AN12689 Porting USB Bootloader from LPC54018 to LPC51U68

Rev. 0 - 02/2020

by: NXP Semiconductors

Application Note

1 Introduction

There are several means for firmware upgrade, including the bootloader, hardware debugger (such as CMSIS-DAP, J-Link, and ULink), and third-party programmer (such as Flash Magic). Efficient and fast flash programming in the production flow is an important part of cost-effective manufacturing and field servicing of MCUs. From the firmware upgrade tools mentioned above, the hardware debugger is suitable for product development. The third-party programmer and bootloader are suitable for mass production. For NXP MCUs, Flash Magic only supports SWD, Ethernet, and UART (except for the USB).

Contents

| 1 Introduction1 |
|--|
| 2 Why to port USB bootloader1 |
| 3 System hardware setup and connection1 |
| 4 USB bootloader porting from LPC54018 to LPC51U683 |
| 5 References 27 |

The secondary bootloader in the NXP MCU bootloader provides multiple serial protocols (such as USB HID/MSC, UART, SPI, I2C, and CAN) for quick and easy programming through the entire product lifecycle. The USB interface is widely used due to high data transmission rate and driver support by various mainstream operating systems.

2 Why to port USB bootloader

As described on the official NXP website, a secondary bootloader is supported in the SDK for i.MXRT1050, LPC54018, LPC54016, and LPC54005. Therefore, the LPC51U68 USB bootloader can be ported from the SDK of the mentioned MCUs. This application note selects LPC54018 as a reference and describes how to port the USB bootloader from LPC54018 to LPC51U68.

| • | Secondary Bootloader |
|----------------------|------------------------------|
| | = I.MX RT1050 |
| | = LPC54018 |
| | = LPC54016 |
| | = LPC54005 |
| | |
| Figure 1. MCUs suppo | rted by secondary bootloader |

3 System hardware setup and connection

3.1 System hardware setup

LPCXpresso51U68 version Rev A is used as the evaluation board for USB bootloader porting. Make sure that jumper JP10 is fitted, so that JP10 connects the VBUS from the target USB connector J5 to the LPC51U68. The other jumpers (configured by default) are described in Table 1 and Figure 2.



Table 1. Board jumper setting

| Jumper number | Description | Status |
|----------------|--|--|
| J1, J2, J7, J8 | Expansion connectors | Open |
| J3 | External processor control header | Open |
| J4 | PMod [™] (SPI / I2C) bridge connector | Open |
| J5 | Target MCU power / USB device connector | Connect to PC via USB cable for firmware update |
| J6 | Link2 micro USB B-type connector | Connect to PC via USB cable for bootloader preprogramming |
| JP1 | LPC51U68 target SWD disable | Open |
| JP2 | Buffer power selection | Fit to Loc |
| JP3 | Isolate the Link2 debug probe (SPI bridge function) from the LPC51U68 target to prevent current leakage. | Open |
| JP4 | Disconnect all reset sources from the LPC51U68 device, except for the reset from the AP control port (J3). | Open |
| JP5 | Reduce the voltage sense resistance. | Open |
| JP6 | Measure the LPC51U68 current consumption. | Open |
| JP7 | Link2 (LPC43xx) force DFU boot | Open |
| JP8 | Tri-color LED anode voltage enable | Open |
| JP9 | Power supply voltage selection | Fit pin 2–3 to select 3.3 V power supply |
| JP10 | Target connector USB (J5) VBUS to LPC51U68 connection | Fit |
| P1 | 10-pin SWD connector | Open |
| P2 | LPC51U68 VDD current monitor Vsense measurement | Open |
| Р3 | FTDI serial header | Open |
| P4 | External ADC reference input | Open |



3.2 Connection

The connection shown in Figure 3 is a must for normal communication between the LPCXpresso51U68 board and PC. The USB cable connected to connector J6 is used to download the bootloader and the cable connected to connector J5 is used to update the user application firmware.



4 USB bootloader porting from LPC54018 to LPC51U68

4.1 Flash-resident USB bootloader porting

4.1.1 Creating new project and adding necessary files

Download SDK_2.6.0_LPCXpresso51U68 from the official NXP website. This application note uses the Keil IDE to
perform USB bootloader porting. Therefore, create a new empty project (as shown in Figure 4) by clicking "Project -> New
uVision Project" and selecting the project directory (as shown in Figure 5), assuming that the project directory is *lboards*.

| | C:\LocalData\NXP_work\10_LPC_Controller\LPC\LPC51L |
|--------------------|--|
| | File Edit View Project Flash Debug Peripheral |
| | New µVision Project |
| | New Multi-Project Workspace |
| | Open Project 2 |
| | Close Project |
| igure 1 Creating | u new Keil project |
| | |
| | Create New Project × |
| | ← → × ↑ 🔤 « Loca > LPC51U68_US8_boot > Ŏ Search LPC51U68_US8_bootlo P |
| | Organize - New folder Project directory |
| | LocalData Name Date modified Type bodyCheck Adatasheet No items match your search. No items match your search. DCIM Debug LPCS1U68_U5 Vector NXP_work 1_invetigate 2_test_qn90 3_C++_Qt Project name > |
| | File name: LPC51U68_USB_Bootloader V |
| | Save as type: Project Files (*.uvproj; *.uvprojx) |
| | Hide Folders Save Cancel |
| igure 5. Selecting | g project directory and names |

- Download SDK_2.6.0_LPCXpresso54018 from the official NXP website.
- Copy the *mcu-boot* folder from the *SDK_2.6.0_LPCXpresso54018\middleware* folder to the *SDK_2.6.0_LPCXpresso51U68\middleware* folder.
- Modify the "LPC54018" string in the *external_memory_property_map_LPC54018.c*, *hardware_init_LPC54018.c*, *memory_map_LPC54018.c*, and *peripherals_LPC54018.c* files in the *lmiddlewarelmcu-bootltargetslLPC51U68lsrcl* folder to "LPC51U68".
- Create a source file named *internalFlashAPI.c* (other names are also OK) and a header file named *internalFlashAPI.h* (other names are also OK) for the LPC51U68 on-chip flash to provide operations such as erase, read, write, and execute in the *ImiddlewareImcu-bootlsrcImemoryIsrc* folder.

- Copy the *pin_mux.c, pin_mux.h, clock_config.c,* and *clock_config.h* files from the *lboardsl/pcxpresso51u68lusb_examples lusb_device_cdc_vcomlfreertos* folder to the *lboards* folder.
- Copy the files shown in Table 2 from the *lboardsl/pcxpresso51u68lusb_exampleslusb_device_msc_ramdisklfreertos* folder to the *lboards* folder.

| Number | File |
|--------|-------------------------|
| 1 | usb_device_ch9.c |
| 2 | usb_device_ch9.h |
| 3 | usb_device_class.c |
| 4 | usb_device_class.h |
| 5 | usb_device_config.h |
| 6 | usb_device_descriptor.c |
| 7 | usb_device_descriptor.h |
| 8 | usb_device_msc.c |
| 9 | usb_device_msc.h |
| 10 | usb_device_msc_ufi.c |
| 11 | usb_device_msc_ufi.h |

Table 2. USB device files

• Copy the *usb_device_hid.c* and *usb_device_hid.h* files from the *lboardsllpcxpresso51u68lusb_examples lusb_device_hid_genericlfreertos* folder to the *lboards* folder.

• Add the groups and files shown in Table 3 to the project created in the first step.

Table 3. Groups and files

| Group | File |
|-------------------------|---|
| device | \devices\LPC51U68\fsl_device_registers.h |
| | \devices\LPC51U68\LPC51U68.h |
| | \devices\LPC51U68\LPC51U68_features.h |
| | \devices\LPC51U68\system_LPC51U68.c |
| | \devices\LPC51U68\system_LPC51U68.h |
| startup | \devices\LPC51U68\arm\startup_LPC51U68.s |
| source-usb-bm_composite | \middleware\mcuboot\src\bm_usb |
| | \bootloader_hid_report_ids.h |
| | \middleware\mcu-boot\src\bm_usb\composite.c |
| | \middleware\mcu-boot\src\bm_usb\composite.h |
| | \middleware\mcu-boot\src\bm_usb\fat_directory_entry.h |
| | \middleware\mcu-boot\src\bm_usb\hid_bootloader.c |
| | \middleware\mcu-boot\src\bm_usb\hid_bootloader.h |
| | \middleware\mcu-boot\src\bm_usb\msc_disk.c |

| Group | File |
|-----------------------|--|
| | \middleware\mcu-boot\src\bm_usb\msc_disk.h |
| | \middleware\mcu-boot\src\bm_usb\usb_descriptor.c |
| | \middleware\mcu-boot\src\bm_usb\usb_descriptor.h |
| | \middleware\mcu-boot\targets\LPC51U68\src \usb_device_config.h |
| source-autobaud | \middleware\mcu-boot\src\autobaud\autobaud.h |
| | \middleware\mcu-boot\src\autobaud\src\autobaud_irq.c |
| source-bootloader | \middleware\mcu-boot\src\bootloader\bl_app_crc_check.h |
| | \middleware\mcu-boot\src\bootloader\bl_command.h |
| | \middleware\mcu-boot\src\bootloader\bl_context.h |
| | \middleware\mcu-boot\src\bootloader\bl_irq_common.h |
| | \middleware\mcu-boot\src\bootloader\bl_peripheral.h |
| | \middleware\mcu-boot\src\bootloader \bl_peripheral_interface.h |
| | \middleware\mcu-boot\src\bootloader\bl_shutdown_cleanup.h |
| | \middleware\mcu-boot\src\bootloader\bl_user_entry.h |
| | \middleware\mcu-boot\src\bootloader\bl_version.h |
| | \middleware\mcu-boot\src\bootloader\bootloader.h |
| source-bootloader-src | \middleware\mcu-boot\src\bootloader\src\bl_app_crc_check.c |
| | \middleware\mcu-boot\src\bootloader\src\bl_command.c |
| | \middleware\mcu-boot\src\bootloader\src\bl_context.c |
| | <pre>\middleware\mcu-boot\src\bootloader\src \bl exception handler.c</pre> |
| | /middleware\mcu-boot\src\bootloader\src\bl_main.c |
| | \middleware\mcu-boot\src\bootloader\src\bl_misc.c |
| | \middleware\mcu-boot\src\bootloader\src \bl_shutdown_cleanup.c |
| | \middleware\mcu-boot\src\bootloader\src\bl_tree_root.c |
| | \middleware\mcu-boot\src\bootloader\src\bl_user_entry.c |
| | \middleware\mcu-boot\src\bootloader\src \usb_hid_msc_peripheral_interface.c |
| source-crc | \middleware\mcu-boot\src\crc\crc16.h |
| | \middleware\mcu-boot\src\crc\crc32.h |
| | \middleware\mcu-boot\src\crc\src\crc16.c |

| Group | File |
|----------------------|---|
| | \middleware\mcu-boot\src\crc\src\crc32.c |
| source-include | \middleware\mcu-boot\src\include\bootloader_common.h |
| | \middleware\mcu-boot\src\include\bootloader_core.h |
| source-memory | \middleware\mcu-boot\src\memory\memory.h |
| source-memory-src | \middleware\mcu-boot\src\memory\src\device_memory.c |
| | \middleware\mcu-boot\src\memory\src\device_memory.h |
| | \middleware\mcu-boot\src\memory\src\memory.c |
| | \middleware\mcu-boot\src\memory\src\normal_memory.c |
| | \middleware\mcu-boot\src\memory\src\normal_memory.h |
| | \middleware\mcu-boot\src\memory\src\pattern_fill.h |
| | \middleware\mcu-boot\src\memory\src\pattern_fill.s |
| | \middleware\mcu-boot\src\memory\src\sram_init.h |
| | \middleware\mcu-boot\src\memory\src\sram_init_lpc.c |
| | \middleware\mcu-boot\src\memory\src\internalFlashAPI.c |
| | \middleware\mcu-boot\src\memory\src\internalFlashAPI.h |
| source-packet | \middleware\mcu-boot\src\packet\command_packet.h |
| | \middleware\mcu-boot\src\packet\serial_packet.h |
| source-packet-src | \middleware\mcu-boot\src\packet\src\serial_packet.c |
| source-property | \middleware\mcu-boot\src\property\property.h |
| source-sbloader | \middleware\mcu-boot\src\sbloader\sb_file_format.h |
| | \middleware\mcu-boot\src\sbloader\sbloader.h |
| source-sbloader-src | \middleware\mcu-boot\src\sbloader\src\sbloader.c |
| source-utilities | \middleware\mcu-boot\src\utilities\fsl_assert.h |
| | \middleware\mcu-boot\src\utilities\fsl_rtos_abstraction.h |
| | \middleware\mcu-boot\src\utilities\vector_table_info.h |
| source-utilities-src | \middleware\mcu-boot\src\utilities\src\fsl_assert.c |
| | \middleware\mcu-boot\src\utilities\src\fsl_rtos_abstraction.c |
| source-drivers | \middleware\mcu-boot\src\drivers\smc\smc.h |
| source-property-src | \middleware\mcu-boot\src\property\src\property_lpc.c |
| LPC51U68 | \middleware\mcu-boot\targets\LPC51U68\src \bootloader_config.h |
| | \boards\clock_config.c |

| Group | File |
|-----------------------------|---|
| | \middleware\mcu-boot\targets\LPC51U68\src \external_memory_property_map_LPC51U68.c |
| | \middleware\mcu-boot\targets\LPC51U68\src \hardware_init_LPC51U68.c |
| | \middleware\mcu-boot\targets\LPC51U68\src \memory_map_LPC51U68.c |
| | \middleware\mcu-boot\targets\LPC51U68\src \peripherals_LPC51U68.c |
| | \middleware\mcu-boot\targets\LPC51U68\src \peripherals_pinmux.h |
| | \middleware\mcu-boot\targets\LPC51U68\src\target_config.h |
| | \middleware\mcu-boot\targets\common\src \pinmux_utility_lpc.c |
| | \boards\pin_mux.c |
| | \boards\pin_mux.h |
| usb-device-class-hid | \boards\usb_device_hid.c |
| | \boards\usb_device_hid.h |
| usb-device-class-msc | \boards\usb_device_msc.c |
| | \boards\usb_device_msc.h |
| | \boards\usb_device_msc_ufi.c |
| | \boards\usb_device_msc_ufi.h |
| usb-device-source | \boards\usb_device_ch9.c |
| | \boards\usb_device_ch9.h |
| | \middleware\usb\device\usb_device_dci.c |
| | \middleware\usb\device\usb_device_dci.h |
| usb-device-class | \boards\usb_device_class.c |
| | \boards\usb_device_class.h |
| usb-device-source-lpcip3511 | \middleware\usb\device\usb_device_lpcip3511.c |
| | \middleware\usb\device\usb_device_lpcip3511.h |
| usb-include | \middleware\usb\include\usb.h |
| | \middleware\usb\include\usb_misc.h |
| | \middleware\usb\include\usb_spec.h |
| osa | \middleware\usb\osa\usb_osa.h |
| | \middleware\usb\osa\usb_osa_bm.c |

| Group | File |
|-----------------------------|---|
| | \middleware\usb\osa\usb_osa_bm.h |
| usb-device-include | \middleware\usb\device\usb_device.h |
| drivers | devices\LPC51U68\drivers\fsl_clock.c |
| | devices\LPC51U68\drivers\fsl_clock.h |
| | devices\LPC51U68\drivers\fsl_common.c |
| | devices\LPC51U68\drivers\fsl_common.h |
| | devices\LPC51U68\drivers\fsl_crc.c |
| | devices\LPC51U68\drivers\fsl_crc.h |
| | devices\LPC51U68\drivers\fsl_inputmux.c |
| | devices\LPC51U68\drivers\fsl_inputmux.h |
| | devices\LPC51U68\drivers\fsl_inputmux_connections.h |
| | devices\LPC51U68\drivers\fsl_iocon.h |
| | devices\LPC51U68\drivers\fsl_power.c |
| | devices\LPC51U68\drivers\fsl_power.h |
| | devices\LPC51U68\drivers\fsl_reset.c |
| | devices\LPC51U68\drivers\fsl_reset.h |
| | devices\LPC51U68\drivers\fsl_iap.c |
| | devices\LPC51U68\drivers\fsl_iap.h |
| | devices\LPC51U68\drivers\fsl_gpio.c |
| | devices\LPC51U68\drivers\fsl_gpio.h |
| source-drivers-microseconds | \middleware\mcu-boot\src\drivers\microseconds \microseconds.h |
| | \middleware\mcu-boot\src\drivers\microseconds\src \microseconds_sysclk.c |

4.1.2 Configuring Keil IDE

4.1.2.1 Selecting MCU part number

Click the "Device" tab and select the MCU part number, as shown in Figure 6.

| | | | | 1 |
|--|--|---|---|---|
| Vendor: NXP Device: LPC51U68JB Toolset: ARM | D64 | Software Pack Pack: NXP.LPC51L URL: http://mcuap | 68_DFP.12.0.0 esso rxp.com/oneis_pack/repc | |
| ARM ARM ARM KEx Serie Stepestor Control Control ARM Control Contre | es is in the second sec | e LPC51U68 are ARM Cortex-M0- bedded applications. These device AM. 256 KB on chip flash, full-spe i, three general-purpose timers, on ry other capabilities (SCTimer/PW kMuBritate Timer (MRT), a Win WDT), eight flexible serial commu- ich can be a USART, SPIs, or I2C ps ADC, and a temperature senso processor available on some devic sy to-use 32-bit core which is code e. The Cortex-M0+ coprocessor o formance with a simple instruction | based microcontrollers for is include 95 KB of on-chip ed USB device interface, and e versatile timer with PWM and M), one RTC/alam timer, one dowed Watchdog Timer vication peripherals (each of interface), and one 12-bit 5.0 . The ARM Costex-M0+ en is an energy-efficient and - and tool-compatible with the flers up to 100 MHz set and reduced code size. | * |
| | OK | Count Delade | Halo | |

4.1.2.2 Generating binary executable file

Click the "User" tab and carry out the configuration (as shown in Figure 7) to generate a binary executable file when the project is compiled.

| no use I under Looper Looped | In a continue i muco i presekti concepti | | |
|---|---|--------|------------|
| Command Items | User Command | | Stop on E |
| Run #1 | | 1 | Not Specif |
| - Run #2 | | 100 | Not Specif |
| Before Build/Rebuild | | | |
| - C Run #1 | | 1 | Not Specif |
| Run #2 | | 2 | Not Specif |
| After Build/Rebuild | | | |
| ✓ Run #1 | SK/ARM/BIN/ElfDwT.exe IL BASEADDRESS(0x0) | | Not Specif |
| ✓ Run #2 | SK\ARM\ARMCC\bin\fromelf.exebinoutput=@L.bin !L | \geq | Not Specif |
| ✓ ✓ | ☐ Start Debugging | | <u> </u> |
| | | | |

4.1.2.3 Seting preprocessor symbols and header file path

Click the "C/C++" tab and use the "_DEBUG=1,DEBUG, CPU_LPC51U68JBD64, USB_STACK_BM, USB_STACK_USE_DEDICATED_RAM=1, BL_TARGET_FLASH" string as the preprocessor symbols (as shown in Figure 8).

| Options for Target 'LPC | C51U68_USB_Bootloader | × |
|---|---|----|
| Device Target Output | Listing User C/C++ Asm Linker Debug Utilities | |
| Preprocessor Symbols Define:DEBUG+1 Undefine: | .DEBUG, CPU_LPC51U68JBD64. USB_STACK_BM. USB_STACK_USE_DEDICATED_R/ | ĀĀ |
| 8. Preprocessor symbols setting |] | |

Click the "C/C++" tab and add the header file search path. Click the "Include Paths" button (as shown in Figure 9) and enter the "Folder Setup" interface to add the header file search path (as shown in Figure 10). Add the header file directories according to Table 4.

| Device Targ | et Output Listing User | C/C+ | * Aam Linker | Debug Utilit | ies | | |
|-------------------|------------------------------|------------|----------------------|----------------|--------------|------------------|-------|
| Preproces | sor Symbols | | | | | | |
| Define: | _DEBUG=1,DEBUG, CPU | LPC51L | J68JBD64, USB_ST | ACK_BM, USE | STACK_U | SE_DEDICATED | RAN |
| Undefine | <u> </u> | | | | | | |
| Language | / Code Generation | | | | | | |
| F Execut | te-only Code | Г | Strict ANSI C | | Warnings: | Al Warnings | - |
| Optimizatio | n: Level 0 (-00) 👻 | F | Enum Container alv | vays int | | Thumb Mod | ie . |
| C Optimiz | te for Time | Г | Plain Char is Signer | 4 | | T No Auto Inc | dudes |
| F Split La | and Store Multiple | Г | Read-Only Position | independent | | C99 Mode | |
| 🔽 One E | LF Section per Function | Г | Read-Write Position | n Independent | | GNU extens | sions |
| Include | | | DCENICO NA | AND DOES | 1100 44 | 1.11 | |
| Paths | | es;vev | ices (LPC51068; /0 | evices (LPC) | U68 drivers, | vniodieware vni | a |
| Misc Controls | | | | | | / | |
| Compiler | -c99 -cou c -cou Cotex | -M0+ -0 -6 | 00-accarinterwork | -solt sections | -I MSIS | /include -l | - |
| control string | ./devices -1 ./devices/LF | C51U68 | -I/devices/LPC51 | U68/drivers | /middlewan | e/mcu-boot/src - | 1 |
| | | Includ | de Paths button | | | | |
| | | OK | Consul | Defaile | 211 | | Help |

| Setur Compiler Include Pathe: | 1 * * 4 |
|---|---------|
| \CMSIS\Include \devices .\devices\LPC51U68 \devices\LPC51U68 .\devices\LPC51U68\drivers \middleware\mcu-boot\src\arc .\middleware\mcu-boot\src\crc .\middleware\mcu-boot\src\crc .\middleware\mcu-boot\src\bootloader .\middleware\mcu-boot\src\bm_usb .\middleware\mcu-boot\src\tritties .\middleware\mcu-boot\src\tritties .\middleware\mcu-boot\src\tritties .\middleware\mcu-boot\src\tritties .\middleware\mcu-boot\src\trivers\microseconds .\middleware\mcu-boot\src\trivers\mcc .\middleware\mcu-boot\src\trivers\mcc .\middleware\mcu-boot\src\trivers\mcc .\middleware\mcu-boot\src\trivers\mcc .\middleware\usb\device .\middleware\usb\device .\middleware\usb\csa | |
| OK Cancel | |

Table 4. Header file directories

| Number | Header file directory |
|--------|---|
| 1 | \CMSIS\Include |
| 2 | \boards |
| 3 | \devices |
| 4 | \devices\LPC51U68 |
| 5 | \devices\LPC51U68\drivers |
| 6 | \middleware\mcu-boot\src |
| 7 | \middleware\mcu-boot\src\autobaud |
| 8 | \middleware\mcu-boot\src\crc |
| 9 | \middleware\mcu-boot\src\bootloader |
| 10 | \middleware\mcu-boot\src\bm_usb |
| 11 | \middleware\mcu-boot\src\include |
| 12 | \middleware\mcu-boot\src\startup |
| 13 | \middleware\mcu-boot\src\utilities |
| 14 | \middleware\mcu-boot\src\drivers |
| 15 | \middleware\mcu-boot\src\drivers\microseconds |
| 16 | \middleware\mcu-boot\src\drivers\smc |
| 17 | \middleware\mcu-boot\src\drivers\lpc_gpio |
| 18 | \middleware\usb\device |

Table 4. Header file directories (continued)

| Number | Header file directory |
|--------|---|
| 19 | \middleware\usb\include |
| 20 | \middleware\usb\osa |
| 21 | \middleware\mcu-boot\targets\LPC51U68\src |

4.1.2.4 Assembly setting

Click the "Asm" tab and perform the assembly language settings (as shown in Figure 11).

| Conditio | nal Assembly Control Symbols | |
|------------------------------|---|-----------------------|
| Defin | e: DEBUGCC_ARM.KEIL | |
| Undefin | e. | |
| Langua | ge / Code Generation | |
| ☐ Rea | d-Only Position Independent T Split Load and Store Multiple | |
| □ □ Rea | d-Write Postion Independent | |
| IT No V | Namings I No Auto Includes | |
| Include | | |
| Control | c -cpreproc | |
| Assemble contro string | r -cpu Cottex M0+ g -apcs+interwork -cpreproc H -/.NTE_LPC51U68_US8_Bootloader g | • |
| | | |

4.1.2.5 Linker setting

Uncheck the "Use Memory Layout from Target Dialog" checkbox to use the scatter file to assign the memory area for the binary executable file and carry out the miscellaneous control settings shown in Figure 12.

USB bootloader porting from LPC54018 to LPC51U68

| Use Men Use Men Make Dont Repo | get Output Listing User C/C++ Aam Linker Debug Utilities mory Layout from Target Dialog X/O Base: |
|--|---|
| disable f Scatter File | Memory Layout from Target Dialog means using Scatter File below Press this button and edit scatter file VObjects/LPC51U68_US8_Bootloader.sct |
| Mac controls | _/devices/LPC51U68/arm/kel_jb_power.jb -remove -datacompressor off -feedback_type=unused.noiw -predefine="-0_ram_vector_table_=0" |
| | -cpu Cottex-M0+ -scatter "/RTE/Device/LPC51U68/BD64/LPC51U68_flash.scf" |
| Linker control string | × |

Click the "Edit" button and edit the scatter file shown in Figure 14. The memory setting in the scatter file is derived from the memory map shown in Figure 13.

USB bootloader porting from LPC54018 to LPC51U68

| | Memory space | | | APB peripherals | |
|---|---------------------------------|---|----------------------|---------------------------------------|-------------|
| Г | (reserved) | 0xFFFF FFFF | ſ | ADC | 0x400A 1000 |
| | private peripheral bus(1) | 0xE010 0000 | | (reserved) | 0x4000 0000 |
| | (reserved) | 0xE000 0000 | | ISP-AP interface | 0x4009 0000 |
| | | 0x4400 0000 | | (reserved) | 0x4009 9000 |
| | (reserved) | 0. 1000.0000 | | Flexcomm Interface 7 | 0x4009 8000 |
| | (reserved) | 0x4200 0000 | | Flexcomm Interface 6 | 0x40097000 |
| Г Г | AHB | 0x400A 1000 |] | Flexcomm Interface 5 | 0x4009 6000 |
| | peripherals | 0x4008 0000 0x4006 0000 0x4004 0000 0x4002 0000 0x4002 0000 memory map figure | 1 | CRC engine | 0x4009 5000 |
| | (reserved) | | | (reserved) | 0x4009 1000 |
| | Asynchronous APB peripherals | | High Speed GPIO | 0x4009 0000 | |
| | APB peripherals on | | (reserved) | 0x4008 C000 | |
| | APB bridge 1 | | Flexcomm Interface 4 | 0x4008 8000 | |
| E E | APB peripherals on | | 1.1.1 | Flexcomm Interface 3 | 0×4008 9000 |
| - | APB bridge 0 | 0x4000 0000 | | Flexcomm Interface 2 | 0x4008 8000 |
| - | (reserved) | 0x2400 0000 | | Flexcomm Interface 1 | 0x40087000 |
| | (reserved) | | | Flexcomm Interface 0 SCTimer / PWM | 0x4008 6000 |
| | (reserved) | 0x2200 0000 | | FS USB device | 0x4008 5000 |
| | (reserved) | 0x2002 8000 | | (reserved) | 0x40084000 |
| | And Bally Cold | 0x2002 0000 | | DMA controller | 0x4008 2000 |
| | (reserved) | | L | (reserved) | 0x4008 1000 |
| | SRAM0 | 0x2001 0000 | ite meene | | |
| L | (64 KB) | read-wi | ite memo | bry area | |
| | (reserved) | 0x0400 8000 | | | |
| Γ | SRAMX | 0.0100 0000 | | | |
| | (32 KB) | 0x0400 0000 | | | |
| | (reserved) | 0x0300 8000 | | | |
| L | Boot ROM | 0~0300.0000 | | | |
| l T | (reserved) | 0.00000000 | | | |
| r i i i i i i i i i i i i i i i i i i i | Flash memory | 0x0004 0000 | ly memo | ny area | |
| L | 256 KB | 0x0000 0000 | ily memo | iy alea | |
| | 1 | - | | | |
| | | 0x0 | 000 0000 | | |
| | active inte | errupt vectors 0x0 | 0000 0000 | | |
| Figure 13. LPC51U68 | 8 memory map | | | | |
| | | | | | |

```
11 armcc -E
            #if (defined(__ram_vector_table__))
              #define _____ram_vector_table_size___
                                                       0x00000400
             felse.
               #define __ram_vector_table_size__
                                                      0x00000000
             #endif
                                                      0x00000000
            #define m_interrupts_start
            #define m interrupts size
                                                      0x00000400
             #define m text start
                                                      0x00000400
            #define m text size
                                                       0x0003FC00
                                                      0x20000000
            #define m_interrupts_ram_start
            #define m interrupts ram size
                                                       ___ram_vector_table_size_
                                                       (m_interrupts_ram_start + m_interrupts_ram_size)
            #define m_data_start
            ∉define m data size
                                                       (0x00010000 - m interrupts ram size)
             /* Sizes */
            #if (defined(__stack_size__))
              #define Stack Size
                                                      __stack_size
             felse
              #define Stack Size
                                                      0x0400
             #endif
            #if (defined(__heap_size__))
                                                       __heap_size
              #define Heap_Size
             felse
                                                      0
              #define Heap Size
             LR_m_text m_interrupts_start m_text_start+m_text_size-m_interrupts_start
             {:load region size_region
              VECTOR ROM m_interrupts_start m_interrupts_size
              (;load address = execution address

    (RESET, +FIRST)

              3
              ER_m_text m_text_start FIXED m_text_size
              (:load address = execution address
* (InRoot$$Sections)
                ANY (+RO)
             #if (defined(__ram_vector_table__))
               VECTOR RAM m_interrupts_ram_start EMPTY m_interrupts_ram_size
             felse
               VECTOR RAM m_interrupts_start EMPTY 0
             #endif
               RW m_data m_data_start m_data_size-Stack_Size-Heap_Size
               (:RW data
                .ANY (+RW +21)
               3
               ARM LIB HEAP +0 EMPTY Heap Size
               (:Heap region growing up
              ARM_LIB_STACK m_data_start+m_data_size EMPTY -Stack_Size
               {;Stack region growing down
             3
Figure 14. Scatter file
```

4.1.2.6 Debug setting

Click the "Debug" tab and select the simulator according to the actual situation. This application note uses the on-board J-Link debugger of the LPCXpresso51U68 board. Check the "Load Application at Startup" and "Run to main()" checkboxes (as shown in Figure 15).

| C Use Simulator with restrictions Settings | Use: J-LINK / J-TRACE Cortex Settings |
|--|---|
| Load Application at Statup Initialization File: Edit. Restore Debug Session Settings Preakpoints Watch Windows & Performance Analyzer Watch Windows & Performance Analyzer Memory Display System Viewer CPU DLL: Parameter: SARMCM3.DLL Dalog DLL: Parameter: | Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Construction of the startup Image: Constartup Imag |
| DARMCM1.DLL PCM0+ | TARMCM1.DLL PCM0+ |
| Warn If outdated Executable is loaded Manage Component | Warn if outdated Executable is loaded |

4.1.3 Compiling and fixing errors

Errors are generated when compiling the project. To fix these errors, make these modifications:

Open ImiddlewareImcu-bootItargetsILPC51U68IsrcIbootIoader_config.h and remove this macro:

#define FSL FEATURE SYSCON FLASH PAGE SIZE BYTES (0)

- Open ImiddlewareImcu-bootIsrclcrcIsrclcrc16.c and modify the header file include command from #include "lpc_crc/ fsl crc.h" to #include "fsl crc.h".
- Open *ImiddlewareImcu-bootIsrclcrclsrclcrc32.c* and modify the header file include command from #include "lpc_crc/
 fsl_crc.h" to #include "fsl_crc.h".
- Open *middlewaremcu-bootlsrclcrclsrclcrc16.c* and modify the "crc16_onfi_update" function. For the specific function definition, see the project code that matches the documentation.
- Open *ImiddlewareImcu-bootItargetsILPC51U68IsrcIhardware_init_LPC51U68.c* and modify the "init_hardware" function. For the specific function definition, see the project code that matches the documentation.
- Open ImiddlewareImcu-bootItargetsILPC51U68IsrcIhardware_init_LPC51U68.c and include the pin_mux.h header file:

#include "pin mux.h"

- Open *ImiddlewareImcu-bootItargetsILPC51U68IsrcIhardware_init_LPC51U68.c* and remove the "spifi_clock_gate", "spifi_source_clock", and "spifi_iomux_config" functions.
- Open *ImiddlewareImcu-bootItargetsILPC51U68IsrcIhardware_init_LPC51U68.c* and modify the "usb_clock_init" function. For the specific function definition, see the project code that matches the documentation.

Porting USB Bootloader from LPC54018 to LPC51U68, Rev. 0, 02/2020

 Open ImiddlewareImcu-bootItargetsILPC51U68IsrcIbootIoader_config.h and modify the "BL_CONFIG_USB_HID", "BL_CONFIG_HS_USB_HID", BL_CONFIG_FLEXCOMM_USART_0", "BL_CONFIG_FLEXCOMM_I2C_2", "BL_CONFIG_FLEXCOMM_SPI_9", and "BL_FEATURE_SPIFI_NOR_MODULE" macros:

```
#define BL_CONFIG_USB_HID (1)
#define BL_CONFIG_HS_USB_HID (0)
#define BL_FEATURE_SPIFI_NOR_MODULE (0)
#define BL_CONFIG_FLEXCOMM_USART_0 (0)
#define BL_CONFIG_FLEXCOMM_I2C_2 (0)
#define BL_CONFIG_FLEXCOMM_SPI_9 (0)
```

- Open *boardslclock_config.c* and add the definition of the "configure_clocks" function. For the specific function definition, see the project code that matches the documentation.
- Open IboardsIclock_config.c and include three header files and one macro definition:

```
#include "bootloader_common.h"
#include "property/property.h"
#include "bootloader/bl_context.h"
#define BOOTLOADER CLOCK FREQ 48000000U
```

• Open *ImiddlewareImcu-bootIsrcIbootloaderIsrcIbI_shutdown_cleanup.c* and add the definition of the "init_interrupts" function. For the specific function definition, see the project code that matches the documentation.

4.1.4 Testing USB communication between LPCXpresso51U68 and PC

To test the USB communication between the LPCXpresso51U68 board and PC, follow these steps:

- · Compile the project.
- Fit the JP10 jumper on the LPCXpresso51U68 board to connect the USB (J5) VBUS to the LPC51U68.
- Use a USB cable to connect the J6 header to the PC and download the project executable binary file to the LPC51U68 flash area.
- Use a USB cable to connect the J5 header to the PC.
- · Power up and reset the LPCXpresso51U68 board.
- Run ImiddlewareImcu-bootIbinIToolsIbIhostIwinIbIhost.exe and enter the blhost command (as shown in Figure 16). If the
 following command feedback appears, it indicates that the USB bootloader porting is implemented successfully. The MCU
 Bootloader host (blhost) User's Guide (document MCUBLHOSTUG) describes the detailed usage of the blhost commands
 mentioned below.



4.1.5 Flash API porting

The USB bootloader for LPC54018 updates the firmware to the on-chip Quad SPI Flash through the SPIFI interface. Unlike LPC54018, LPC51U68 contains a 256-KB on-chip flash, which can be accessed through the Flash In-Application Programming (IAP). Therefore, the LPC51U68 USB bootloader requires the Flash API which implements the read, write, and erase operations to update the firmware to the on-chip flash. The Flash API for LPC51U68 is implemented in the *internalFlashAPI.c* and *internalFlashAPI.h* files created in step 8 in Creating new project and adding necessary files.

4.1.6 Memory map porting

The memory maps for LPC54018 and LPC51U68 are different. Open *ImiddlewareImcu-bootItargetsILPC51U68Isrc Imemory_map_LPC51U68.c* and redefine the "g_memoryMap" structure, as shown in Figure 17.

```
memory_map_entry_t g_memoryMap[] =
{
    fif BL_FEATURE_USING_INTERNAL_PLASH
        { 0x00000000, 0x00003ffff, kMemoryNotExecutable | kMemoryType_FLASH, &g_internalFlashInterface },
        tendif
        { 0x20000000, 0x00007fff, kMemoryIsExecutable | kMemoryType_RAM, &g_normalMemoryInterface },
        { 0x20000000, 0x2000ffff, kMemoryNotExecutable | kMemoryType_RAM, &g_normalMemoryInterface },
        { 0x20000000, 0x2000ffff, kMemoryNotExecutable | kMemoryType_Device, &g_deviceMemoryInterface },
        { 0x40000000, 0x40001fff, kMemoryNotExecutable | kMemoryType_Device, &g_deviceMemoryInterface },
        { 0x40000000, 0x40003ffff, kMemoryNotExecutable | kMemoryType_Device, &g_deviceMemoryInterface },
        { 0x40000000, 0x40003ffff, kMemoryNotExecutable | kMemoryType_Device, &g_deviceMemoryInterface },
        { 0x4000000, 0x40000, 0x400a0fff, kMemoryNotExecutable | kMemoryType_Device, &g_deviceMemoryInterface },
        { 0x4000000, 0x400a0fff, kMemoryNotExecutable | kMemoryType_Device, &g_deviceMemoryInterface },
        { 0x40000000, 0x400a0fff, kMemoryNotExecutable | kMemoryType_Device, &g_deviceMemoryInterface },
        { 0x40000000, 0x400a0fff, kMemoryNotExecutable | kMemoryType_Device, &g_deviceMemoryInterface },
        { 0x40000000, 0x400a0fff, kMemoryNotExecutable } },
        { 0x40000000, 0x400a0fff, kMemoryNotExecutable },
        { 0x40000000, 0x400a0fff, kMemoryNotExecutable },
        { 0x40000000, 0x400a0fff, kMemoryNotExecutable },
        { 0x40000000, 0x400a0fff, kMe
```

Figure 17. LPC51U68 memory map structure

Open *ImiddlewareImcu-bootItargetsILPC51U68IsrcIbootIoader_config.h* and define the "BL_FEATURE_USING_INTERNAL_FLASH" macro:

#define BL_FEATURE_USING_INTERNAL_FLASH (1)

Open ImiddlewareImcu-bootIsrcImemoryImemory.h and export "g_internalFlashInterface":

```
#if BL_FEATURE_USING_INTERNAL_FLASH
extern const memory_region_interface_t g_internalFlashInterface;
#endif
```

4.1.7 Testing flash operations for flash-resident version

To test flash operations such as erase, read, write, and reset for a flash-resident version, follow these steps:

- Compile the project.
- Fit the JP10 jumper on the LPCXpresso51U68 board to connect the USB (J5) VBUS to the LPC51U68.
- Use a USB cable to connect the J6 header and PC and download the project executable binary file to the LPC51U68 flash area.
- Use a USB cable to connect the J5 header and PC.
- Power up and reset the LPCXpresso51U68 board.
- Run ImiddlewareImcu-bootIbinIToolsIbIhostIwinIbIhost.exe and enter the blhost commands shown in Figure 18, Figure 19, and Figure 20. If the following command feedbacks appear, it indicates that the bootloader porting of flash operations is implemented successfully. Note that gpio_led_output.bin is an example of a user application binary file (other user binary file names are also OK). Note that "0x80d5" in the execute command is a routine entry address which is a reset handler address in a vector table and it can be read from a binary file (as shown in Figure 21).



USB bootloader porting from LPC54018 to LPC51U68



4.2 RAM-resident USB bootloader porting

4.2.1 Configuring Keil IDE

The project configurations for the RAM-resident version are basically consistent with the flash-resident version. The following are the differences between the RAM-resident version and the flash-resident version.

4.2.1.1 Setting preprocessor symbols and header file path

Click the "C/C++" tab and use character string "_DEBUG=1,DEBUG, CPU_LPC51U68JBD64, USB_STACK_BM, USB_STACK_USE_DEDICATED_RAM=1, BL_TARGET_RAM" as the preprocessor symbols (see Figure 22).

USB bootloader porting from LPC54018 to LPC51U68

| Device Target Output Listing User | C/C++ Aam Linker Debug Utilities | 1 | |
|---|--|----------------------|--|
| Preprocessor Symbols | | | |
| Define: U_LPC51U68JBD64, USB_ | STACK_BM, USB_STACK_USE_DEDICATED_RAM- | 1. BL_TARGET_RAM | |
| Undefine: | | | |
| Language / Code Generation | | | |
| F Execute only Code | Strict ANSI C Warnings: | No Warnings 💌 | |
| Optimization: Level 3 (O3) | F Enum Container always int | Thursb Mode | |
| Coptimize for Time | Plain Char is Signed | T No Auto Includes | |
| Splt Load and Store Multiple | F Read-Only Postion Independent | I C99 Mode | |
| One ELF Section per Function | Read-Wite Postion Independent | IT GNU extensions | |
| Include | holude: \.\.\.\.devices: \.\.\.\.devices\LPC5 | 1068 | |
| Mac Controls | ag_suppress=1296,186 | | |
| Compiler control string | MICROLIB_g_W_03 -apcs-interwork -split_sectio 4 / / / / / /devices 4 / / / / /devices/LPC51U6 | n-1-1.7.7.4 A [4] | |
| | K Cancel Defaults | Help | |

4.2.1.2 Linker setting

The scatter file for the RAM-resident version is shown in Figure 23. It is different from that for the flash-resident version.

```
#! armcc -E
            /* Sizes */
            #if (defined(_stack_size_))
              #define Stack Size
                                                     stack size
            #else
              #define Stack Size
                                                     0x2000
            #endif
            #if (defined( heap size ))
              #define Heap_Size
                                                     __heap_size__
            felse
              #define Heap Size
                                                     0x1000
            [endif
            #define m_interrupts_start
                                                    0x20000000
            #define m_interrupts_size
                                                    0x00000400
            define m text start
                                                     0x20000400
                                                     0x0000FC00
            #define m text size
            LR_m_text m_text_start m_text_size {    ; load region size_region
              ER m text m text start m text size = Stack Size-Heap Size ; load address = execution address
                * (InRoot$$Sections)
               .ANY (+RO)
               .ANY (+RW +ZI)
              ARM LIB HEAP (ImageLimit(ER m text)) EMPTY Heap Size { ; Heap region growing up
              ARM LIB STACK (ImageLimit(ARM LIB HEAP) + Stack Size) EMPTY -Stack Size [ ; Stack region growing down
              1
            1
            LR_m_interrupts m_interrupts_start m_interrupts_size (
              VECTOR ROM m interrupts start m interrupts size { ; load address = execution address
                * (RESET, +FIRST)
              3
            Ł
            LR_m_interrupts_ram_m_interrupts_start_m_interrupts_size {
              VECTOR_RAM m_interrupts_start m_interrupts_size { ; load address = execution address
               .ANY (.m_interrupts_ram)
              1
            1
Figure 23. Scatter file
```

4.2.1.3 Debug setting

Click the "Debug" tab and select the simulator according to the actual situation. This application note uses the on-board J-Link debugger of the LPCXpresso51U68 board. Uncheck the "Load Application at Startup" checkbox (as shown in Figure 24). Click the "Edit" button in the "Initialization File" area and edit the *JLink Settings.ini* file (as shown in Figure 25).

| | Options for Target "LPC51U68_FlashLoader" | × |
|-------------------------|---|---|
| | Device Target Output Listing User C/C++ Asm | Linker Debug Utilities |
| | C Use Simulator with restrictions Settings | Use: J-LINK / J-TRACE Cortex Settings |
| | I Load Application at Startup I Run to main() Initialization File: | Load Application at Startup Run to main() Initialization File: |
| | Edt. | . JLink Settings ini Edit |
| | Restore Debug Session Settings | Restore Debug Session Settings |
| | Freakpoints Toolbox | I Reakpoints I Toobox |
| | Watch Windows & Performance Analyzer | Watch Windows |
| | IV mentary Display IV System viewer | V Mentury Liapiay V System viewer |
| | CPU DLL: Parameter: | Driver DLL: Parameter: |
| | SARMCM3.DLL | SARMCM3.DLL |
| | Dialog DLL: Parameter: | Dialog DLL: Parameter: |
| | DARMONT DEL JOCMU+ | TARMENT.DLL DCMU+ |
| | Wam if outdated Executable is loaded | Wam if outdated Executable is loaded |
| | Manage Component W | ewer Description Files |
| | OK Ca | ncel Defaults Help |
| | | |
| Figure 24. Debug se | etting | |
| | | |
| / | * Convergent (c) 2015 Ereescale Semico | nductor Inc |
| | * All rights reserved. | nuccor, me. |
| | · | |
| | */ | se |
| | | |
| F | SP = RDWORD(0x2000000): | // Setun Stack Pointer |
| | PC = _RDWORD(0x20000004); | // Setup Program Counter |
| | _WDWORD(0xE000ED08, 0x20000000); | <pre>// Setup Vector Table Offset Register</pre> |
| 3 | | |
| L | .OAD %L INCREMENTAL setup(); | // Download to RAM |
| g | , main | |
| Figure 25. Initializati | on file | |

4.2.1.4 Utilities setting

Click the "Utilities" tab and uncheck the "Update Target before Debugging" checkbox (as shown in Figure 26). Click the "Settings" button and enter the "Cortex JLink/JTrace Target Driver Setup" interface. The download and programming algorithm settings are shown in Figure 27.

| Options for Target "LPC51U68_FlashLoader" |
|--|
| Device Target Output Listing User C/C++ Aam Linker Debug Utilities |
| Configure Rash Menu Command |
| Use Target Driver for Rash Programming Vise Debug Driver |
| Use Debug Driver Settings Update Target before Debugging |
| Int File:Edt |
| C Use External Tool for Rash Programming |
| Command: |
| Arguments: |
| E Run Independent |
| Configure Image File Processing (FCARM) |
| Output File: Add Output File to Group: |
| joevice |
| image rues Hoot Polder: |
| |
| OK Cancel Defaults Help |
| Figure 26. Utilities setting |
| |
| Cortex JLink/JTrace Target Driver Setup |
| Debug Trace Rash Download |
| Download Function RAM for Algorithm |
| C Erase Sectors Verfy Start: 0x2000000 Size: 0x1000 |
| (* Do not Erase Reset and Run |
| Programming Algorithm Device Size Device Type Address Range |
| |
| |
| |
| Stat: Sze |
| |
| Add Bemove |
| |
| |
| |
| |
| OK Cancel Apply |
| Figure 27. Download and programming algorithm |

4.2.2 Testing flash operations for RAM-resident version

To test the flash operations such as erase, read, write, and reset for the RAM-resident version, follow these steps:

- · Compile the project.
- Fit the JP10 jumper on the LPCXpresso51U68 board to connect the USB (J5) VBUS to the LPC51U68.
- Use a USB cable to connect the J6 header and PC and download the project executable binary file to the LPC51U68 RAM area.
- Use a USB cable to connect the J5 header and PC.
- Power up and reset the LPCXpresso51U68 board.
- Run *ImiddlewareImcu-bootIbinIToolsIbIhostIwinIbIhost.exe* and enter the "bIhost" command (as shown in Figure 28, Figure 29, and Figure 30). If the following command feedbacks appear, the bootloader-porting flash operations are implemented successfully.



References



5 References

- SDK_2.6.0_LPCXpresso51U68 URL: https://www.nxp.com/products/processors-and-microcontrollers/armmicrocontrollers/general-purpose-mcus/lpcxpresso51u68-for-the-lpc51u68-mcus:OM40005
- SDK_2.6.0_LPCXpresso54018 URL: https://www.nxp.com.cn/design/microcontrollers-developer-resources/lpcxpressoboards/lpcxpresso54018-development-board:OM40003#buy

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.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, 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, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorlQ, QorlQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, UMEMS, 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. 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: 02/2020 Document identifier: AN12689

