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>
<thead>
<tr>
<th>Items</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBL</td>
<td>Secondary bootloader</td>
</tr>
<tr>
<td>DSBL</td>
<td>Dual image secondary bootloader</td>
</tr>
<tr>
<td>DSBL_APP</td>
<td>Example application demo to demonstrate dual image bootloader feature and work with DSBL</td>
</tr>
<tr>
<td>MCUBOOT</td>
<td>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.</td>
</tr>
<tr>
<td>blhost</td>
<td>PC Command Line Interface (CLI) tools to implement MCUBOOT protocol. It is a part of MCUBOOT software package.</td>
</tr>
</tbody>
</table>

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.

![Flash partition diagram](image)

**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.
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 0x0000_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).
The image header is in a 24 bytes structure, as described in Table 2.

### Table 2. Image header structure

<table>
<thead>
<tr>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Header maker set to 0xFEEDA5A5</td>
</tr>
<tr>
<td>0x04</td>
<td>Image Type (NORMAL = 0 or NO_CRC = 1)</td>
</tr>
<tr>
<td>0x08</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| 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.
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:

```
lpc55s36_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

<table>
<thead>
<tr>
<th>Project name</th>
<th>Location in SDK</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpc55xx_dsbl</td>
<td>\boards\lpcxpresso55s36\dual_sbl\</td>
<td>Dual image second bootloader project</td>
</tr>
<tr>
<td>lpc55xx_dsbl_app</td>
<td>\boards\lpcxpresso55s36\dual_sbl\</td>
<td>Demo application project</td>
</tr>
</tbody>
</table>

- **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.
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.

![COM2 - PuTTY](image)

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.

![Screenshot](image)

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.

![Command](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.
9. Reopen the UART terminal and press the **RESET** button.

Figure 9. Download log for flash_program.bat
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.

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

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

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:

```c
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.
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 <string.h>
#define DIMAGE_DEBUG

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

/* generate image via: ./image gen
```
5 Revision history

<table>
<thead>
<tr>
<th>Rev.</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>25 December 2021</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
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, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

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