AN13497

Firmware Update Using Secondary Bootloader

Rev. 0 — 25 December 2021 Application Note

Contents

1 Introduction....1
1.1 Glossary....1
2 Implementation....1
2.1 Overview....1
2.2 Boot flow....2
2.3 Application image format....3
3 Demo....5
3.1 Demo introduction....5
3.2 Hardware setup....6
3.3 Steps to run the demo....7
3.4 Methods to reenter DSBL....10
3.5 Modifying application image version information....10
4 Consideration and Limitation....11
4.1 About flash read operation....11
4.2 UART multiplex....11
4.3 Enabling/Disabling debug log....11
5 Revision history....12

1 Introduction

Dual image update (reliable update) is an important feature for advanced bootloaders. It assures that at least one image is bootable and works properly at any time. It prevents image lost during the image update period. The mechanism behind the dual image boot loader is simple. If any accident happens, it always makes a copy of previous image. The bootloader detects and uses previous image as the bootable image.

However, the ROM bootloader of LPC55xx does not support dual image feature yet. So, this application note implements a simple dual image update example on LPC55xx. It is useful to users who must implement a second dual image bootloader on LPC55xx series.

1.1 Glossary

Table 1. Abbreviation

Items	Description
SBL	Secondary bootloader
DSBL	Dual image secondary bootloader
DSBL_APP	Example application demo to demonstrate dual image bootloader feature and work with DSBL
MCUBOOT	NXP unified bootloader solution, including protocol, PC software, documentation, and so on. It enables quick and easy programming through the entire product life cycle. For details, see MCUBOOT: MCU Bootloader for NXP Microcontrollers.
blhost	PC Command Line Interface (CLI) tools to implement MCUBOOT protocol. It is a part of MCUBOOT software package.

2 Implementation

2.1 Overview

To ensure a reliable update, implement a dual image layout. Download the image to a temporary region called receive region. In every power cycle, the bootloader checks (integrity check passed) the image in receive region. If the downloaded new image has higher version number than the current image, DSBL copies the image from receive region to the main region. To track the latest version in both regions, locate a version flag in the image.

In summary:

- Receive region:
  - Bootloader always downloads new code to this area.
- Main region:
  - To store a correct image copied from the receive region.
  - DSBL finally jumps to the main image (if exists) to run the application code.

The communication interface in this application note is via UART for demo purpose. Users can easily extend communication interface to others, such as, I2C SPI. The communication protocol follows NXP MCUBOOT protocol. It is compatible with LP55xx ROM. Following MCUBOOT protocol is helpful for users to reuse PC `blhost` software.

Figure 1 shows the overview of flash partition.

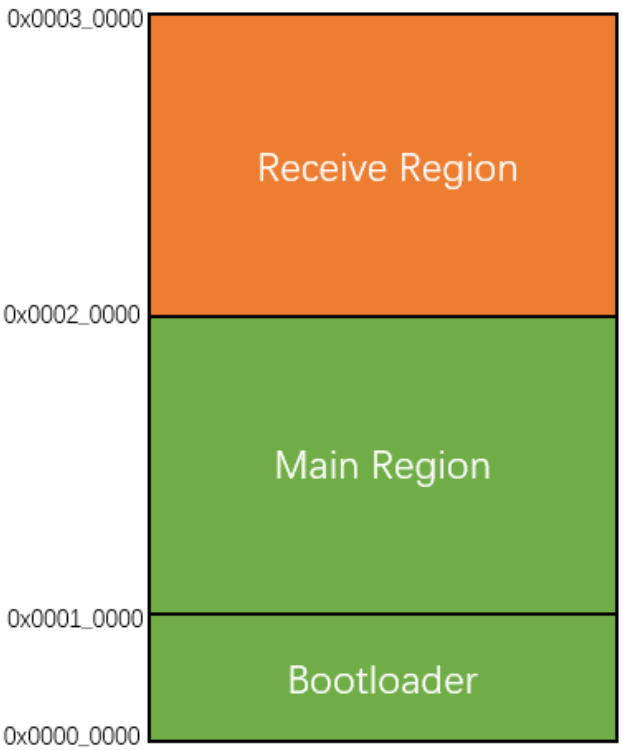

Figure 1. Flash partition

2.2 Boot flow

The DSBL is used to manage images and boot application. Every time when the part is powered on or reset, the DSBL code is executed. Figure 2 shows the DSBL boot flow.

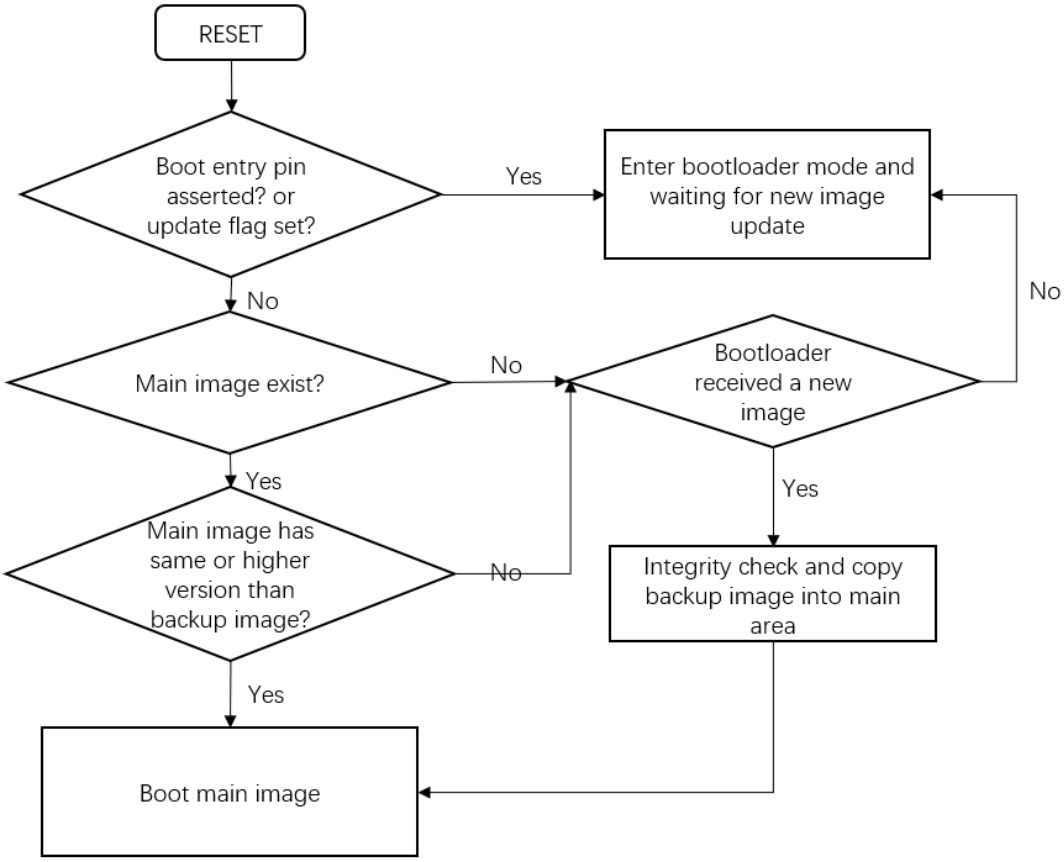

Figure 2. DSBL boot flow

2.3 Application image format

2.3.1 Image memory layout

Figure 3 shows the dual enhanced image type. It contains an image marker at offset `0x24`. It must contain a valid image header in the image pointed to at offset `0x28`. The starting address of the image is at `0x0001_0000` (main region starts address). The image header can reside in any area inside the image. In most cases, the image header is at the end of vector table (offset `0x140`).

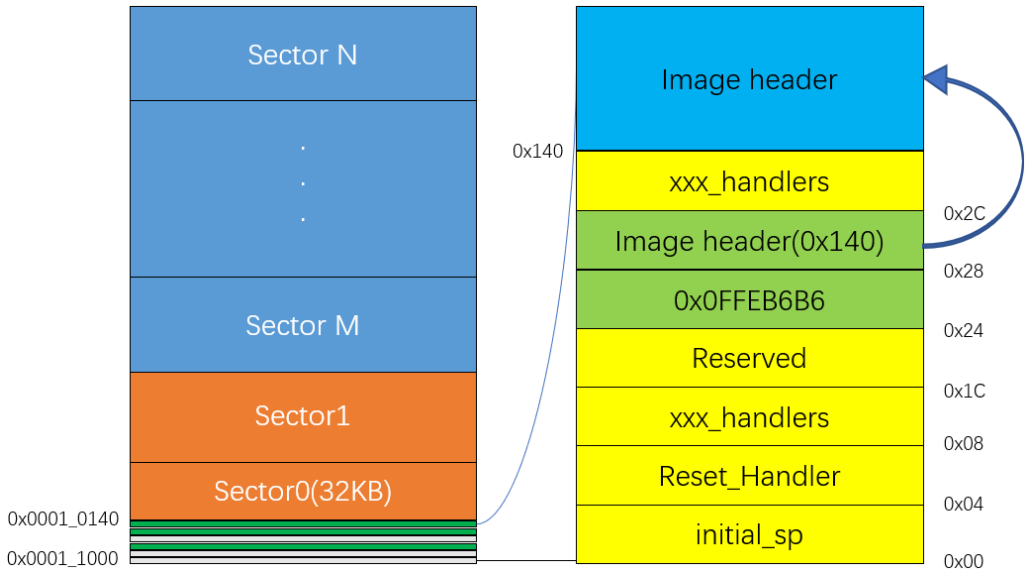

Figure 3. Application image layout

The image header is in a 24 bytes structure, as described in Table 2.

Table 2. Image header structure

Offset	Description
0x00	Header maker set to `0xFEEDA5A5`
0x04	Image Type (NORMAL = 0 or NO_CRC = 1)
0x08	Reserved
0x0C	Image length The length must be the actual length – 4 if CRC value field falls within the length.
0x10	CRC value
0x14	Version

With LPC55xx parts, use the external tools, `image_generator.exe`, to add CRC32 value for entire image binary to the image header.

2.3.2 Image creation

2.3.2.1 Modifying start-up files in IDE

To add image marker and image header, modify start-up files.

KEIL

```
        DATA

__vector_table
        DCD     sfe(CSTACK)
        DCD     Reset_Handler

        DCD     NMI_Handler
        DCD     HardFault_Handler
        DCD     MemManage_Handler
        DCD     BusFault_Handler
        DCD     UsageFault_Handler
__vector_table_0x1c
        DCD     0
        DCD     0xFFFFFFFF ; ECRP
        DCD     0x0FFEB6B6 ; Single Enhanced Image Flag
        DCD     __ImageMarker
        DCD     SVC_Handler
        DCD     DebugMon_Handler
        DCD     0
        DCD     PendSV_Handler
        DCD     SysTick_Handler

        DCD     SMARTCARD1_IRQHandler  ; Smart card 1 interrupt
__ImageMarker
        DCD     0xFEEDA5A5 ; Image Marker
        DCD     0x0 ; Image Type Normal: 0, NO CRC: 1
        DCD     0x0 ; Reserved
        DCD     0x0 ; Image Length
        DCD     0x0 ; CRC Value
        DCD     0x2 ; Version
__Vectors_End

__Vectors       EQU   __vector_table
__Vectors_Size  EQU     __Vectors_End - __Vectors
```

Figure 4. Adding image marker and image header in Keil (put image header at the end of vector table)

2.3.2.2 Using external tools to add length and CRC value in image header

When the image type word in the image header is `0x00` (NORMAL), to add length and CRC value into image header, use the external tools, `image_generator.exe` located at:

/pc55s36_dsbl\boards\lpcxpresso55s36\dual_sbl\lpc55xx_dsbl_app\tools

Double-click `post_build.bat`, and the script calls **image_generator.exe** and generates the binary named `dsbl_app_crc.bin` in this folder. Download the `.bin` file to the receive region. For a step-by-step guide about how to use those tools, see Demo.

3 Demo

3.1 Demo introduction

The demo contains two projects based on SDK, as described in Table 3.

Table 3. Demo project description

Project name	Location in SDK	Description
lpc55xx_dsbl	\boards\lpcxpresso55s36\dual_sbl\	Dual image second bootloader project
lpc55xx_dsbl_app	\boards\lpcxpresso55s36\dual_sbl\	Demo application project

- `lpc55xx_dsbl` stands for **lpc55xx dual image second bootloader**. It is executed at boot up. This program communicates with PC host, checks image, and copy tasks. It is the first project to download into EVK board.
- `lpc55xx_dsbl_app` stands for **lpc55xx dual image second bootloader application example**. It is almost same as the `hello_world` example. The differences are:
    1. This image has an image marker and an image header resided after the vector table. DSBL can regionalize this image.
    2. To put loading/starting address into the main image region, modify the linker starting address from `0x0000_0000` to `0x0001_0000`.

3.2 Hardware setup

3.2.1 LPC55S36-EVK

The hardware uses LPCXpresso55S36 board, as shown in Figure 5. Make sure you have read board user guide and familiar with basic function of the board, such as, the positions of the reset button/user button and the debug connector.

Figure 5. LPCXpresso55S36 board

This demo uses Debug and UART USB connector (J1) as the debug interface and UART-USB bridge. It uses USR button (SW3) as the entry pin of the second bootloader.

3.3 Steps to run the demo

Before running the demo, make sure that:

- You have basic knowledge about LPCXpresso55xx board.
- Related LPC-Link II debugger driver is installed.
- The `hello_world` example runs successfully on the SDK folder.
- UART communication with PC is verified to be successful.

To run the demo, perform the following steps:

1. Connect USB with Debug and UART USB connector (1). The board is powered on to establish debug and UART connection.
2. Open, compile, and download the `lpc55xx_dsbl` project. Open the serial terminal with **115200-N-8-N-1**.
3. Hold the wake-up button and then press the **RESET** button. This action forces the DSBL to enter the boot loader mode. When entering this mode, DSBL does not boot any application, but wait for UART connection.

4. By default, the `lpc55xx_dsbl` enables the debug log. The terminal puts information, as shown in Figure 6. The log indicates that DSBL runs successfully and enters the boot loader mode.

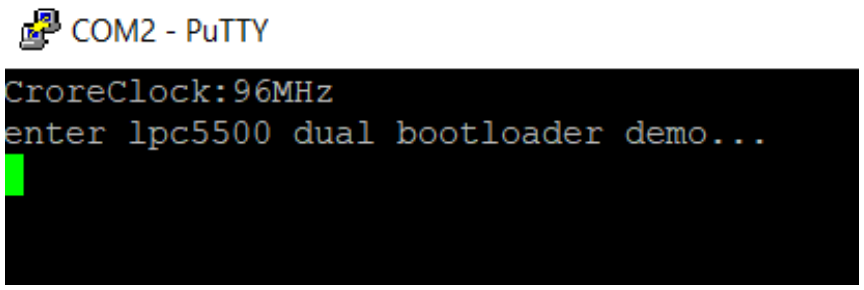

Figure 6. DSBL enters boot loader mode

5. Open and compile the `lpc55xx_dsbl_app` project. Do not use IDE to download `lpc55xx_dsbl_app` project. Otherwise, it is meaningless to demonstrate boot loader feature.
6. Open the `\boards\lpcxpresso55s36\dual_sbl\lpc55xx_dsbl_app\cm33_core0\tools` folder and double click `post_build.bat`. This action generates `dsbl_app_crc.bin` which adds CRC and image length information to `image_generator.exe.`, as shown in Figure 7.

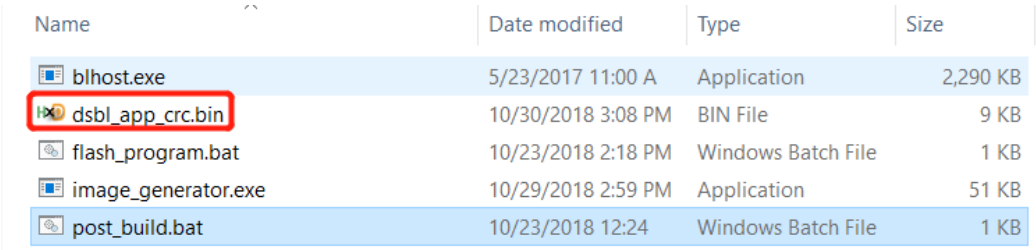

Figure 7. Using post_build.bat to generate dsbl_app_crc.bin

The **dsbl_app_crc.bin** is the binary image to be download to the receive region.

7. Close serial terminal, open bash window or command window, and execute `flash_program.bat`. This script calls `blhost.exe` and downloads `dsbl_app_crc.bin` to the receive region. To run `flash_program.bat`, two parameters are required: UART COM index and the full name of the app image.

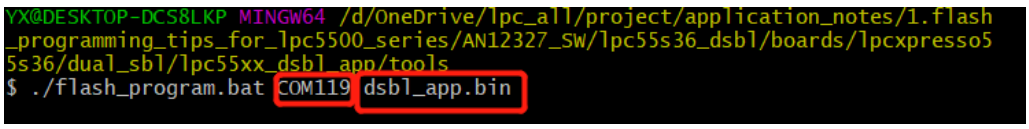

Figure 8. Downloading dsbl_app_crc.bin via blhost

8. After executing the script, download the new image to the receive region, as shown in Figure 9.

```
D:\OneDrive\lpc_all\project\application_notes\1.flash_programming_tips_for_lpc55
00_series\AN12327_SW\lpc55s36_dsbl\boards\lpcxpresso55s36\dual_sbl\lpc55xx_dsbl_
app\tools>blhost.exe -p COM119 get-property 3
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 131072 (0x20000)
Flash Start Address = 0x00020000

D:\OneDrive\lpc_all\project\application_notes\1.flash_programming_tips_for_lpc55
00_series\AN12327_SW\lpc55s36_dsbl\boards\lpcxpresso55s36\dual_sbl\lpc55xx_dsbl_
app\tools>blhost.exe -p COM119 get-property 4
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 65536 (0x10000)
Flash Size = 64 KB

D:\OneDrive\lpc_all\project\application_notes\1.flash_programming_tips_for_lpc55
00_series\AN12327_SW\lpc55s36_dsbl\boards\lpcxpresso55s36\dual_sbl\lpc55xx_dsbl_
app\tools>blhost.exe -p COM119 get-property 11
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 512 (0x200)
Max Packet Size = 512 bytes

D:\OneDrive\lpc_all\project\application_notes\1.flash_programming_tips_for_lpc55
00_series\AN12327_SW\lpc55s36_dsbl\boards\lpcxpresso55s36\dual_sbl\lpc55xx_dsbl_
app\tools>blhost.exe -p COM119 get-property 16
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 305419896 (0x12345678)
System Device ID = 0x12345678

D:\OneDrive\lpc_all\project\application_notes\1.flash_programming_tips_for_lpc55
00_series\AN12327_SW\lpc55s36_dsbl\boards\lpcxpresso55s36\dual_sbl\lpc55xx_dsbl_
app\tools>blhost.exe -p COM119 flash-erase-region 0x20000 0x10000
Ping responded in 1 attempt(s)
Inject command 'flash-erase-region'
Successful generic response to command 'flash-erase-region'
Response status = 0 (0x0) Success.

D:\OneDrive\lpc_all\project\application_notes\1.flash_programming_tips_for_lpc55
00_series\AN12327_SW\lpc55s36_dsbl\boards\lpcxpresso55s36\dual_sbl\lpc55xx_dsbl_
app\tools>blhost.exe -p COM119 write-memory 0x20000 dsbl_app.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 8508 (0x213c) 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 8508 of 8508 bytes.
```

Figure 9. Download log for flash_program.bat

9. Reopen the UART terminal and press the **RESET** button.

```
CoreClock:96000000Hz
boot application...
scan golden region...
image found: 0x00010000
sheader_marker    :0xFEEDA5A5
image_type        :0x00000000
reserved          :0x00000000
img_len           :8504
crc_value         :0xEA79A534
version           :0x00000007
scan backup region...
crc check failed
golen image ok, no backup image, boot
dsbl: boot @ 0x00010000
i am golden image
```

Figure 10. Image copied to main region and booted

The log, **image found: 0x0001_0000**, indicates that DSBL has detected there is an image resided in golden region.

3.4 Methods to reenter DSBL

Besides using the wake-up button to enter DSBL, there are two more methods to enter DSBL forcefully for application update.

3.4.1 Reinvoke

Define the `sbl_api` structure in your application, as shown in Figure 11. Then, call `re_invoke`.

```
sbl_api->reinvoke();
```

This action forces CPU to jump to DSBL immediately, just like `ROM_API` reinvoked in legacy LPC parts.

```
typedef struct
{
    void (*reinvoke)(void);
    void (*set_update_flag)(void);
    void (*test)(void);
}sbl_api_t ;

static  sbl_api_t  *sbl_api  = (sbl_api_t *)(0x400);
```

Figure 11. DSBL API structure

3.4.2 Set_update_flag

This method is similar to reinvoke, but this API does not enter DSBL immediately. It lets DSBL enter the update mode at next power cycle. A non-volatile update flag is set. DSBL counts the update fail times. If updating image in receive region fails too much (the default value is three times), the DSBL clears `update_flag` and boots main image. Otherwise, on each power cycle, DSBL does not boot main image but wait for a successfully download operation. Calling this API is same as reinvoke:

```
sbl_api->set_update_flag();
```

3.5 Modifying application image version information

To update application image version information, modify version word in image header.

NOTE

DSBL copies received image to main region only if the received image has higher version number than that of the main image.

```
        DCD     PVTVF1_RED_IRQHandler  ; PVT interrupts
__deimage_header
                DCD     0xFEEDA5A5              ; Image marker
                DCD     0x00000000              ; Image type Normal: 0, NO CRC:
                DCD     0x00000000              ; Reserved
                DCD     0x00000000              ; Image length
                DCD     0x00000000              ; CRC value
                DCD     0x00000007              ; Version
__Vectors_End
```

Figure 12. Updating DSBL_APP image version information

4 Consideration and Limitation

4.1 About flash read operation

In most cases, AHB can read flash directly. But in LPC55xx, any attempt to directly read an erased flash (erased but not written) may lead to Hard Fault due to ECC mechanism of flash. This issue brings inconvenience to bootloader development. To tackle this problem, implement a **Non-AHB method to read flash data API** to replace AHB directly read. The code for **Non-AHB method to read flash data API** is in `memory.c`. For details, check the code.

4.2 UART multiplex

In this demo, the following three functions use the same UART:

1. DSBL debug log output
2. Application demo log output
3. Communication interface for DSBL to download image

Consequently, there is a UART multiplex conflict issue. Whenever using `blhost` to downloading image, close UART terminal to release PC COM port resource for `blhost`.

4.3 Enabling/Disabling debug log

To enable/disable DSBL debug log, use macro. Comment macro `DIMAGE_DEBUG` in `dimage.h`, and all debug outputs are disabled, as shown in Figure 13.

```
#include <stdlib.h>
#include <stdint.h>

#define DIMAGE_DEBUG

#if defined(DIMAGE_DEBUG)
#include <stdio.h>
#define DIMAGE_TRACE  printf
#else
#define DIMAGE_TRACE(...)
#endif

/* generate image via: ./image gener
```

Figure 13. Enabling/Disabling DSBL debug log

5 Revision history

Rev.	Date	Description
0	25 December 2021	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

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

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

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

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

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

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

Date of release: 25 December 2021

Document identifier: AN13497

arm