AN14460

How to program MCX N series internal flash through ISP

Rev. 2.0 — 15 April 2025 Application note

Document information

Information	Content
Keywords	MCXNx4x/Nx3x, ISP, Flash Programming, blhost, MCUXpresso Secure Provisioning, MCX N, FRDM-MCXN947
Abstract	This application note describes how to use USB/UART/SPI/I2C ISP to program internal flash of MCX N series MCUs via blhost or MCUXpresso Secure Provisioning.

1 Introduction

MCX N series of highly integrated Arm Cortex-M33 microcontrollers are designed for high performance and low power consumption. MCX N series has a built-in 256 KB ROM for storing the boot code and time-critical software library routines. After a reset, the Cortex-M33 processor starts its code execution from the ROM memory.

This application note describes how to use USB/UART/SPI/I^2C ISP to program internal flash of MCX N series MCUs via blhost or MCUXpresso Secure Provisioning Tool.

This document is based on the MCX N series MCU. However, it is able to apply to the MCX full series if its ISP protocol is supported by Blhost or MCUXpresso Secure Provisioning.

2 ISP feature in MCX N series

In-System Programming (ISP) in the MCX N series provides the following features:

- Multiple peripheral interfaces support (such as USB, UART, SPI, I^2C, CAN)
- Automatic detection of the active peripheral
- UART peripheral implements auto-baud detection
- CAN peripheral implements auto-baud detection for predefined baud rates: 1 Mbit/s, 500 kbit/s, 250 kbit/s, 125 kbit/s
- Common packet-based protocol for all peripherals
- Packet error detection and retransmission
- Flash-resident configuration options (in CMPA)
- RAM protection used by the bootloader while it is running
- Retrieval of the device properties, such as flash memory and RAM size
- Multiple options for executing the bootloader, either at system startup or under application control at runtime
- Support for internal flash memory access
- Support for encrypted image downloading
- External flash memory access
- In CMPA, the usage of the blhost tool command 'write-memory' to program or update the CMPA

3 Requirements

Hardware Requirements:

- FRDM-MCXN947
- Windows PC
- Type C USB cable

Software Requirements:

- MCUXpresso Secure Provisioning Tool v9.0.1

4 Hardware settings

This is the FRDM-MCXN947 board. The layout of RESET button (SW1), ISP button (SW3), MCU-Link USB connector (J17), and High-Speed USB connector (J11) are shown in Figure 1.

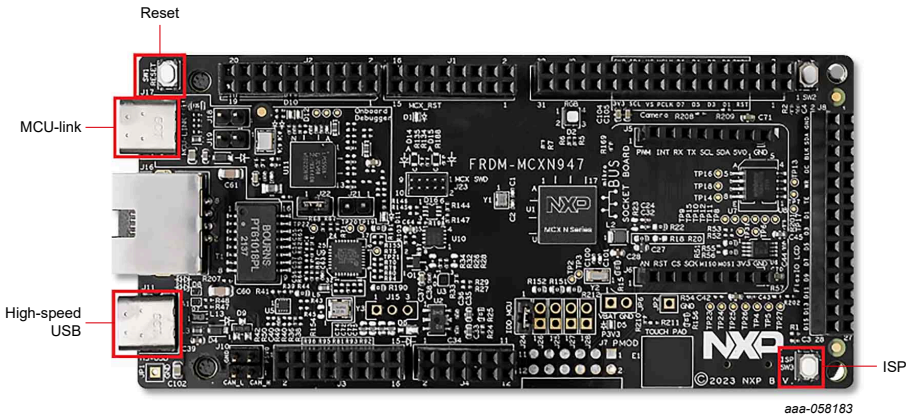

Figure 1. FRDM-MCXN947

To enable the SPI bridge and I^{2}C bridge functions, reworks for FRDM-MCXN947 are required. To enable the SPI bridge, populate R136, R137, R138, R139 as 0 Ω. To enable I^{2}C bridge, populate R142 and R143 as 0Ω, also R140 and R141 as 2.2 KΩ.

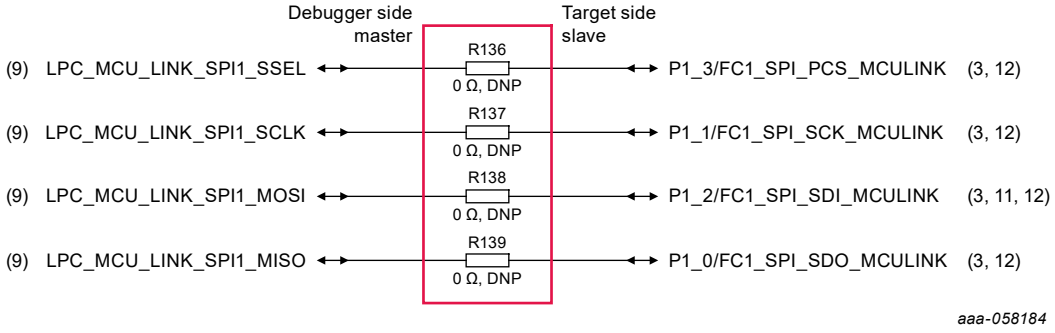

Figure 2. Resistors needed to rework for SPI bridge

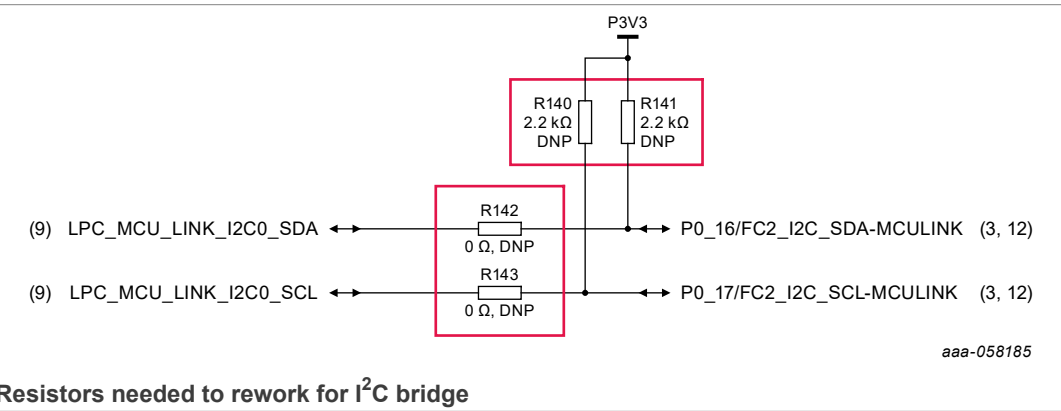

Figure 3. Resistors needed to rework for I^{2}C bridge

For the placement of the resistors needed to rework, refer to Figure 4.

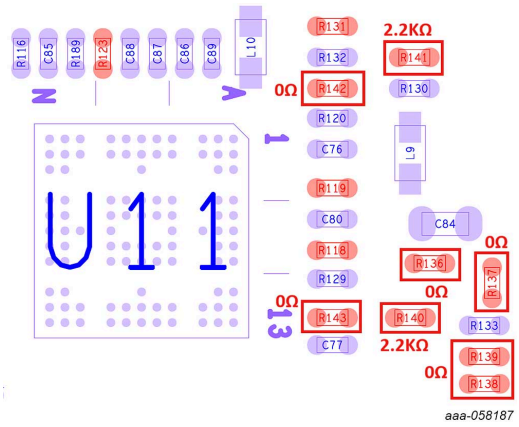

Figure 4. Placement of resistors needed to rework

5 Flash programming

The MCX N series includes ISP functions to support image programming via the serial interface (UART, I^2C, SPI, CAN) and USB HID. NXP provides both a command-line tool (blhost) and a GUI tool (MCUXpresso Secure Provisioning) to support flash programming via In-System Programming (ISP). The ISP boot flow is shown in Figure 5.

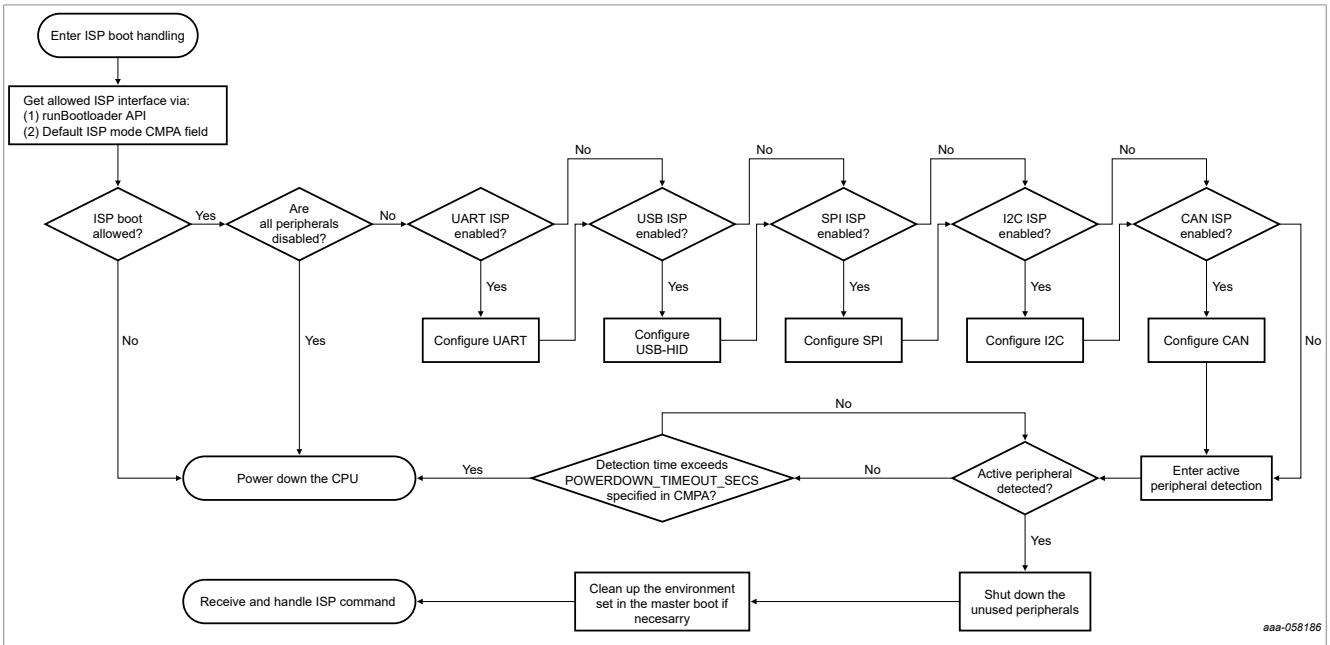

Figure 5. ISP boot flow

5.1 Flash programming with MCUXpresso Secure Provisioning

To perform flash programming via MCUXpresso Secure Provisioning, follow the steps below:

1. Enter ISP mode via the operation sequence below:
   a. Click the **RESET** button (SW1)
   b. Click the **ISP** button (SW3)
   c. Unclick the **RESET** button (SW1)

d. Unclick the **ISP** button (SW3)

2. Open MCUXpresso Secure Provisioning and click the **Selection of the target processor** button marked as 1 to open the **Select Processor** window. If the current processor is what is required, click **OK**, if not, create a workspace by clicking **create a new workspace**.

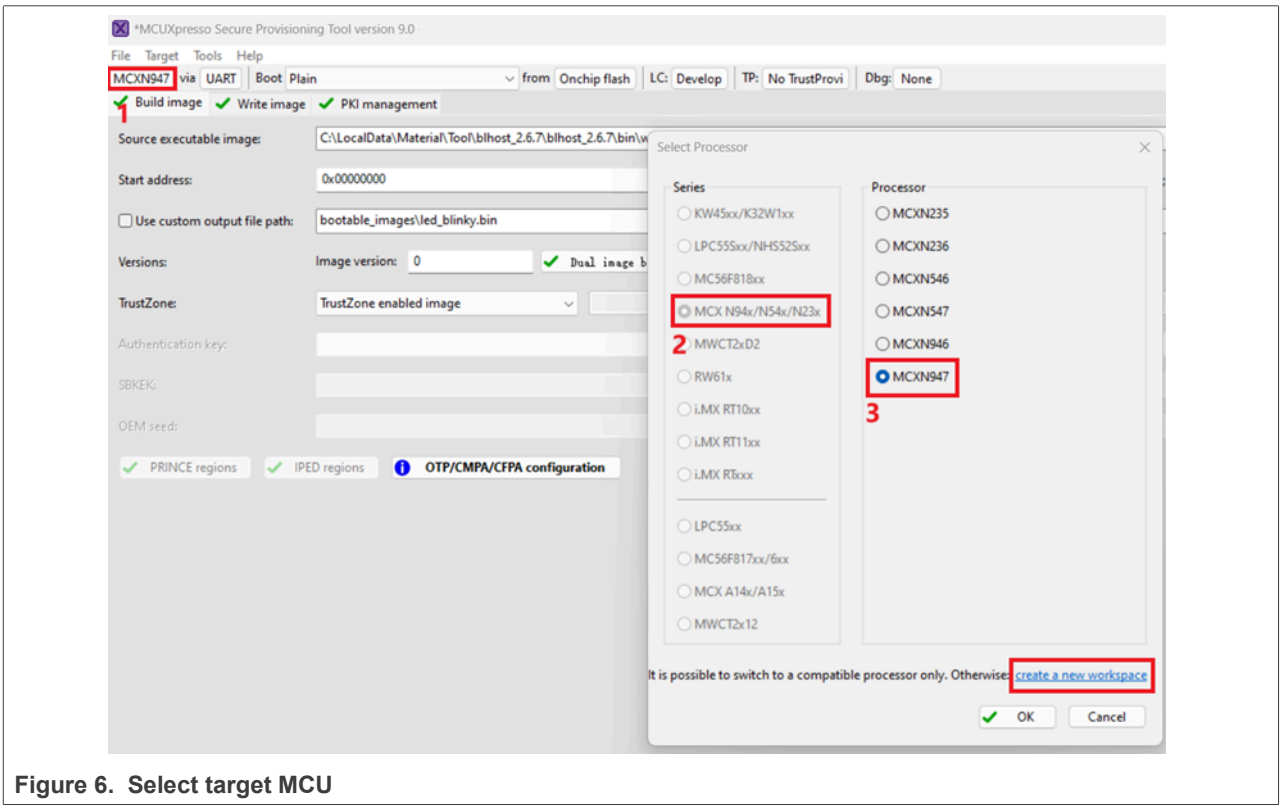

Figure 6. Select target MCU

3. Select the desired communication interface for flash programming. As for the supported communication interfaces, it depends on the ISP features of the selected processor. For MCXN947, USB HID, UART, SPI, and I^2C are supported. Click the **Selection of the communication interface** button marked as 1 to open the **Connection with Target Processor** window.

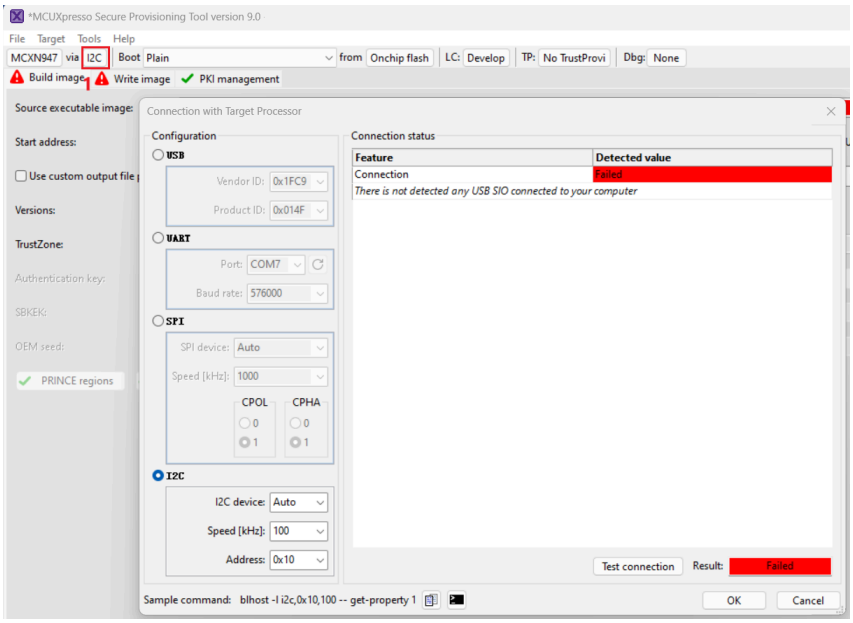

Figure 7. Select communication interface

4. To program flash via the USB interface, the USB connector is J11. Since the Vendor ID and Product ID are automatically filled based on the selected processor, check the connection between the PC and the target processor by clicking the **Test connection** button. If the result is OK, it means that the connection is established successfully.

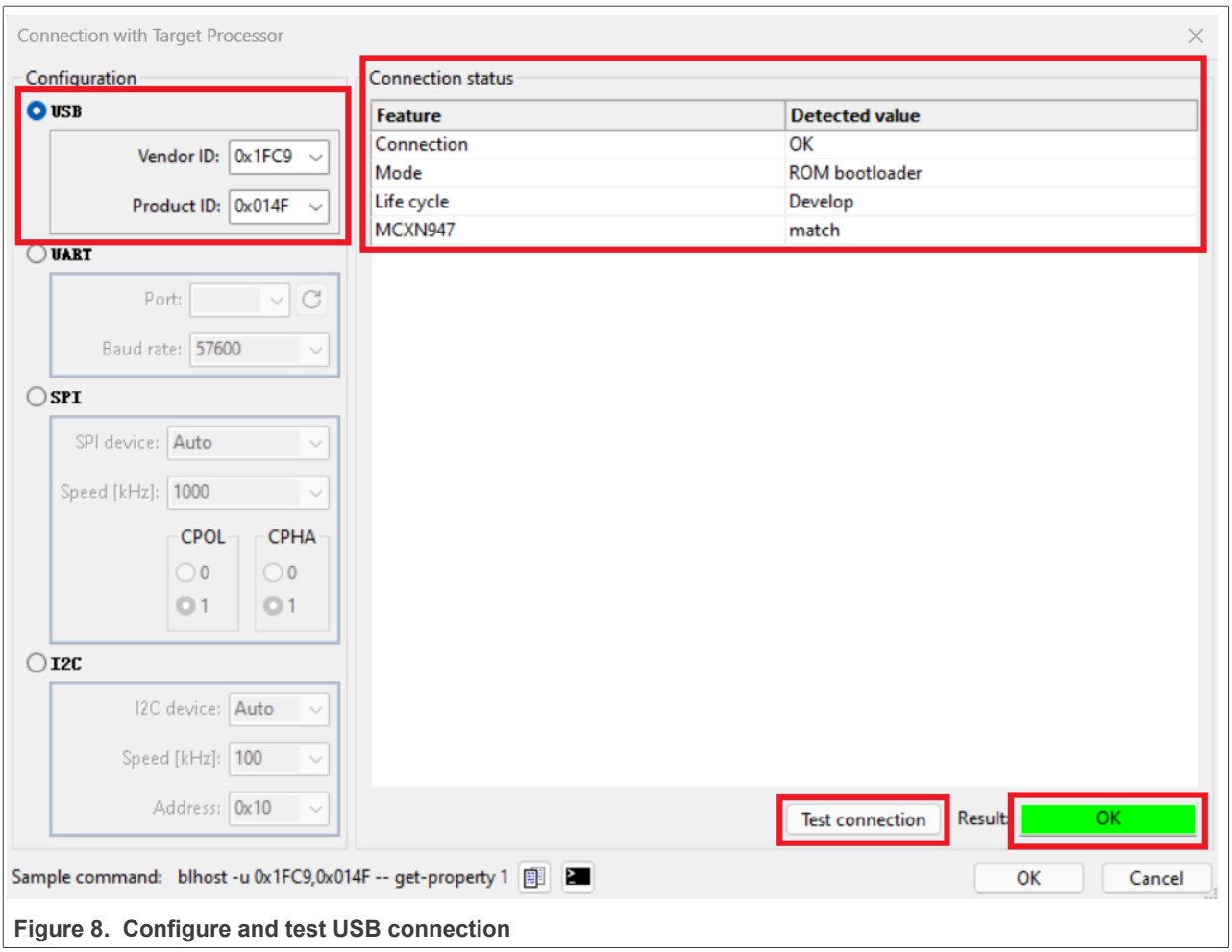

Figure 8. Configure and test USB connection

To program flash via the UART/SPI/I^{2}C interface, the USB connector is J17. For UART, select the correct port number and proper baud rate as shown in Figure 9.

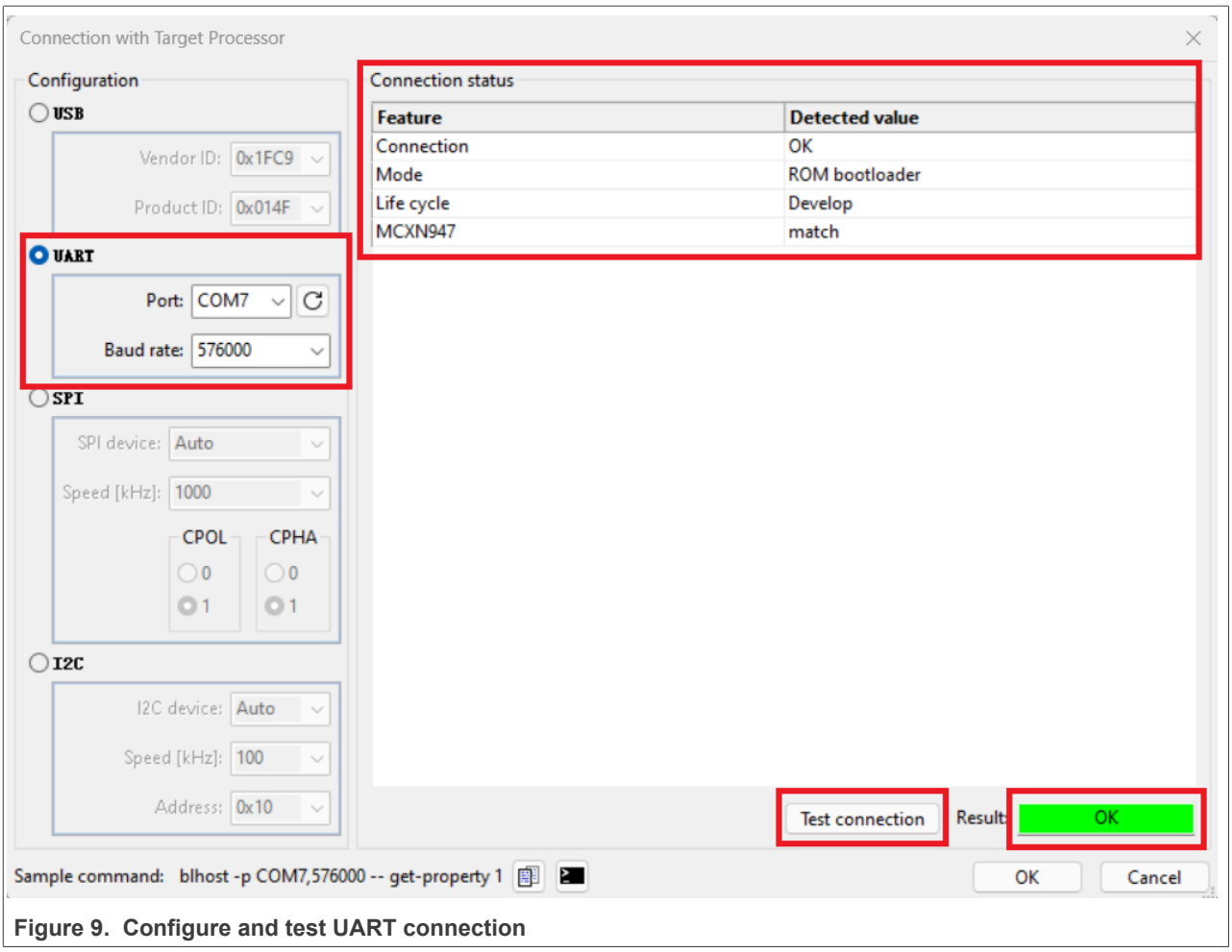

Figure 9. Configure and test UART connection

For SPI, set the communication speed, CPOL, and CPHA. Use the default settings as shown in Figure 10.

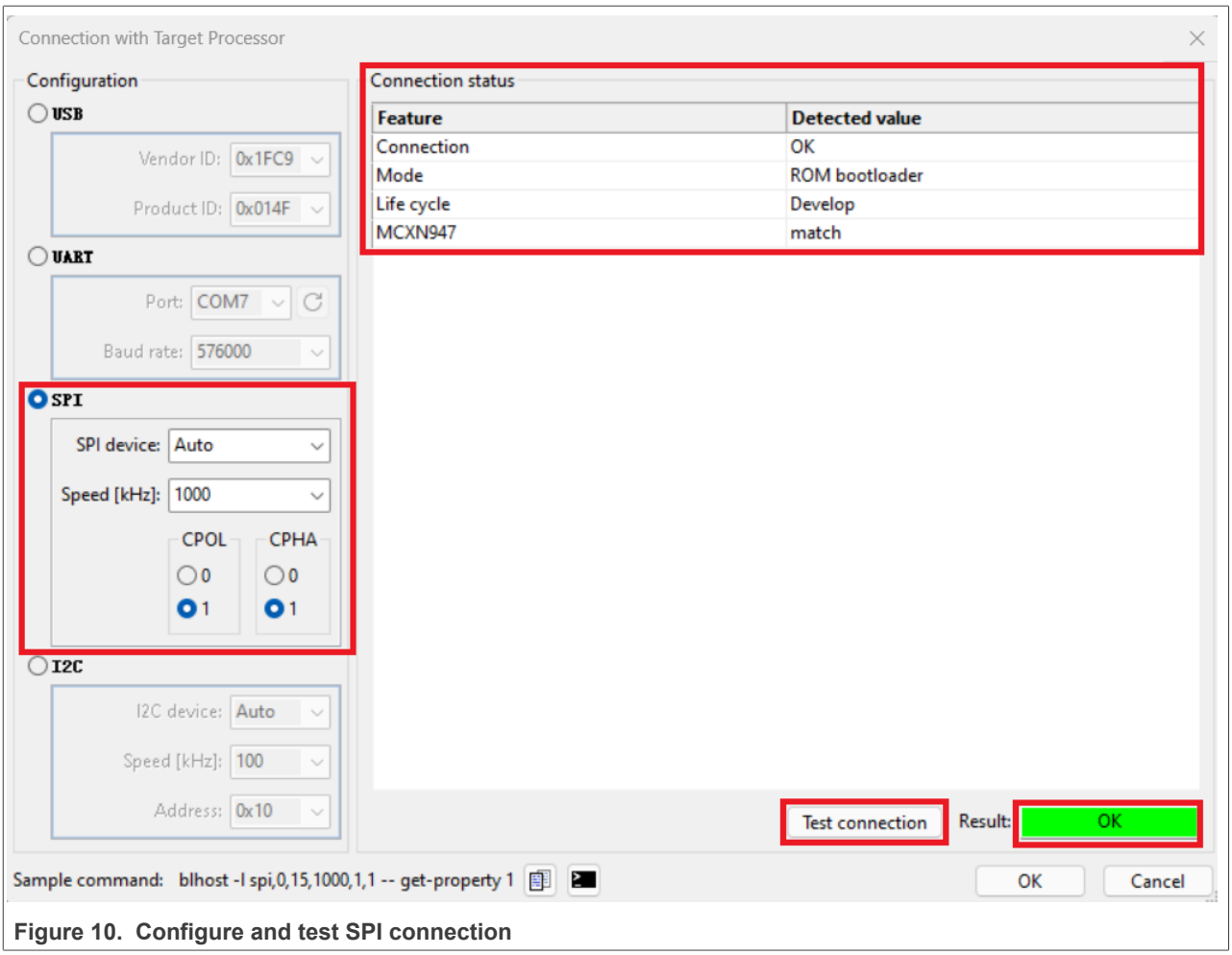

Figure 10. Configure and test SPI connection

For I^2C, set communication speed and address. Use the default settings as shown in Figure 11.

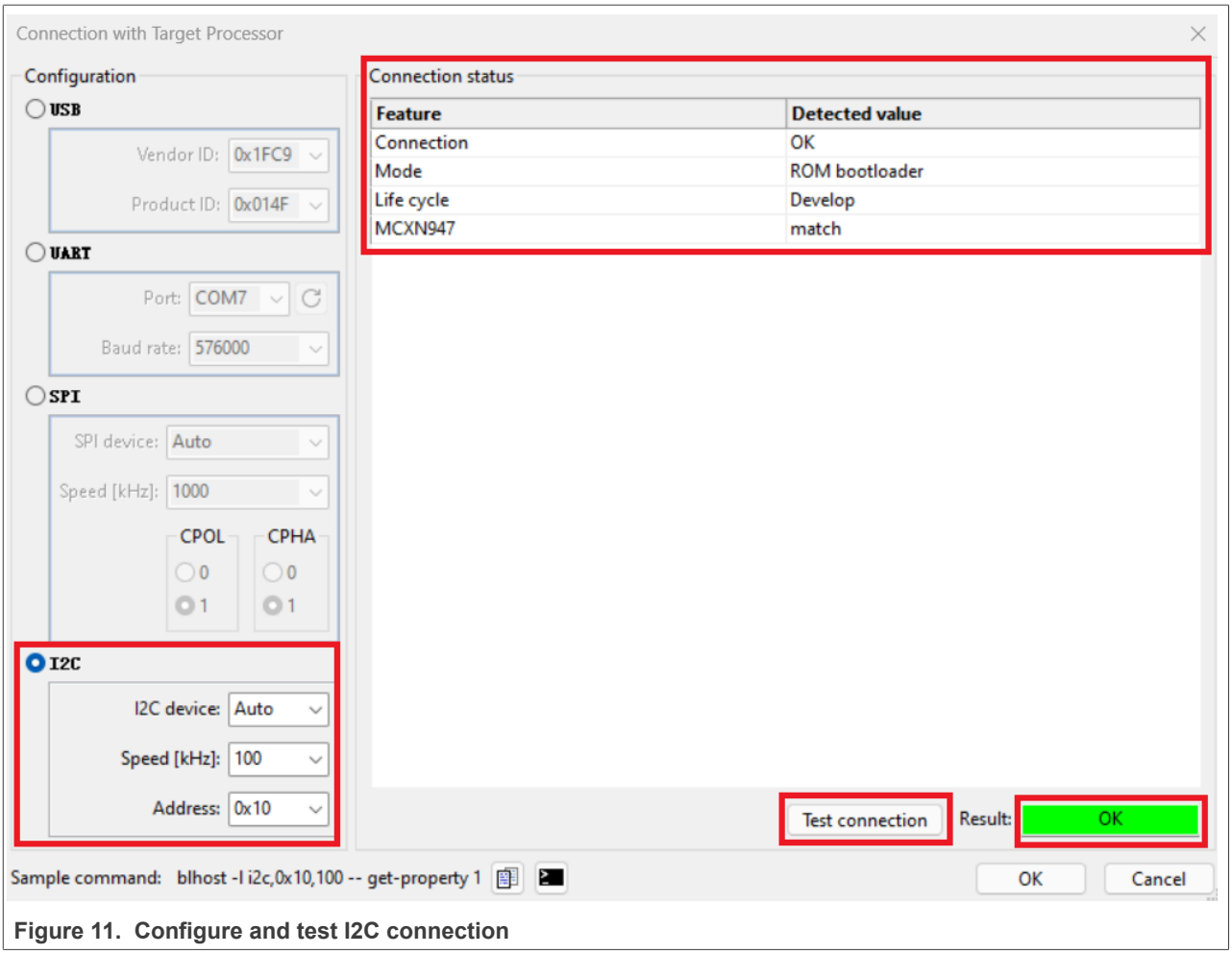

Figure 11. Configure and test I2C connection

5. Once the communication interface is configured completely, perform the build image operation by clicking the **Build image** button. Check the log information to grasp the details of the image building.

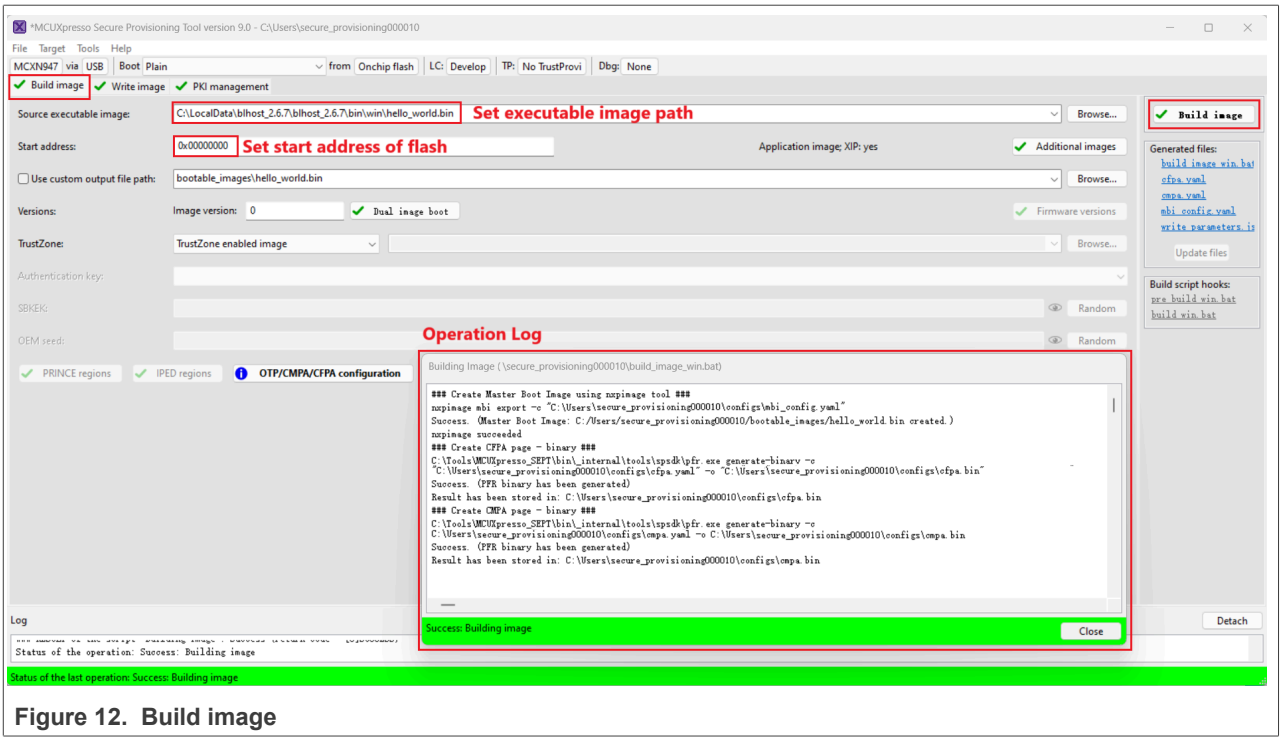

Figure 12. Build image

6. Perform the write image operation by clicking the **Write image** button to erase the flash and program the build image into the flash.

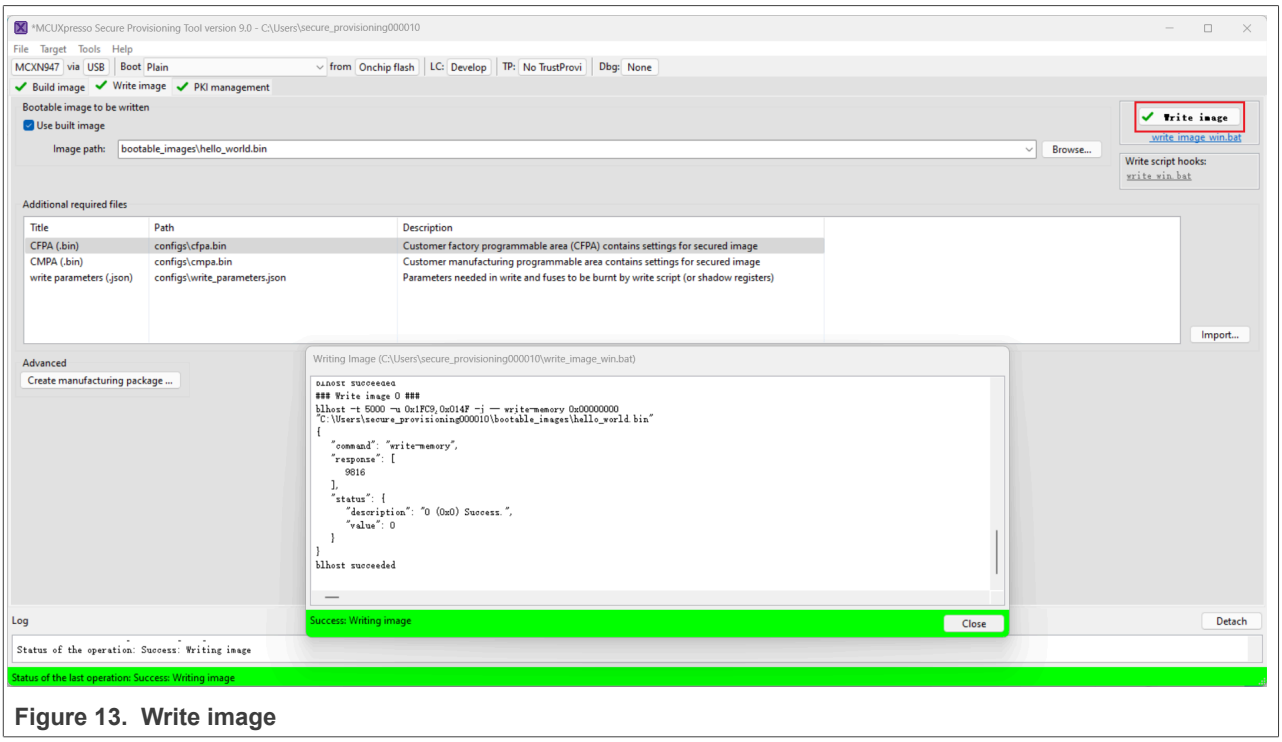

Figure 13. Write image

7. Power on the board or reset the board to run the executable image.

5.2 Flash programming with blhost

To erase and program flash via blhost, follow the steps below:

1. Enter ISP mode via the below operation sequence:
   a. Click the **RESET** button (SW1)
   b. Click the **ISP** button (SW3)
   c. Unclick the **RESET** button (SW1)
   d. Unclick the **ISP** button (SW3)
2. Open a command-line terminal by clicking the icon below in the **Connection with Target Processor** window as shown in Figure 14.

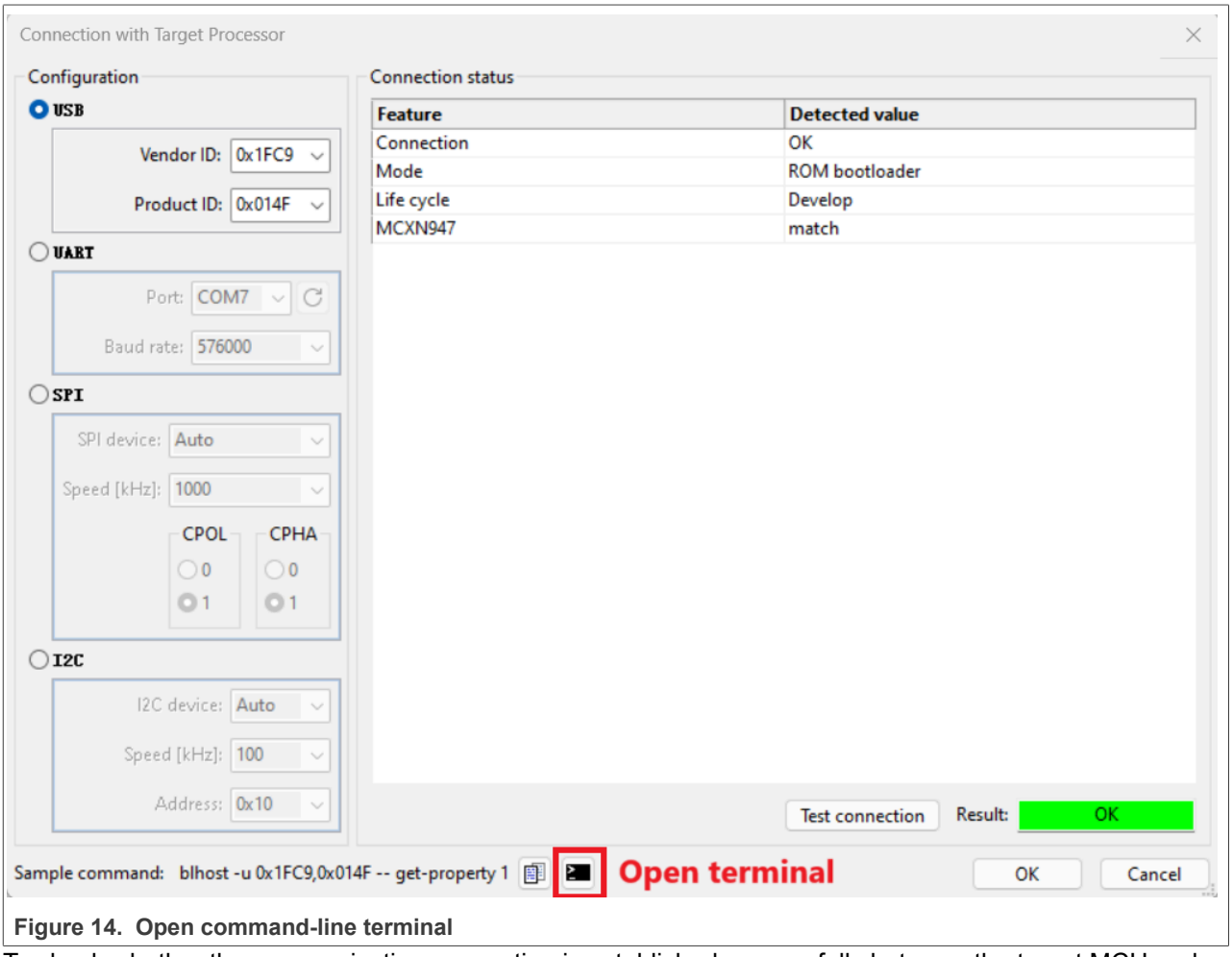

Figure 14. Open command-line terminal

3. To check whether the communication connection is established successfully between the target MCU and the command-line terminal on the PC, perform the `get-property` command as shown below.
   For USB: `blhost -u 0x1FC9,0x014F -j -- get-property 1`

```
blhost -u 0x1FC9,0x014F -j -- get-property 1
{
   "command": "get-property",
   "response": [
      1258488320
   ],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 15. Check USB ISP connection

For UART: `blhost -p COM14,115200 -j -- get-property 1`

```
blhost -p COM14,115200 -j -- get-property 1
{
   "command": "get-property",
   "response": [
      1258488320
   ],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 16. Check UART ISP connection

For SPI: `blhost -l spi,0,15,1000,1,1 -j -- get-property 1`

```
blhost -l spi,0,15,1000,1,1 -j -- get-property 1
{
   "command": "get-property",
   "response": [
      1258488320
   ],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 17. Check SPI ISP connection

For I^2C: `blhost -l i2c,0x10,400 -j -- get-property 1`

```
blhost -l i2c,0x10,400 -j -- get-property 1
{
   "command": "get-property",
   "response": [
      1258488320
   ],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 18. Check I²C ISP connection

4. Erase the target flash region with `flash-erase-region` or `flash-erase` all blhost commands.
   For USB: `blhost -u 0x1FC9,0x014F -j -- flash-erase-all`

```
blhost -u 0x1FC9,0x014F -j -- flash-erase-all
{
   "command": "flash-erase-all",
   "response": [],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 19. Erase flash via USB ISP

For UART: `blhost -p COM14,115200 -j -- flash-erase-all`

```
blhost -p COM14,115200 -j -- flash-erase-all
{
   "command": "flash-erase-all",
   "response": [],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 20. Erase flash via UART ISP

For SPI: `blhost -l spi,0,15,1000,1,1 -j -- flash-erase-all`

```
blhost -l spi,0,15,1000,1,1 -j -- flash-erase-all
{
   "command": "flash-erase-all",
   "response": [],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 21. Erase flash via SPI ISP

For I²C: `blhost -l i2c,0x10,400 -j -- flash-erase-all`

```
blhost -l i2c,0x10,400 -j -- flash-erase-all
{
   "command": "flash-erase-all",
   "response": [],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 22. Erase flash via I²C ISP

5. Program the build image into the target flash with the `write-memory` blhost command.
   For USB: `blhost -u 0x1FC9,0x014F -j -- write-memory 0x0 C:\LocalData\bin\led_blinky.bin`

```
blhost -u 0x1FC9,0x014F -j -- write-memory 0x0 C:\LocalData\bin\led_blinky.bin
{
   "command": "write-memory",
   "response": [
      5800
   ],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 23. Program flash via USB ISP

For UART: `blhost -p COM14,115200 -j -- write-memory 0x0 C:\LocalData\bin\led_blinky.bin`

```
blhost -p COM14,115200 -j -- write-memory 0x0 C:\LocalData\bin\led_blinky.bin
{
   "command": "write-memory",
   "response": [
      5800
   ],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 24. Program flash via UART ISP

For SPI: `blhost -l spi,0,15,1000,1,1 -j -- write-memory 0x0 C:\LocalData\bin\led_blinky.bin`

```
blhost -l spi,0,15,1000,1,1 -j -- write-memory 0x0 C:\LocalData\bin\led_blinky.bin
{
   "command": "write-memory",
   "response": [
      5800
   ],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 25. Program flash via SPI ISP

For I²C: `blhost -l i2c,0x10,400 -j -- write-memory 0x0 C:\LocalData\bin\led_blinky.bin`

```
blhost -l i2c,0x10,400 -j -- write-memory 0x0 C:\LocalData\bin\led_blinky.bin
{
   "command": "write-memory",
   "response": [
      5800
   ],
   "status": {
      "description": "0 (0x0) Success.",
      "value": 0
   }
}
```

Figure 26. Program flash via I2C ISP

6. Power on the board or reset the board to run the executable image, the RGB led(D2) must be blinking with the time interval of 1 s. Use the `reset` blhost command to reset the board.

6 Conclusion

This document introduces two ways to implement flash erasing and programming on MCX N series via ISP including MCUXpresso Secure Provisioning and blhost. This document gives detailed steps to describe how to perform flash programming with MCUXpresso Secure Provisioning and what blhost commands are used to check communication connection, erase, and program flash.

7 Revision history

Table 1. Revision history

Document ID	Release date	Description
AN14460 v.2.0	15 April 2025	Figure 5 is updated, minor editorial changes.
AN14460 v.1.0	5 November 2024	Initial version

8 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. 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 B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents