

AN14393

Using Serial Downloader Option in i.MX RT1180 via LPSPI

Rev. 1.0 — 4 September 2024

Application note

Document information

Information	Content
Keywords	AN14393, MIMXRT1180-EVK, i.MX RT1180, serial downloader, MCU-Link Pro, BusPal, USBSIO
Abstract	This document describes two methods for Serial Downloader via LPSPI on the i.MX RT1180.



1 Introduction

This document describes two methods for Serial Downloader via LPSPI on the i.MX RT1180 as follows:

- [Serial Downloader via LPSPI by BusPal and MIMXRT1010-EVK](#): BusPal acts as a bus translator between UART and SPI. It is realized by the software and must run on one hardware platform. BusPal can receive a blhost command by UART and send it to the MIMXRT1180-EVK platform by SPI.
- [Serial Downloader via LPSPI by USBSIO and MCU-Link Pro](#): USBSIO is the feature supported by MCU-Link Pro. It is a firmware based on LPC55S69 and can communicate with serial I/O devices. USBSIO can receive the blhost command by USB and send it to the MIMXRT1180-EVK platform by SPI.

1.1 Serial Downloader

i.MX RT1180 Serial Downloader (serial boot) helps download a boot image to the boot device and then the image can be run on these devices.

[Table 1](#) lists Serial Downloader peripherals PinMux on i.MX RT1180.

Table 1. Serial Downloader peripherals PinMux

Peripheral	Instance	Port (IO function)	PAD
LPUART	1	LPUART1_TX	GPIO_AON_08
		LPUART1_RX	GPIO_AON_09
LPSPI	1	LPSPI1_SCK	GPIO_AON_04
		LPSPI1_PCS0	GPIO_AON_05
		LPSPI1_SDO	GPIO_AON_06
		LPSPI1_SDI	GPIO_AON_07
USB	1	USB1_DN	USB1_DN
		USB1_DP	USB1_DP

[Figure 1](#) shows the Serial Downloader diagram. The blhost on the PC can communicate with i.MX RT1180 via USB and LPUART directly. However, if the user wants to communicate with LPSPI, either USBSIO or BusPal are required. The MCU-Link Pro has the integrated USBSIO feature and BusPal can be customized with many platforms.

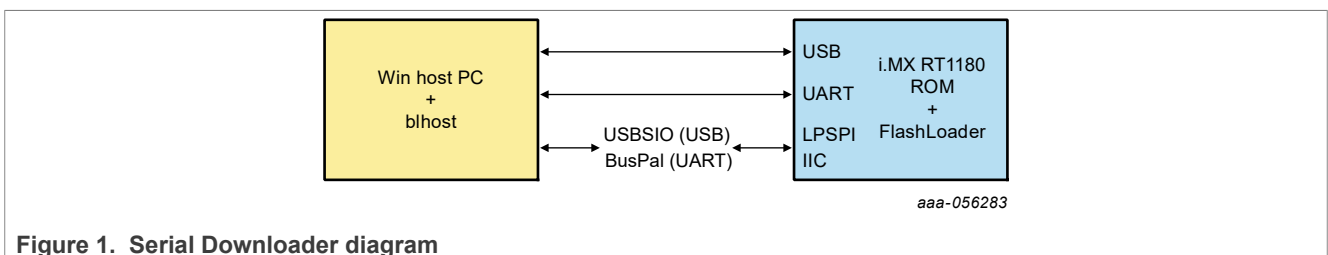


Figure 1. Serial Downloader diagram

1.2 MCU-Link Pro

NXP and Embedded Artists have developed the MCU-Link Pro. The MCU-Link Pro is a fully featured debug probe that can be used with MCUXpresso IDE and third-party IDEs that support CMSIS-DAP or J-Link protocols. [Figure 2](#) shows the MCU-Link Pro.

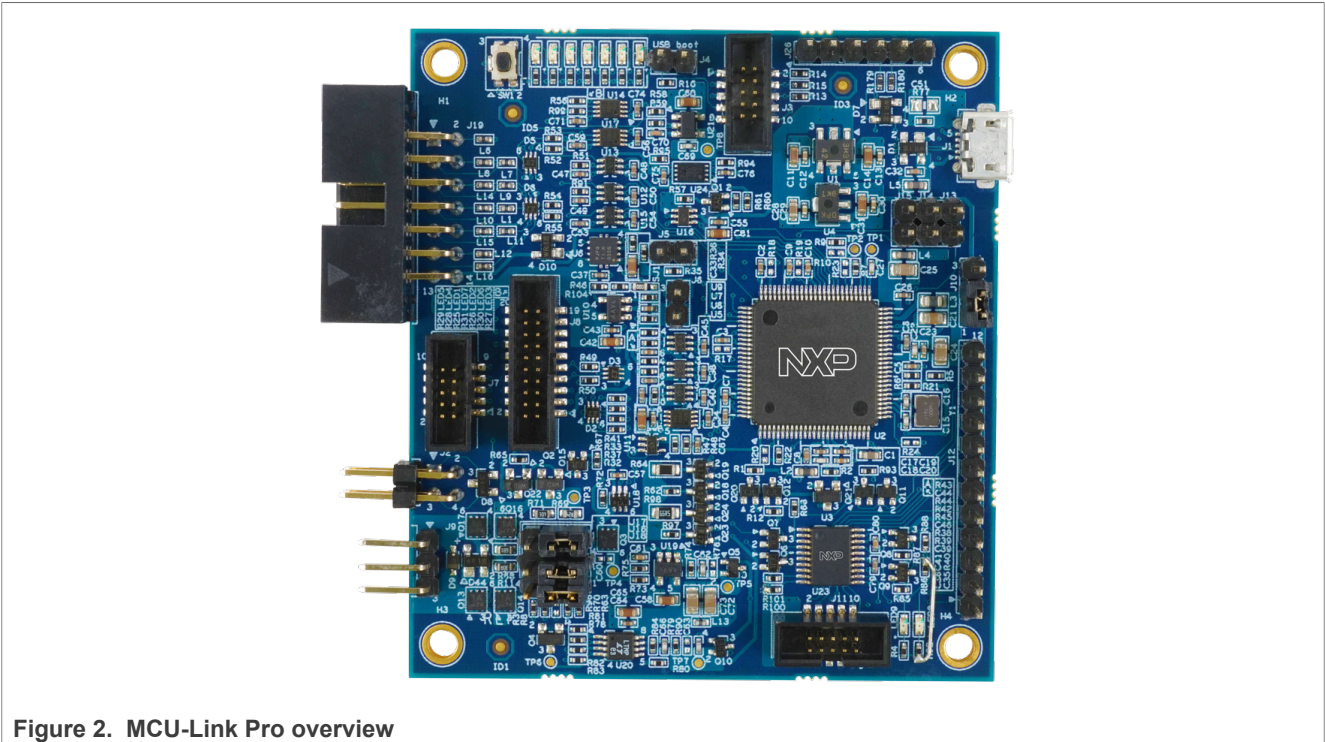


Figure 2. MCU-Link Pro overview

The MCU-Link Pro includes the following features:

- CMSIS-DAP firmware supports all NXP Arm Cortex-M-based MCUs with SWD debug interfaces
- SEGGER J-Link firmware option
- High-speed USB host interface
- Dual USB to target UART bridges (VCOM)
- Simultaneous target supply and current measurement
- SWO profiling and I/O features
- CMSIS-SWO support
- USB SPI and I2C bridges for programming/provisioning and host-based application development
- Option to power target system up to 350 mA (at 1.8 V or 3.3 V)
- Analog signal trace input
- Onboard user-programmable LPC804 for peripheral emulation
- Multiple status LEDs for diagnosis of issues
- Target reset button

1.3 Blhost

The blhost application is used on a host computer to issue commands to an NXP platform running an implementation of the MCU bootloader. The blhost application with the MCU bootloader allows a user to program a firmware application onto the MCU device without a programming tool. Blhost can be download from the NXP official website.

The blhost application can connect with an MCU bootloader via IIC SPI UART and USB. The blhost also supports BusPal and USBSIO, because the IIC and SPI commands are only valid for the Arm-Linux blhost. If the host PC is used, BusPal and USBSIO are required.

1.4 BusPal

BusPal is an embedded software tool available as a companion to blhost. The tool functions as a bus translator, establishing a connection with the blhost over UART and with the target device over I2C and SPI. It assists the blhost in carrying out commands and responses from the target device.

The source code for BusPal is provided with the MCU bootloader release and can be customized to run on other platforms. [Figure 3](#) illustrates the role BusPal plays in blhost communication with the target device.

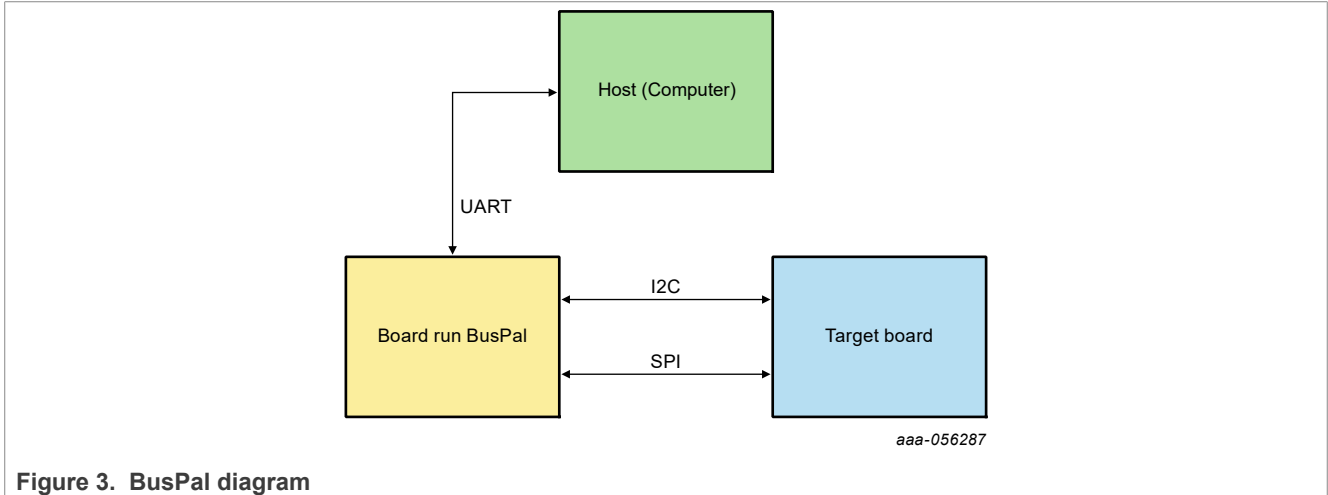


Figure 3. BusPal diagram

1.5 USBSIO

USBSIO, also named as LPC USB Serial I/O (LPCUSBSIO), is a firmware based on LPC55S69 and can communicate with serial I/O devices connected to MCU-Link Pro.

The USBSIO contains two parts:

- *LPC USB Serial I/O Library*: It is a generic API provided to PC applications. This part is built in the blhost.
- *LPC Serial I/O port*: It is the firmware that receives USB messages and transfers them to devices using I2C, SPI, and GPIO interfaces. This part is built in the MCU-Link Pro and MCU-Link does not support USBSIO.

2 Quick start

This chapter introduces a quick start for the MCU-Link Pro, MIMXRT1010-EVK board, MIMXRT1180-EVK board, and hardware preparation.

2.1 Prepare MCU-Link Pro

Ensure that the MCU-Link Pro firmware has been downloaded to the board. MCU-Link Pro firmware can be downloaded from [Getting Started with the MCU-Link Pro](#).

[Figure 4](#) shows the digital I/O connector (J19) on MCU-Link Pro.

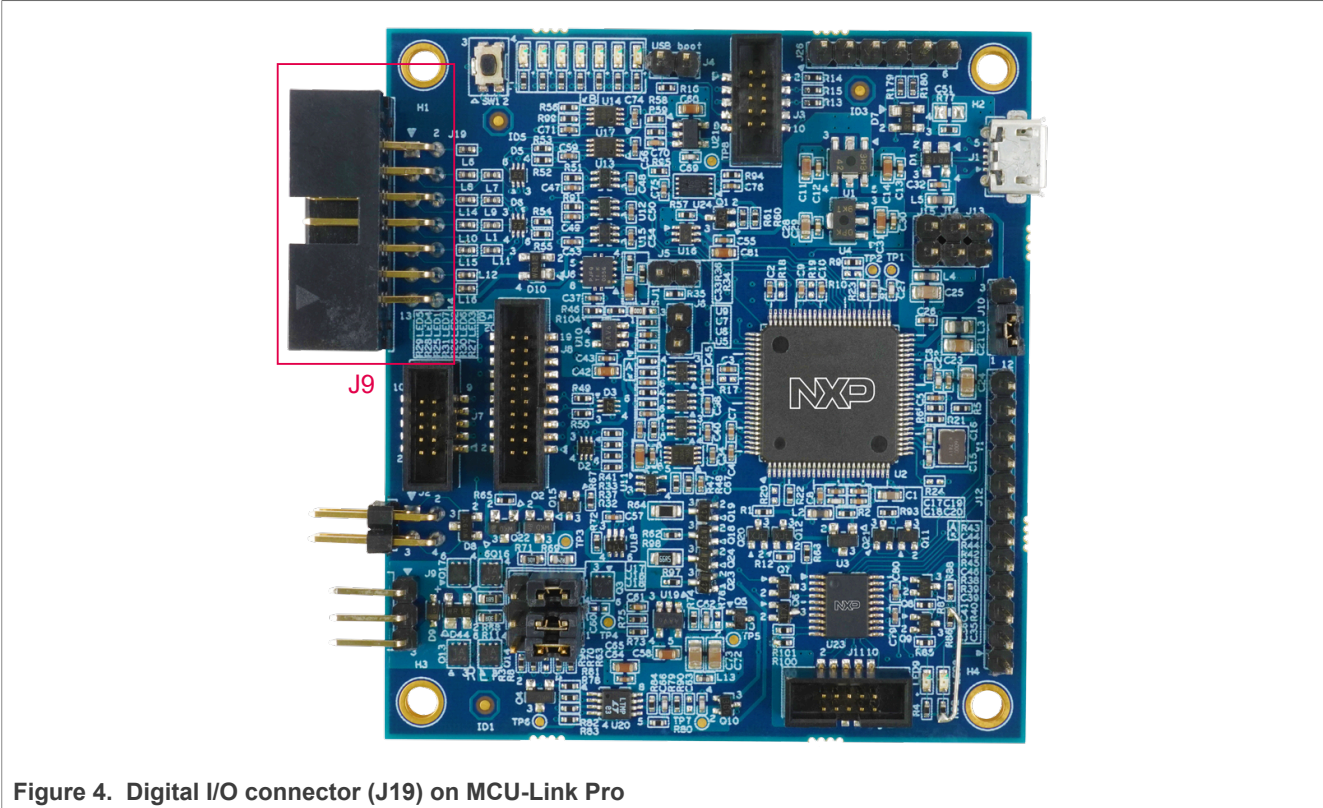


Figure 4. Digital I/O connector (J19) on MCU-Link Pro

Table 2 shows the connection pins for SPI on the digital I/O connector (J19).

Table 2. Connection pins for SPI on digital I/O connector (J19)

J19 pin	Signal
1	GND
2	SPI PCS
3	SPI MOSI
4	SPI CLK
5	SPI MISO

2.2 Prepare MIMXRT1010-EVK board

In this document, the BusPal runs on the MIMXRT1010-EVK platform. Ensure that the BusPal code has been downloaded to the MIMXRT1010-EVK board.

Download the BusPal example code from [NXP Kinetis Bootloader 2.0.0 package](#). It can also be ported to other customized platforms using the same link.

Note: MIMXRT1010-EVK is not included in this package, therefore, porting of BusPal is required.

Figure 5 shows the digital I/O connector J56, J57, and J60 on the MIMXRT1010-EVK.

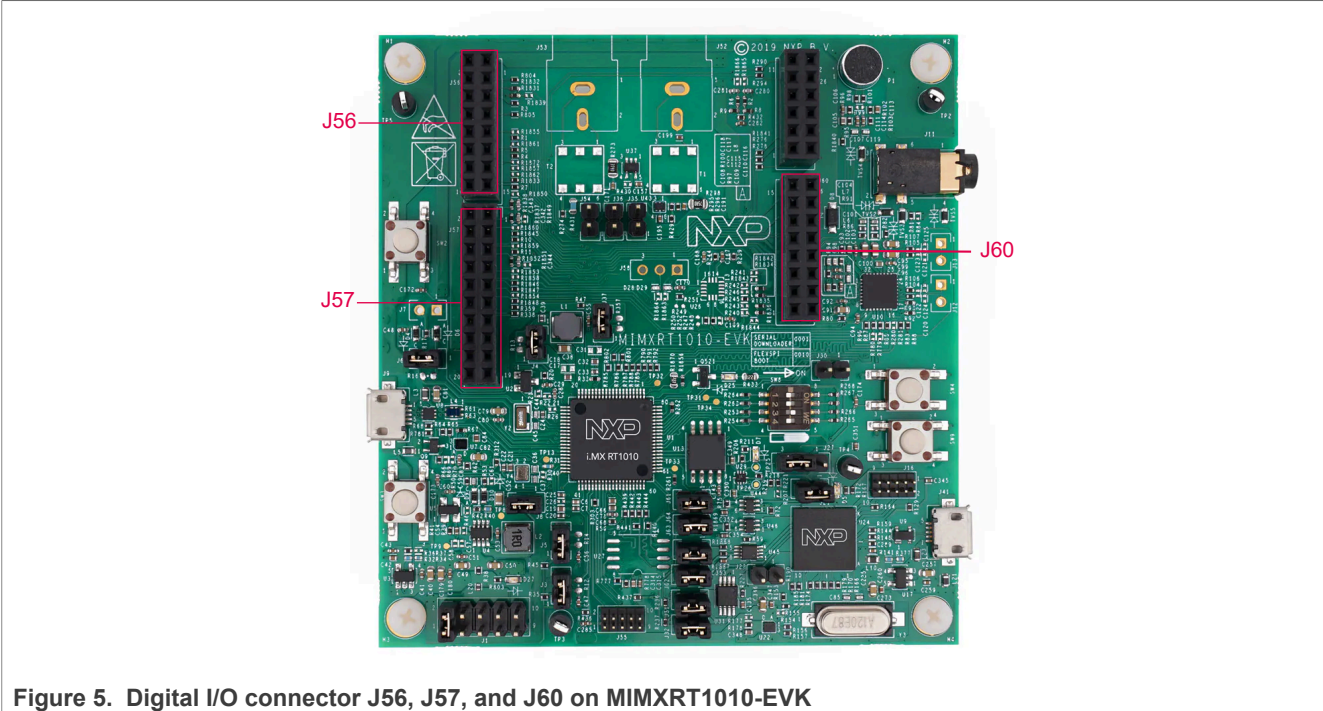


Figure 5. Digital I/O connector J56, J57, and J60 on MIMXRT1010-EVK

[Table 3](#) shows the connection pins for UART on the J56.

Table 3. Connection pins for UART on J56

J56 pin	Signal
2	LPUART1_RXD
4	LPUART1_TXD

[Table 4](#) shows the connection pins for SPI on the J57.

Table 4. Connection pins for SPI on J57

J57 pin	Signal
6	LPSPI1_PCS0
8	LPSPI1_MOSI
10	LPSPI1_MISO
12	LPSPI1_SCK

2.3 Prepare MIMXRT1180-EVK board

The MIMXRT1180-EVK board is a platform designed to show the commonly used features of the i.MX RT1180 processor. However, the LPSPI1 pins on EVK cannot be connected directly.

[Figure 6](#) shows that the LPSPI1 pins can be reached on U13.

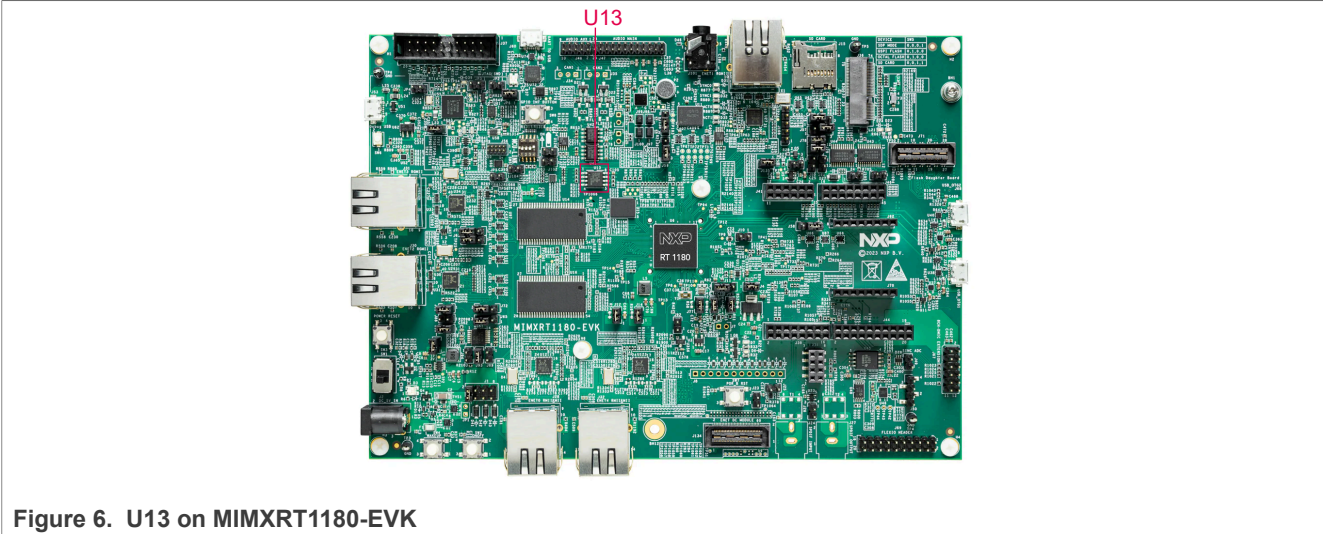


Figure 6. U13 on MIMXRT1180-EVK

Table 5 shows the connection pins for LPSPI.

Table 5. Connection pins for LPSPI on U13

U13	Signal
1	LPSP11_PCS0
2	LPSP11_MISO
5	LPSP11_MOSI
6	LPSP11_SCK

Both flying wires on these pins can be used to connect LPSPI signals on this board.

2.4 Serial downloader via LPSPI and BusPal hardware preparation

Table 6 shows the pins connection between MIMXRT1180-EVK and MIMXRT1010-EVK when BusPal is used.

Table 6. Hardware connection between MIMXRT1180-EVK and MIMXRT1010-EVK

MIMXRT1180-EVK	Pin	MIMXRT1010-EVK	Pin
LPSP11_PCS0	U13-1	LPSP11_PCS0	J57-6
LPSP11_MISO	U13-2	LPSP11_MOSI	J57-8
LPSP11_MOSI	U13-5	LPSP11_MISO	J57-10
LPSP11_SCK	U13-6	LPSP11_SCK	J57-12
GND	J37-20	GND	J57-14

Table 7 shows the pins connection between the MIMXRT1010-EVK and the host PC via the USB to UART connector.

Table 7. Hardware connection between host PC and MIMXRT1010-EVK

MIMXRT1010-EVK board	Pin
LPUART1_RXD	J56-2
LPUART1_TXD	J56-4
GND	J60-14

2.5 Serial downloader via LPSPI and USBIO hardware preparation

If using USBIO for Serial Downloader, MCU-Link Pro is used. [Table 8](#) shows the pins connection between the MIMXRT1180-EVK and MCU-Link Pro.


Table 8. Hardware connection between MIMXRT1180-EVK and MCU-Link Pro

MIMXRT1180-EVK	Pin	MCU-Link Pro	Pin
LPSP11_PCS0	U13-1	LPSP11_PCS0	J19-2
LPSP11_MISO	U13-2	LPSP11_MOSI	J19-3
LPSP11_MOSI	U13-5	LPSP11_MISO	J19-5
LPSP11_SCK	U13-6	LPSP11_SCK	J19-4
GND	J37-20	GND	J19-1

3 Serial Downloader via LPSPI by BusPal and MIMXRT1010-EVK

After hardware preparation is completed, to start the Serial Downloader via LPSPI by BusPal and MIMXRT1010-EVK, perform the following steps:

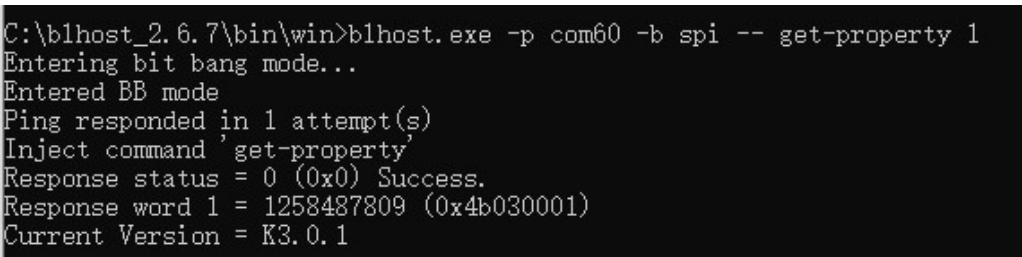
1. Switch the MIMXRT1180-EVK board to Serial Downloader mode.
2. Run the command-line tool and go to the `blhost/bin/win` directory as shown in [Figure 7](#).



```
C:\blhost_2.6.7\bin\win>
```

Figure 7. Go to the blhost Window directory

3. Enter `blhost.exe -p com1 -b spi -- get-property 1`. Check if the ROM can communicate successfully. See [Figure 8](#).



```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- get-property 1
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258487809 (0x4b030001)
Current Version = K3.0.1
```

Figure 8. Communicate with ROM via BusPal

SPI default parameters are as follows:

- Port 0 (0)
- Pin 0 (0)
- 100 kbit/s (100)
- Active low polarity (1)
- Falling edge sampling (1)
- Command send = `blhost.exe -p com1 -b spi,0,0,100,1,1 -- get-property 1`

4. Enter `blhost.exe -p com1 -b spi -- load-image <file>`. This step downloads and runs the flashloader. Ensure to add the container to the flashloader. SDK generates the flashloader. See [Figure 9](#).


```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- load-image ../../flashloader/flashloader_utility.bin
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'load-image'
Preparing to send 98868 (0x18234) bytes to the target.
(1/1)100% Completed!
Successful generic response to command 'load-image'
Response status = 0 (0x0) Success.
Response word 1 = 0 (0x0)
Wrote 98868 of 98868 bytes.
```

Figure 9. Download and run the flashloader via BusPal

- Now, blhost can communicate with the flashloader. Enter `blhost.exe -p com1 -b spi -- get-property 1` again. Check if the flashloader can communicate successfully. See [Figure 10](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- get-property 1
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258424320 (0x4b020800)
Current Version = K2.8.0
```

Figure 10. Communicate with flashloader via BusPal

- Set FlexSPI instance in TCM and enter `blhost.exe -p com1 -b spi -- fill-memory 0x20000000 4 0xcf900001`. See [Figure 11](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- fill-memory 0x20000000 4 0xcf900001
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 11. Set FlexSPI instance in TCM via BusPal

- Configure the memory and enter `blhost.exe -p com1 -b spi -- configure-memory 9 0x20000000`. See [Figure 12](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- configure-memory 9 0x20000000
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 12. Configure memory via BusPal

- Set NOR flash configuration parameters in TCM and enter `blhost.exe -p com1 -b spi -- fill-memory 0x20000000 4 0xc0000006`. See [Figure 13](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- fill-memory 0x20000000 4 0xc0000006
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 13. Set NOR flash configuration parameters via BusPal

9. Configure the memory and enter `blhost.exe -p com1 -b spi -- configure-memory 9 0x20000000`. See [Figure 14](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- configure-memory 9 0x20000000
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 14. Configure memory via BusPal

10. Erase flash region and enter `blhost.exe -p com1 -b spi -- flash-erase-region 0x38000000 0x40000`. See [Figure 15](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- flash-erase-region 0x38000000 0x40000
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.
```

Figure 15. Erase flash region via BusPal

11. To confirm the flash have been erased, read the flash memory. Enter `blhost.exe -p com1 -b spi -- read-memory 0x38000000 16`. See [Figure 16](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- read-memory 0x38000000 16
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 16 (0x10)
Read 16 of 16 bytes.
```

Figure 16. Read flash via BusPal

12. Add FDCB XIP used and enter `blhost.exe -p com1 -b spi -- fill-memory 0x20000000 4 0xf000000f`. See [Figure 17](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- fill-memory 0x20000000 4 0xf000000f
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 17. Add FDCB XIP used via BusPal

13. Configure the memory and enter `blhost.exe -p com1 -b spi -- configure-memory 9 0x20000000`. See [Figure 18](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- configure-memory 9 0x20000000
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 18. Configure memory via BusPal

14. Write the image and enter `blhost.exe -p com1 -b spi -- write-memory <path>`. See [Figure 19](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- write-memory 0x38001000 ../../image/1180_quadflash_helloWorld.bin
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 58236 (0xe37c) bytes to the target.
Successful generic response to command 'write-memory'
(1/1)100% Completed!
Successful generic response to command 'write-memory'
Response status = 0 (0x0) Success.
Wrote 58236 of 58236 bytes.
```

Figure 19. Write memory via BusPal

15. To confirm if the write is successful, read the image. Enter `blhost.exe -p com1 -b spi -- read-memory 0x38001000 0x400`. See [Figure 20](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -p com60 -b spi -- read-memory 0x38001000 0x80
Entering bit bang mode...
Entered BB mode
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
00 a0 00 87 00 00 00 00 00 00 01 90 00 00 00
00 a0 00 00 7c 43 00 00 00 b0 00 38 00 00 00
00 b0 00 38 00 00 00 00 13 02 00 00 00 00 00
5c ba e3 61 5a 42 c3 dd 7d 37 4e 19 5f 97 04 21
3a 0f bc 7e 98 f5 ea 9a 73 7d 36 4b 46 b8 94 05
da f8 24 f9 50 c0 43 a0 9a 88 8e 38 36 94 4a 2c
ea c2 6c 2e 4b 88 e7 e9 0e ef ff 19 72 53 84 a5
00 00 00 00 00 00 00 00 00 00 00 00 00 00
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 128 (0x80)
Read 128 of 128 bytes.
```

Figure 20. Read memory via BusPal

16. Switch the MIMXRT1180-EVK board to Internal boot mode. [Figure 21](#) shows the message displayed by UART.

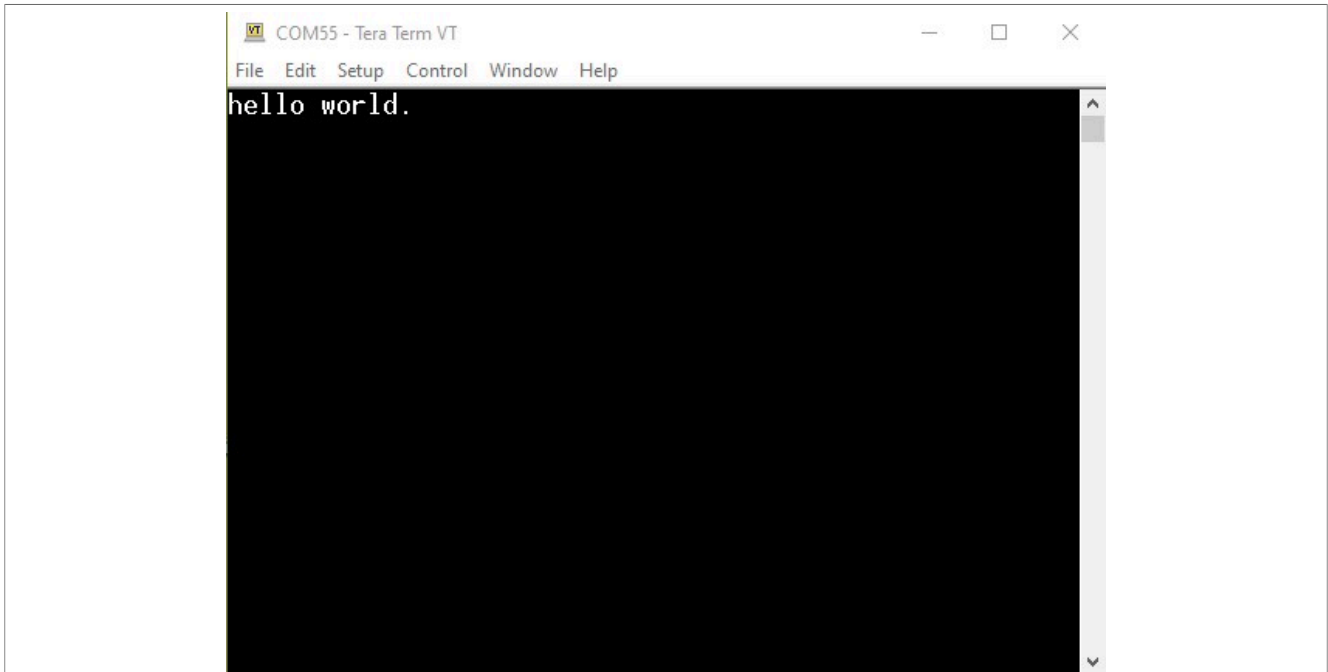


Figure 21. Boot success

4 Serial Downloader via LPSPI by USBSIO and MCU-Link Pro

After hardware preparation is completed, to start the Serial Downloader via LPSPI by USBSIO and MCU-Link Pro, perform the following steps:

1. Switch the MIMXRT1180-EVK board to Serial Downloader mode.

2. Run the command-line tool and go to the `blhost/bin/win` directory as shown in [Figure 7](#).
3. Enter `blhost.exe -u [<vid>,<pid>] -l spi -- get-property 1`. Check if the ROM can communicate successfully. See [Figure 22](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258487809 (0x4b030001)
Current Version = K3.0.1
```

Figure 22. Communicate with ROM via USBIO

4. Enter `blhost.exe -u [<vid>,<pid>] -l spi -- load-image <file>`. This step downloads and runs the flashloader. Ensure to add the container to the flashloader. SDK generates the flashloader. See [Figure 23](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- load-image ../../flashloader/flashloader_utility.bin
Ping responded in 1 attempt(s)
Inject command 'load-image'
Preparing to send 98868 (0x18234) bytes to the target.
(1/1)100% Completed!
Successful generic response to command 'load-image'
Response status = 0 (0x0) Success.
Response word 1 = 0 (0x0)
Wrote 98868 of 98868 bytes.
```

Figure 23. Download and run the flashloader via USBIO

5. Now, `blhost` can communicate with the flashloader. Enter `blhost.exe -u [<vid>,<pid>] -l spi -- get-property 1` again. Check if the flashloader can communicate successfully. See [Figure 24](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258424320 (0x4b020800)
Current Version = K2.8.0
```

Figure 24. Communicate with flashloader via USBIO

6. Set FlexSPI instance in TCM and enter `blhost.exe -u [<vid>,<pid>] -l spi -- fill-memory 0x20000000 4 0xcf900001`. See [Figure 25](#)

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- fill-memory 0x20000000 4 0xcf900001
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 25. Set FlexSPI instance in TCM via USBIO

7. Configure the memory and enter `blhost.exe -u [<vid>,<pid>] -l spi -- configure-memory 9 0x20000000`. See [Figure 26](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- configure-memory 9 0x20000000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 26. Configure memory via USB/SIO

8. Set NOR flash configuration parameters in TCM and enter `blhost.exe -u [<vid>,<pid>] -l spi -- fill-memory 0x20000000 4 0xc0000006`. See [Figure 27](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- fill-memory 0x20000000 4 0xc0000006
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 27. Set NOR flash configuration parameters via USB/SIO

9. Configure the memory and enter `blhost.exe -u [<vid>,<pid>] -l spi -- configure-memory 9 0x20000000`. See [Figure 28](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- configure-memory 9 0x20000000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 28. Configure memory via USB/SIO

10. Erase flash region and enter `blhost.exe -u [<vid>,<pid>] -l spi -- flash-erase-region 0x38000000 0x40000`. See [Figure 29](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- flash-erase-region 0x38000000 0x40000
Ping responded in 1 attempt(s)
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.
```

Figure 29. Erase flash region via USB/SIO

11. To confirm flash have been erased, read flash memory. Enter `blhost.exe -u [<vid>,<pid>] -l spi -- read-memory 0x38000000 16`. See [Figure 30](#). If successful, the result is similar to the one shown in [Figure 29](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- read-memory 0x38000000 16
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 16 (0x10)
Read 16 of 16 bytes.
```

Figure 30. Read flash via USB/SIO

12. Add FDCB XIP used and enter `blhost.exe -u [<vid>,<pid>] -l spi -- fill-memory 0x20000000 4 0xf000000f`. See [Figure 31](#). If successful, the result is similar to the one shown in [Figure 30](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- fill-memory 0x20000000 4 0xf000000f
Ping responded in 1 attempt(s)
Inject command 'fill-memory'
Successful generic response to command 'fill-memory'
Response status = 0 (0x0) Success.
```

Figure 31. Add FDCB via USBTIO

13. Configure the memory and enter `blhost.exe -u [<vid>,<pid>] -l spi -- configure-memory 9 0x20000000`. See [Figure 32](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- configure-memory 9 0x20000000
Ping responded in 1 attempt(s)
Inject command 'configure-memory'
Successful generic response to command 'configure-memory'
Response status = 0 (0x0) Success.
```

Figure 32. Configure memory via USBTIO

14. Write the image and enter `blhost.exe -u [<vid>,<pid>] -l spi -- write-memory <path>`. See [Figure 33](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- write-memory 0x38001000 ../../image/1180_quadflash_helloworld.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 58236 (0xe37c) bytes to the target.
Successful generic response to command 'write-memory'
(1/1)100% Completed!
Successful generic response to command 'write-memory'
Response status = 0 (0x0) Success.
Wrote 58236 of 58236 bytes.
```

Figure 33. Write image via USBTIO

15. To confirm if the write is successful, read the image. Enter `blhost.exe -u [<vid>,<pid>] -l spi -- read-memory 0x38001000 0x100`. See [Figure 34](#).

```
C:\blhost_2.6.7\bin\win>blhost.exe -u 0x1fc9,0x0143 -l spi -- read-memory 0x38001000 0x80
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Successful response to command 'read-memory'
00 a0 00 87 00 00 00 00 00 00 01 90 00 00 00
00 a0 00 00 7c 43 00 00 00 b0 00 38 00 00 00 00
00 b0 00 38 00 00 00 00 00 13 02 00 00 00 00 00 00
5c ba e3 61 5a 42 c3 dd 7d 37 4e 19 5f 97 04 21
3a 0f bc 7e 98 f5 ea 9a 73 7d 36 4b 46 b8 94 05
da f8 24 f9 50 c0 43 a0 9a 88 8e 38 36 94 4a 2c
ea c2 6c 2e 4b 88 e7 e9 0e ef ff 19 72 53 84 a5
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(1/1)100% Completed!
Successful generic response to command 'read-memory'
Response status = 0 (0x0) Success.
Response word 1 = 128 (0x80)
Read 128 of 128 bytes.
```

Figure 34. Read image via USBTIO

16. Switch MIMXRT1180-EVK board to Internal boot mode. [Figure 35](#) shows the message displayed by UART.

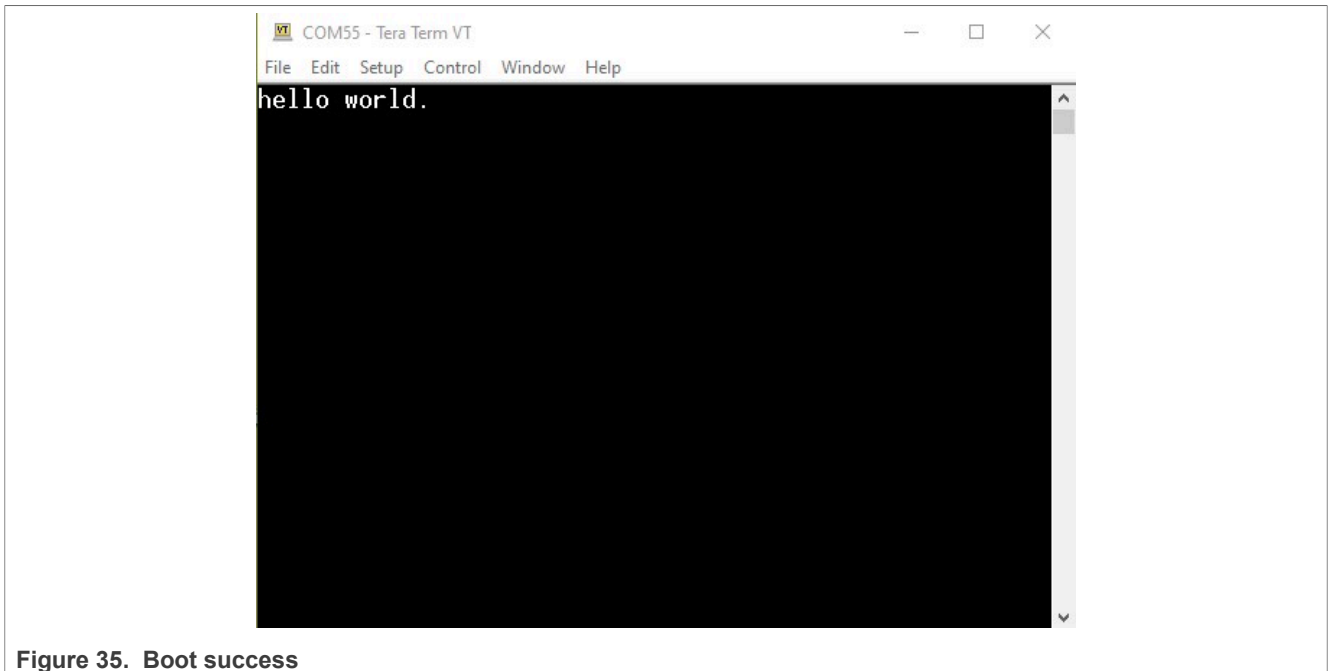


Figure 35. Boot success

5 Note about the source code in the document

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

Copyright 2024 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6 Revision history

[Table 9](#) summarizes the revisions to this document.

Table 9. Revision history

Document ID	Release date	Description
AN14393 v.1.0	4 September 2024	Initial public release

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.

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, μ Vision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) 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.

J-Link — is a trademark of SEGGER Microcontroller GmbH.

Contents

1	Introduction	2
1.1	Serial Downloader	2
1.2	MCU-Link Pro	2
1.3	Blhost	3
1.4	BusPal	4
1.5	USBSIO	4
2	Quick start	4
2.1	Prepare MCU-Link Pro	4
2.2	Prepare MIMXRT1010-EVK board	5
2.3	Prepare MIMXRT1180-EVK board	6
2.4	Serial downloader via LPSPI and BusPal hardware preparation	7
2.5	Serial downloader via LPSPI and USBSIO hardware preparation	8
3	Serial Downloader via LPSPI by BusPal and MIMXRT1010-EVK	8
4	Serial Downloader via LPSPI by USBSIO and MCU-Link Pro	12
5	Note about the source code in the document	16
6	Revision history	16
	Legal information	18

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.