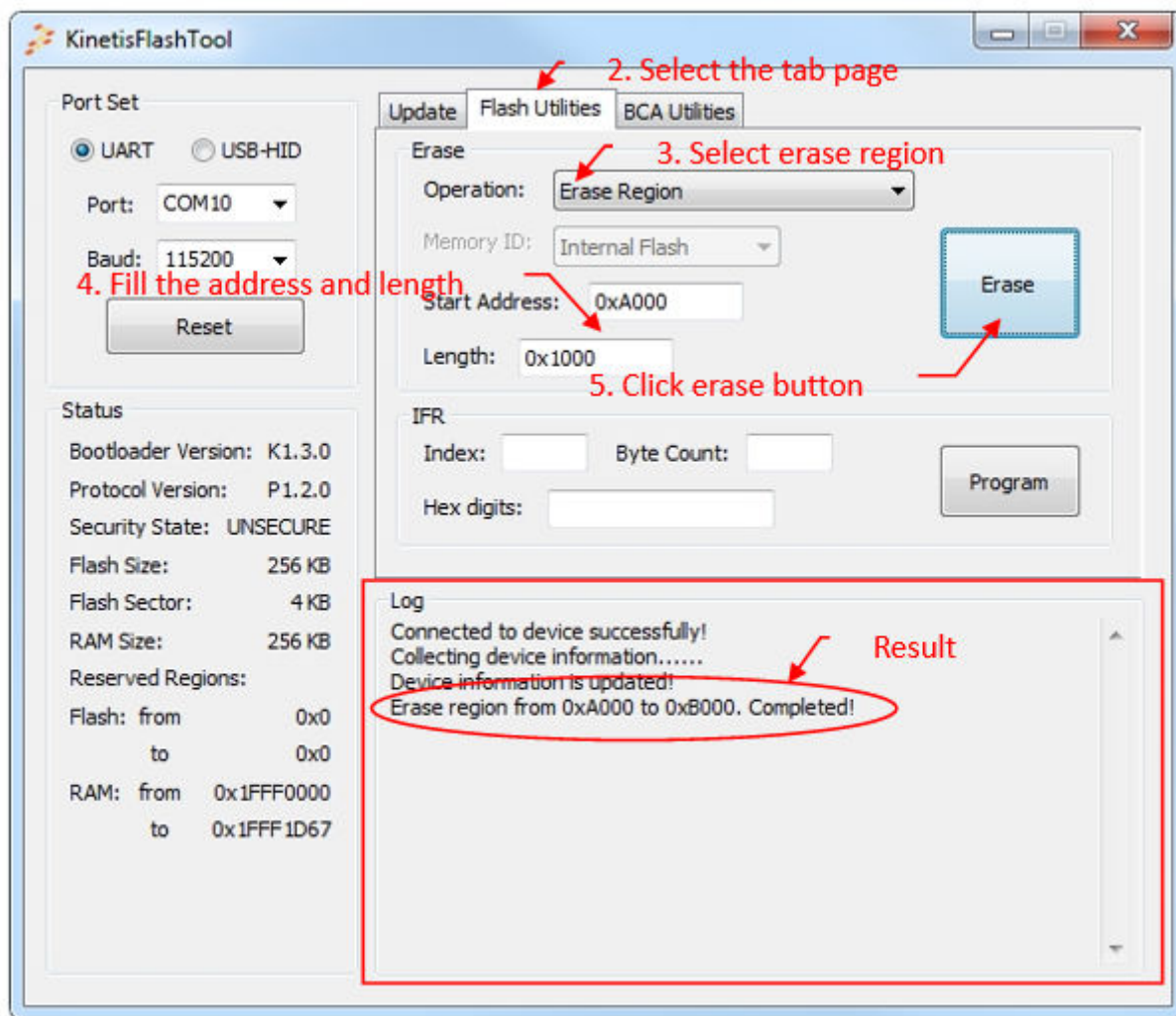


Figure 44. Flash erase region

5.4.2 Flash erase all

1. Follow steps 1 and 2 in the flash erase region.
2. Select the erase region operation.
3. Select the memory ID.
4. Click the "Erase" button.

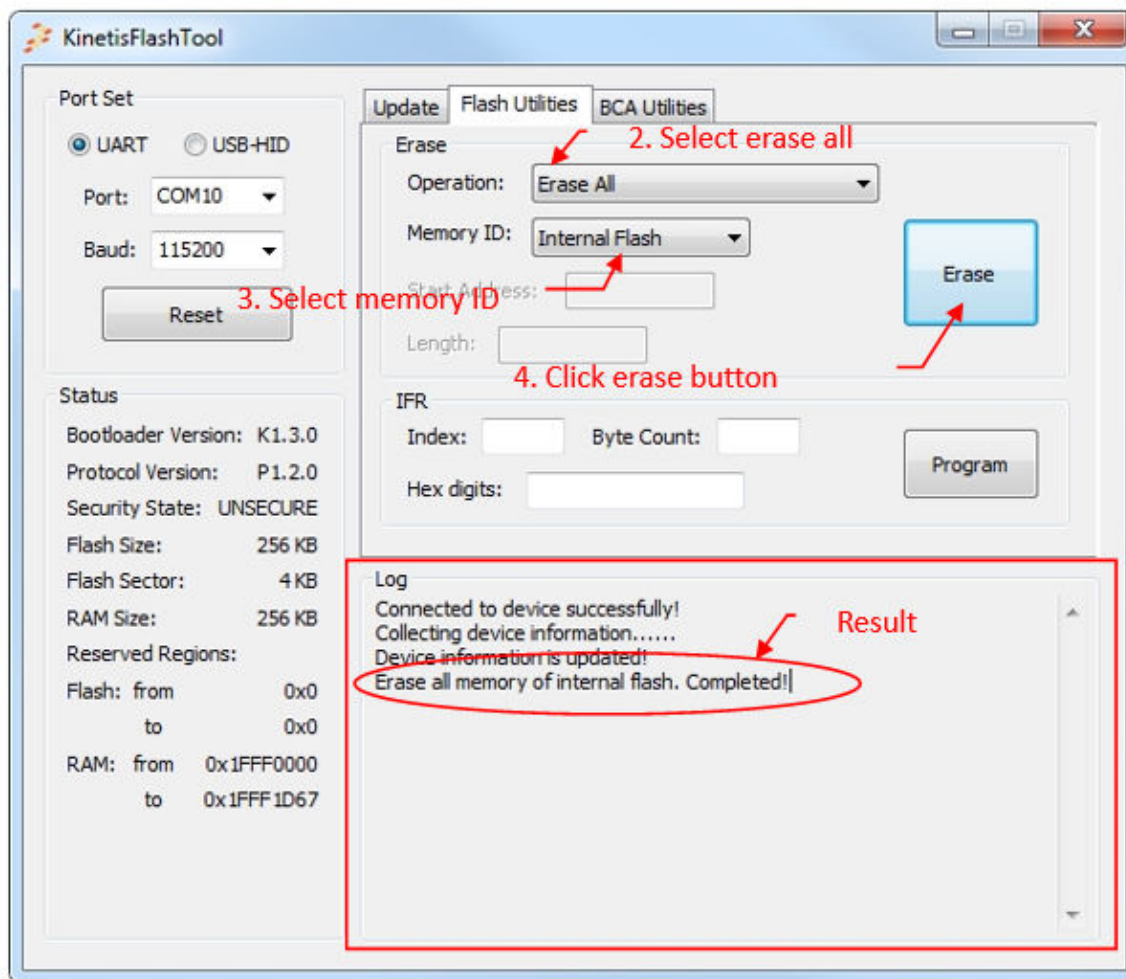


Figure 45. Flash erase all

5.4.3 Flash erase all and unsecure

1. Follow steps 1 to 2 in the flash erase region.
2. Select the erase region operation.
3. Click the "Erase" button.

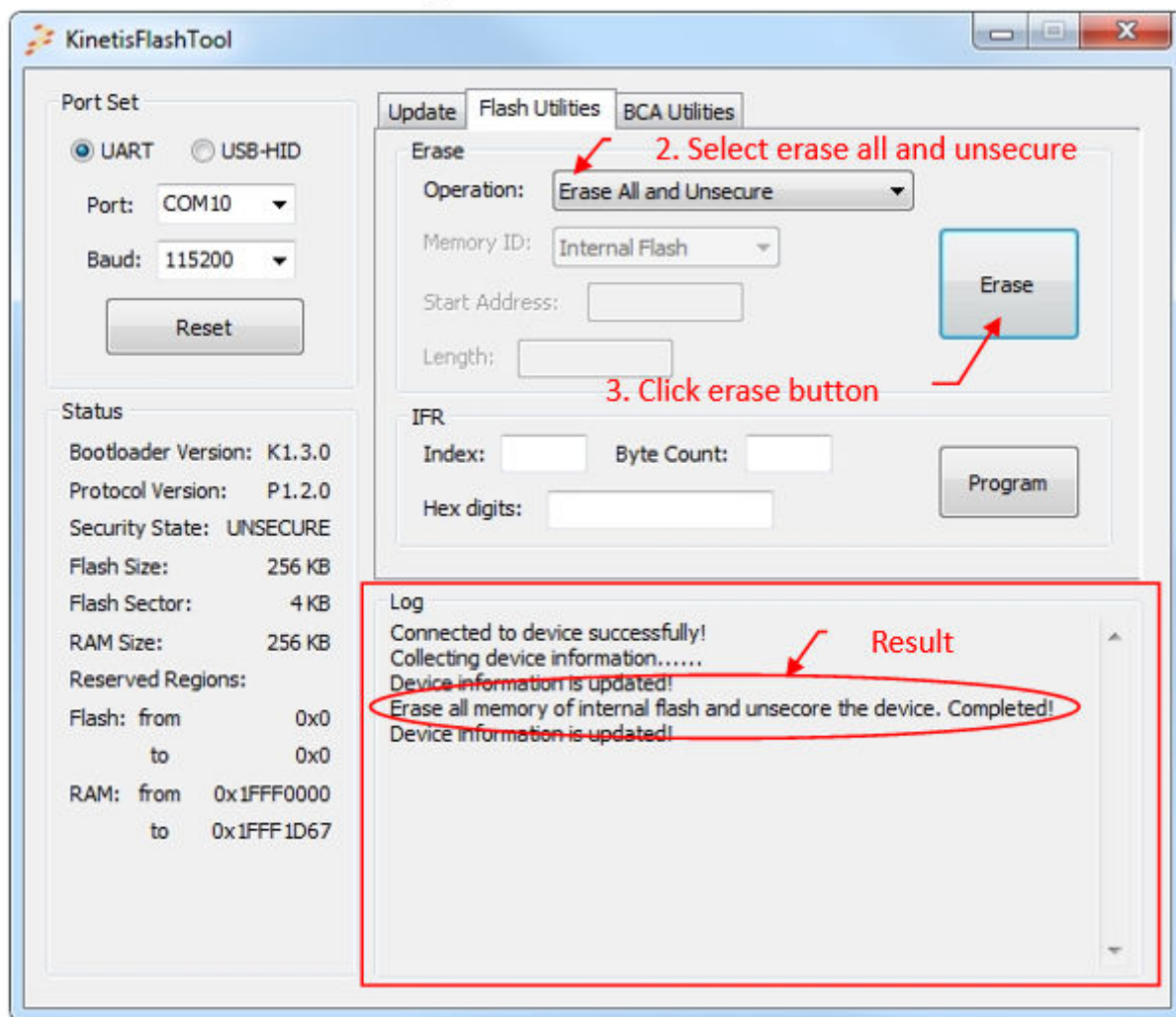


Figure 46. Flash erase all and unsecure

5.5 Program IFR

1. Follow steps 1 to 2 of the normal update to connect to the target device.
2. Select the "flash utilities" tab.
3. Fill the index and the byte count.
4. Fill the IFR digits.
5. Click the "Program" button.

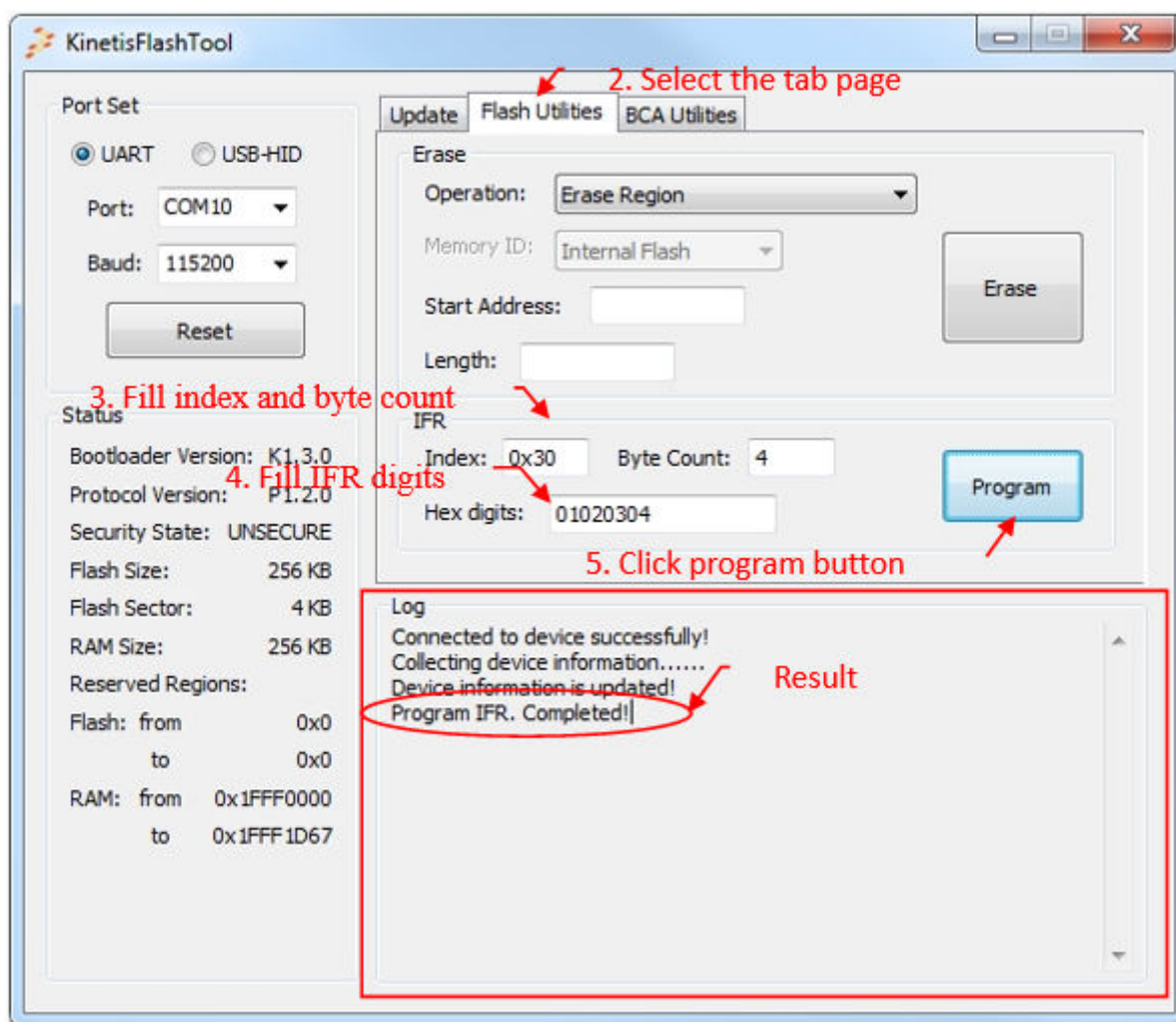
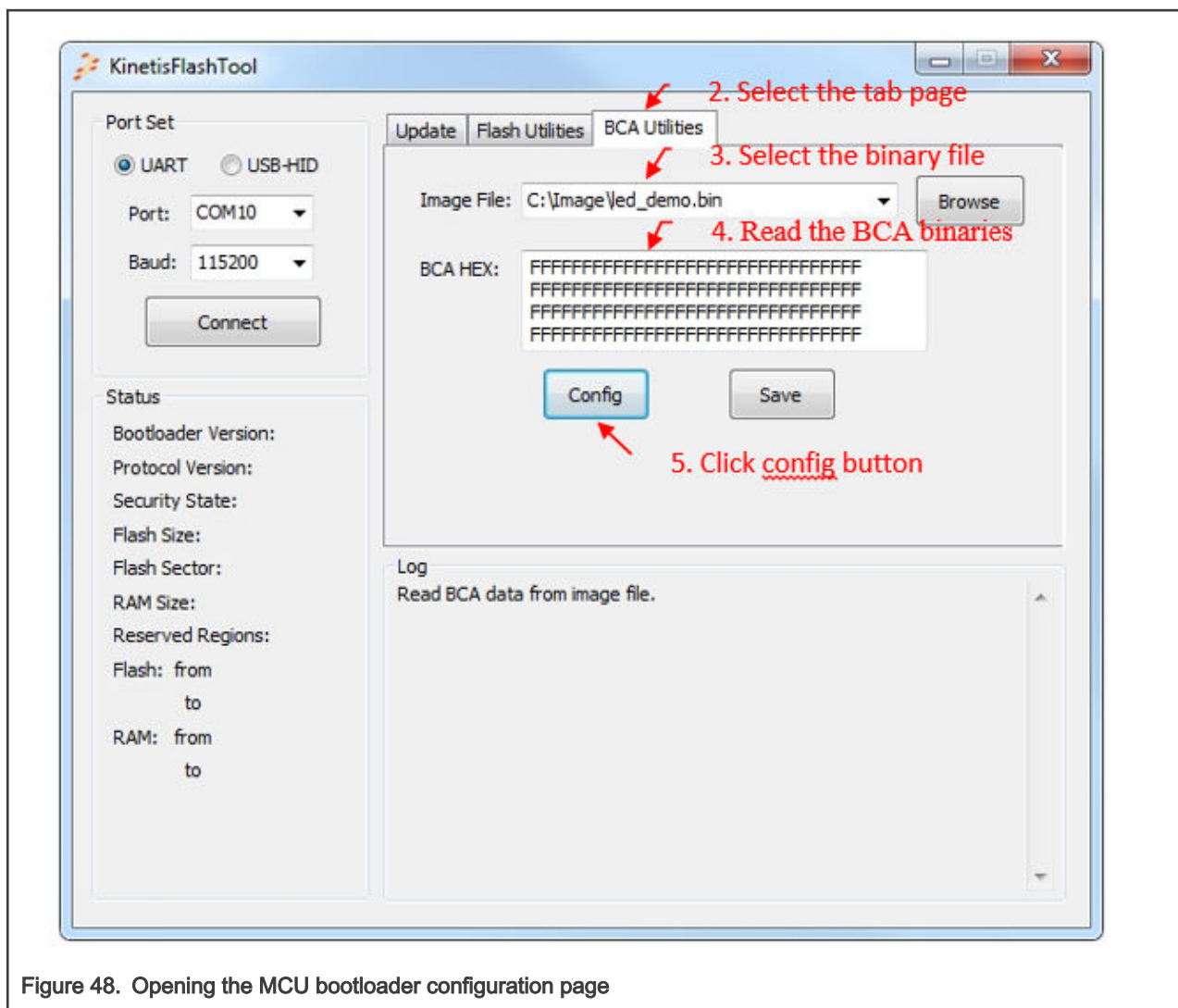


Figure 47. Program IFR

5.6 Configure BCA

1. Launch the Kinetis Flash Tool.
2. Select the BCA utilities tab page.
3. Select the binary image file. If the image file is not in the combo box list, type the full path file name into the textbox of this control, or use the "Browse" button to specify the image file.
4. When a binary file is selected, the BCA binaries are displayed in the BCA hex textbox.
5. Click the "Config" button, and the MCU bootloader configuration page opens.



6. Configure the BCA.
7. Press the "OK" button.

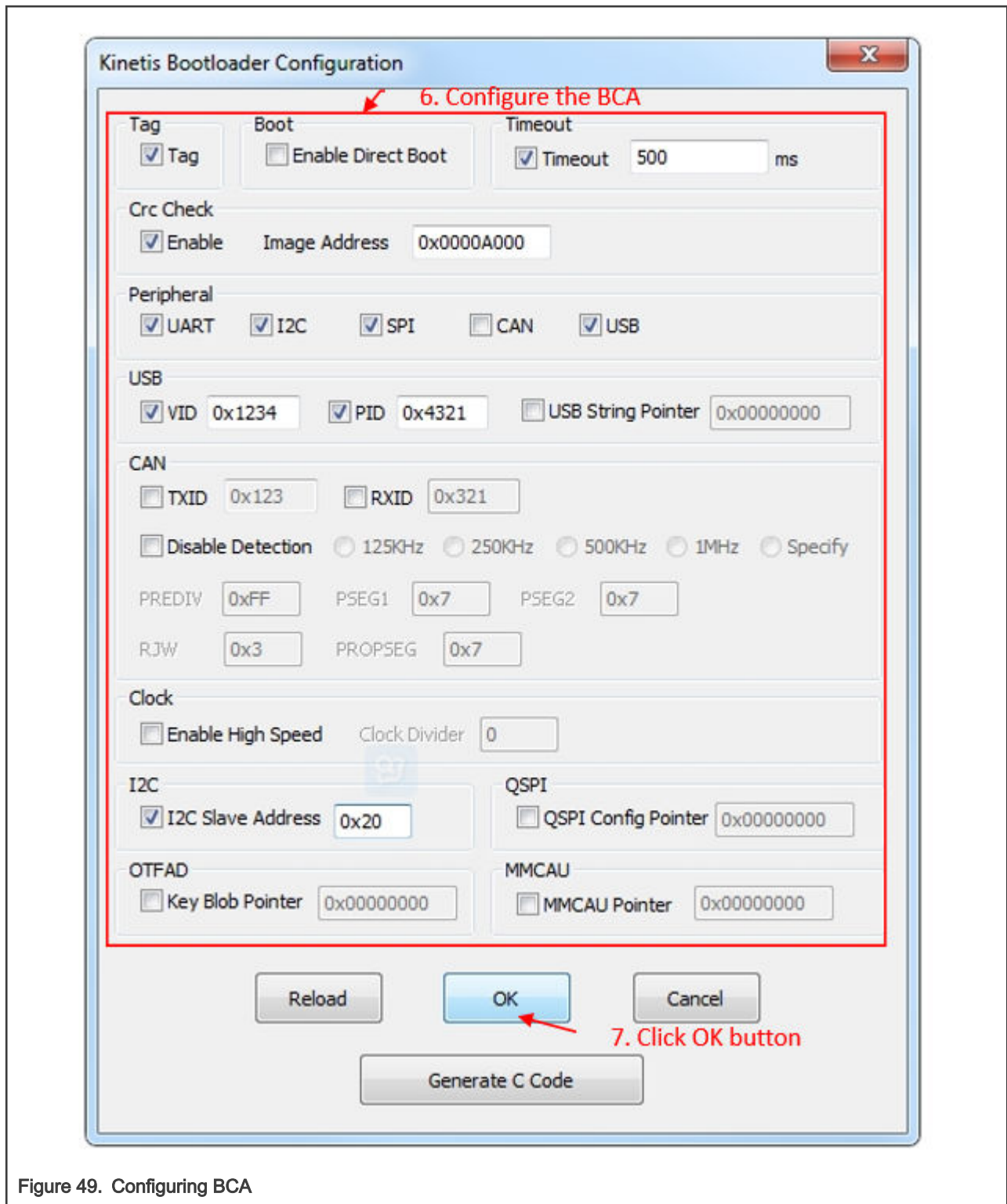


Figure 49. Configuring BCA

8. The new BCA binaries are updated in the textbox.
9. Click the "Save" button to save the new binaries to the binary file.

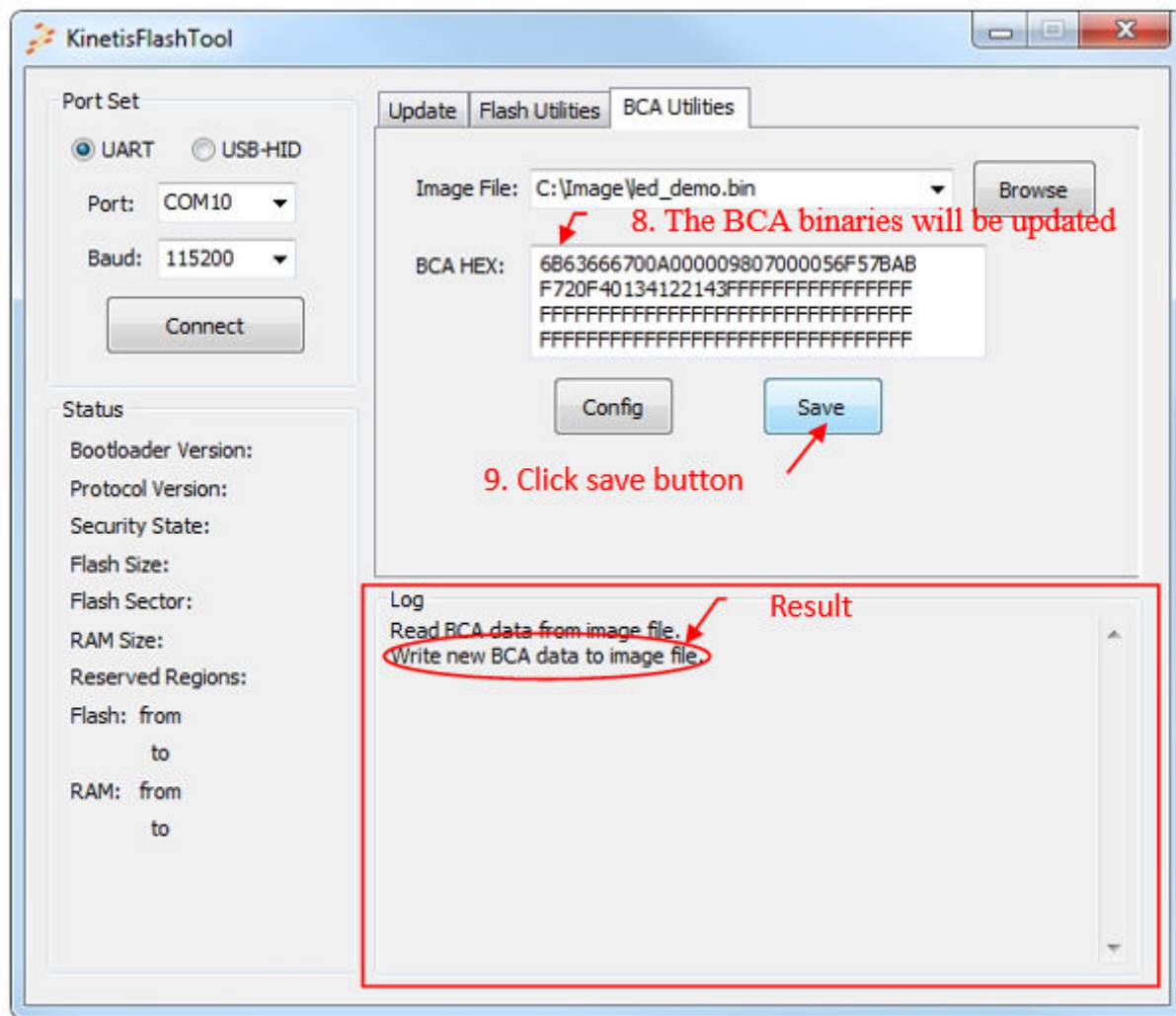


Figure 50. Saving the new configuration

5.7 Generate C config file

1. Follow steps 1 through 6 of the configure BCA section. Step 3 is optional. If no file is selected, the BCA hex textbox is filled with all 0xFFs.
2. Press the "generate C code" button.

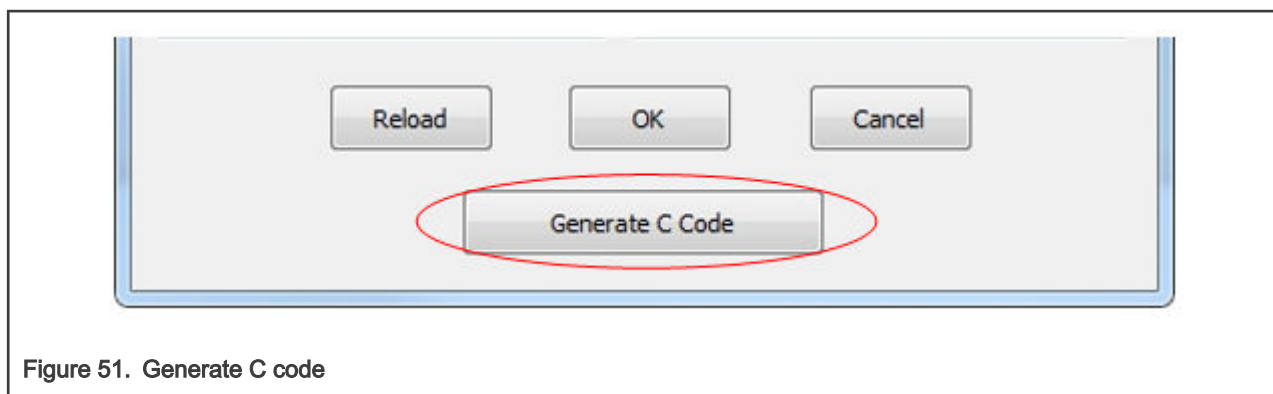


Figure 51. Generate C code

3. The "Save As" dialog opens. Specify the file path and name.
4. Click the "Save" button in the "Save As" dialog.

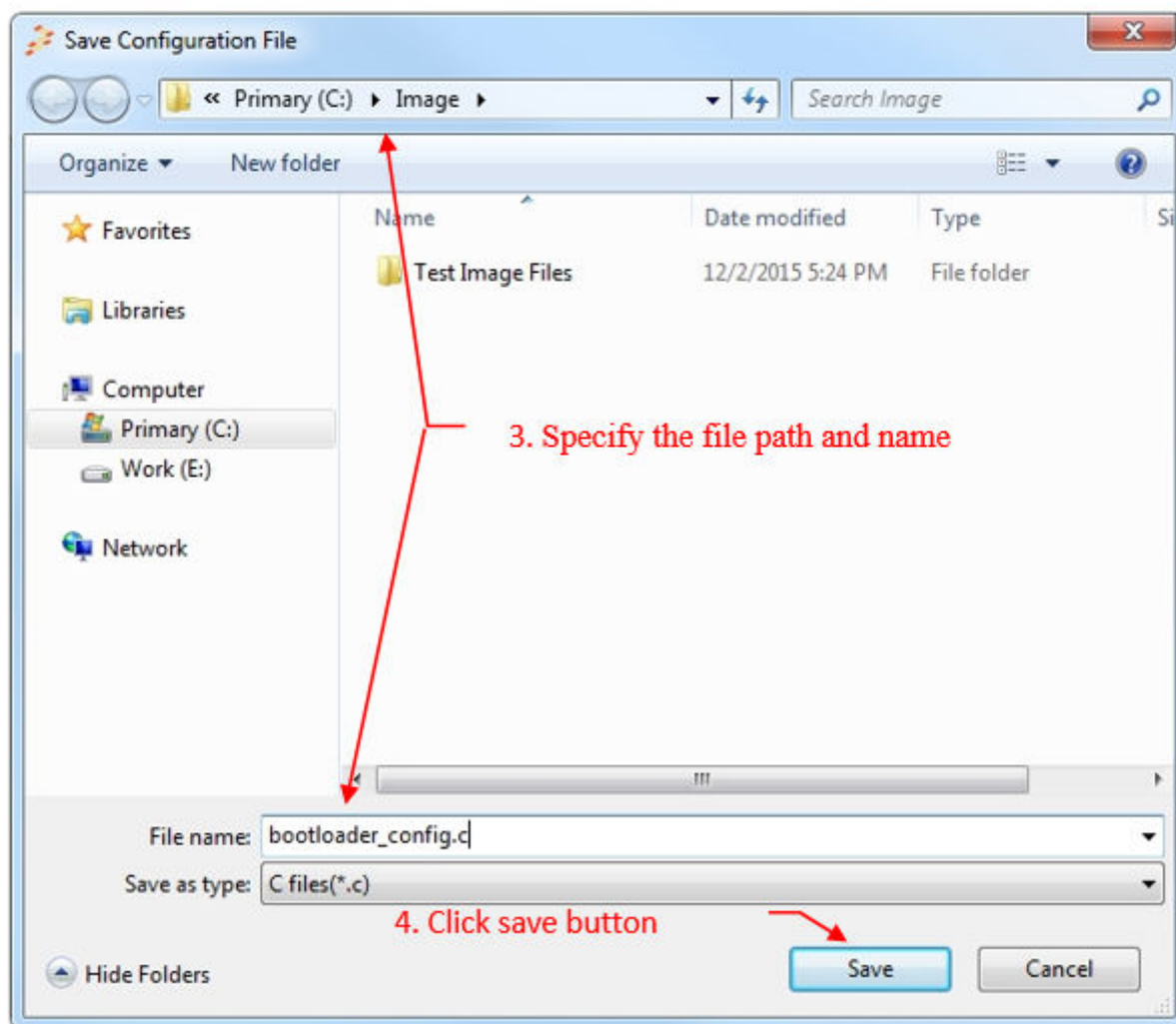


Figure 52. Save the C configuration file

5.8 Integrate config file to user project

Use the K64 LED demo IAR IDE project for example.

1. Add the *bootloader_config.c* file to the project.

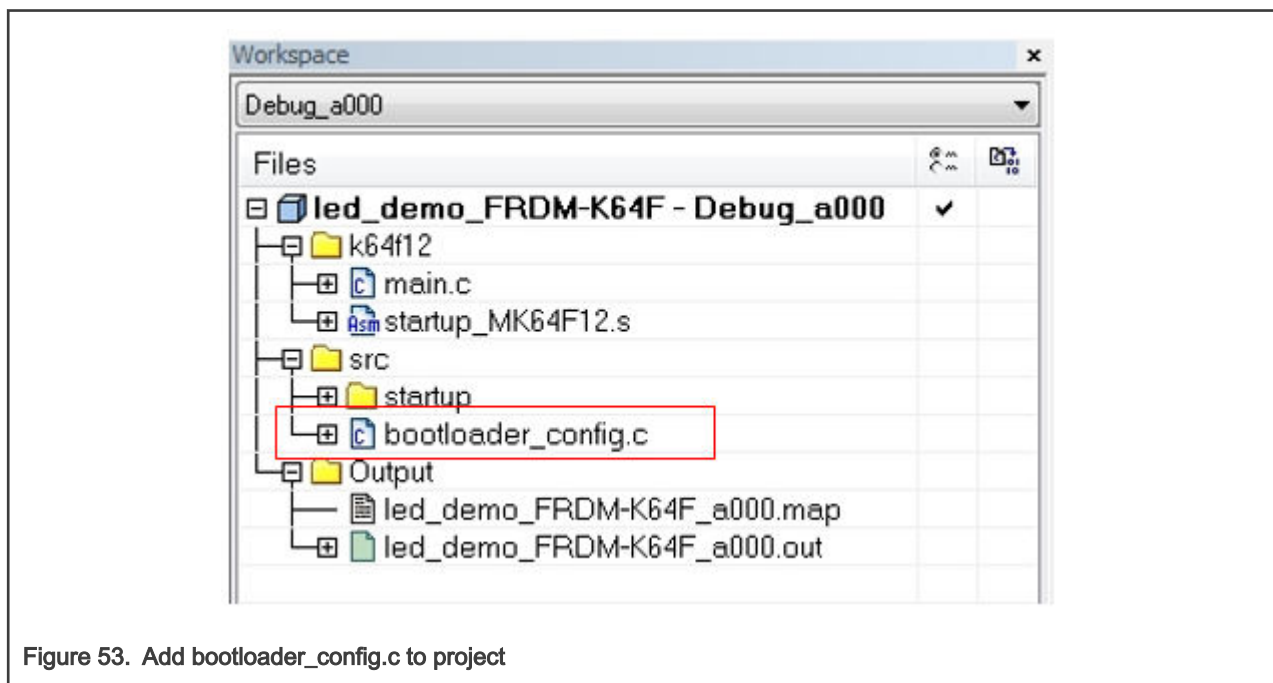


Figure 53. Add bootloader_config.c to project

2. Remove the `__bootloaderConfigurationArea` section in the *startup_MK64F12.s* file.

```

startup_MK64F12.s
#ifdef BL_HAS_BOOTLOADER_CONFIG

    __bootloaderConfigurationArea ; 0x3c0
        DCD    'kcfg'           ; [00:03] tag - Tag value used to validate the boot
        DCD    0xFFFFFFFF       ; [04:07] crcStartAddress
        DCD    0xFFFFFFFF       ; [08:0b] crcByteCount
        DCD    0xFFFFFFFF       ; [0c:0f] crcExpectedValue
        DCB    0xFF             ; [10:10] enabledPeripherals
        DCB    0xFF             ; [11:11] i2cSlaveAddress
        DCW    5000             ; [12:13] peripheralDetectionTimeoutMs - Timeout
        DCW    0xFFFF           ; [14:15] usbVid
        DCW    0xFFFF           ; [16:17] usbPid
        DCD    0xFFFFFFFF       ; [18:1b] usbStringsPointer
        DCB    0xFF             ; [1c:1c] clockFlags - High Speed and other clock
        DCB    0xFF             ; [1d:1d] clockDivider - One's complement of clock
        DCW    0xFFFF           ; [1e:1f] reserved

#ifdef BL_HAS_FLASH_CONFIG
    ; Fill to align with flash configuration field.
    REPT    (0x400-0x3e0)/4    ; 0x3e0 - 0x3ff
    DCD    0xFFFFFFFF
    ENDR
#endif

#else
#ifdef BL_HAS_FLASH_CONFIG
    ; Fill to align with flash configuration field.
    REPT    (0x400-0x3c0)/4    ; 0x3c0-0x400
    DCD    0xFFFFFFFF
    ENDR
#endif
#endif

```

Figure 54. Bootloader config in startup ASM file

3. Add the *BOOTLOADER_CONFIG* global macro.

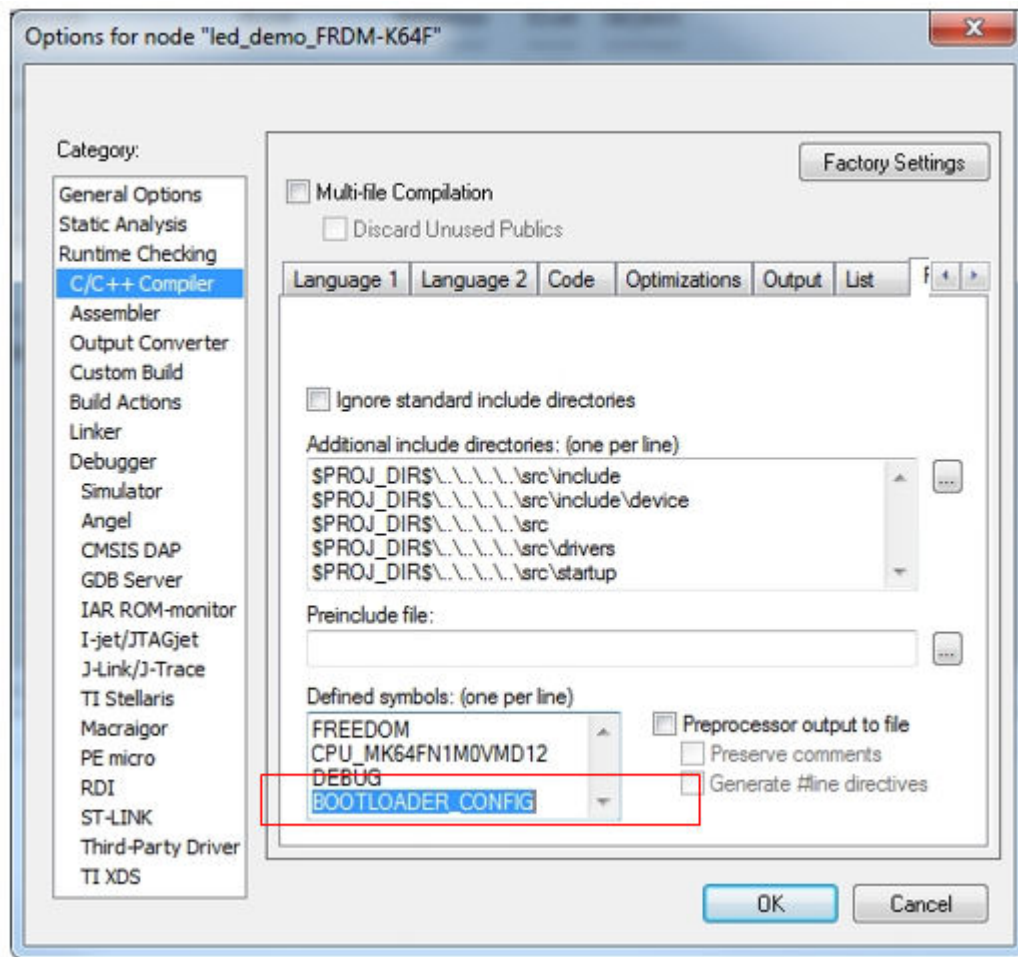


Figure 55. Add global macro

4. Modify the linker configuration file and add the configuration of *BootloaderConfig*.

```
startup_MK64F12.s K64FN1Mxxx12_application_0xA000.icf
define symbol __application_startaddress = 0xA000; // User-defined
define symbol m_bootloader_config_start = __application_startaddress + 0x3C0;
define symbol m_bootloader_config_end = __application_startaddress + 0x3FF;
define region m_bootloader_config_region = mem:[from m_bootloader_config_start to m_bootloader_config_end];
place in m_bootloader_config_region{ section BootloaderConfig };
```

Figure 56. Add BootloaderConfig

5. Compile the project.

Here are examples of the linker configuration files:

- IAR

```
define symbol __application_startaddress = 0; // User-defined
define symbol m_bootloader_config_start = __application_startaddress + 0x3C0;
define symbol m_bootloader_config_end = __application_startaddress + 0x3FF;
define region m_bootloader_config_region = mem:[from m_bootloader_config_start to
m_bootloader_config_end];
place in m_bootloader_config_region{ section BootloaderConfig };
```

- **Keil**

```
#define __application_startaddress = 0 // User-defined
#define m_bootloader_config_start __application_startaddress + 0x3C0
#define m_bootloader_config_size 0x00000040
LR_m_bootloader_config m_bootloader_config_start m_bootloader_config_size{
    ER_m_bootloader_config m_bootloader_config_start m_bootloader_config_size{ ; load address =
    execution address
        * (BootloaderConfig)
    }
}
```

- **GCC**

```
MEMORY
{
    m_bootloader_config(RX) : ORIGIN = 0x000003C0, LENGTH = 0x00000040
}
.bootloader_config :
{
    . = ALIGN(4);
    KEEP(*(.BootloaderConfig)) // Bootloader Configuration Area (BCA)
    . = ALIGN(4);
} > m_bootloader_config
```

5.9 Exit program

The Kinetis Flash Tool offers two ways to exit the program:

1. Pressing the "ESC" button.
2. Clicking the "Exit" button at the right top of the main window.

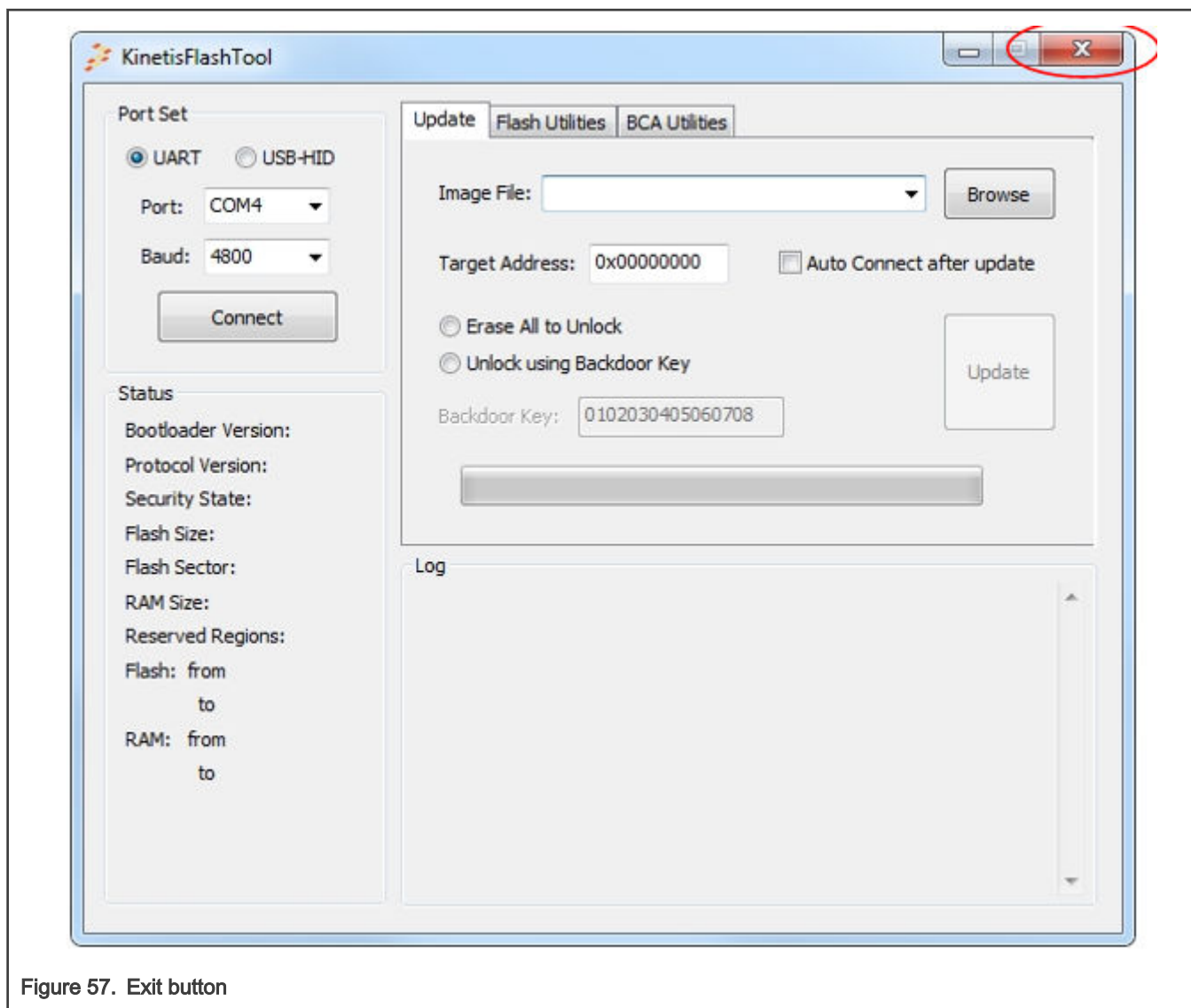


Figure 57. Exit button

If the communication is established, the Kinetis Flash Tool sends a reset command and sets the device back to the get-active-peripheral state before exiting.

6 Revision history

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

Table 2. Revision history

Revision number	Date	Substantive changes
0	04/2016	Kinetis bootloader v2.0 release
1	05/2018	MCU Bootloader v2.5.0 release
2	09/2018	MCU Bootloader v2.6.0 release
3	11/2018	MCU Bootloader v2.7.0 release
4	12/2020	MCU Bootloader v2.7.1 release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Right to make changes - NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, ICODE, JCOP, LIFE, VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, Tower, TurboLink, EdgeScale, EdgeLock, eIQ, and Immersive3D are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© NXP B.V. 2018–2020.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 28 December 2020
Document identifier: MBOOTFLT00LUG

