AN12594

OTA Process introduction

Rev. 0 — September 2019 Application Note

Contents

1 Introduction

The LPC5411x are Arm® Cortex®-M4 based microcontrollers for embedded applications. These devices include:

- an optional Arm Cortex-M0+ coprocessor
- up to 192 KB of on-chip SRAM
- up to 256 KB on-chip flash

Over-The-Air (OTA) is a procedure to update the firmware without using physical wires. OTA is a solution for the current **LPC54114 BLE Audio System**. When a product is ready and released in the field, OTA can be used to upload new firmware that brings new features.

- For a customer, OTA is convenient because the Headset does not need to be connected to a PC.
- For a manufacturer, OTA reduces the BOM cost as no USB hardware needs to be present.

This document provides **OTA Operation Steps** to support firmware update in the memory of LPC54114 with the tools of **NXH3670_SDK_Gaming_G3.0**. For more specific OTA introductions including HAPI, Concept, refer to HAPI OTA.

2 Concepts

2.1 OTA update process

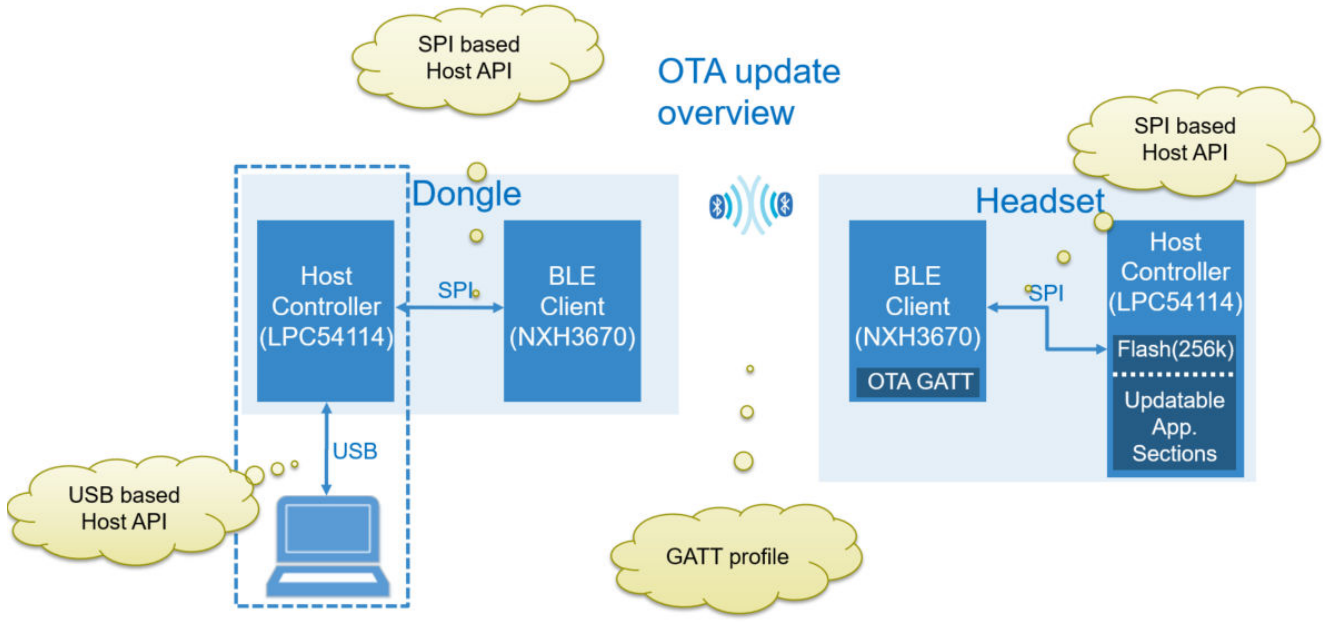

Figure 1. OTA setup

The setup requires a Dongle board, Headset board, PC and a USB cable. The USB cable is used to connect the Dongle and PC.

A typical scenario based on LPC54114 can be summarized as below:

1. Dongle and Headset are initially programmed over USB interface.
2. Dongle and Headset are paired.
   - Pairing Data (PD) of Headset is independent on the PD of Dongle.

     Once the Headset is booted up, it retrieves PD from its own Memory that is written in advance and will not store extra PD.
   - PD of Dongle is dependent on the PD of Headset.

     Once the Dongle is booted up, it retrieves PD and pair with Headset, and then stores the Headset device information in Dongle's PD section.
3. Dongle is re-flashed with the **OTA_Dongle** application, which is the start of the OTA process.
   - **OTA_Dongle** can be used as VCOM to transfer data between PC and LPC54114 throught the USB.
   - Before re-flashing operation, make sure that the PD is not erased (Dongle is responsible for getting Headset's device information and **OTA_Dongle** is not). Two NXH devices are paired firstly and then connected, so they can't be connected if the user has erased PD date stored in Dongle's Flash.
   - In the **Debug mode**, the code takes up much space. The Headset does not have eough space to store **Headset** and **OTA_Headset** at the same time. So user can re-flash Headset with the **OTA_Headset** application to test OTA function.

     NOTE

     It is recommended to use the same `layout_release_sdk.yml` for both Headset and OTA_Headset demos. Only the application and NXH3670's binaries to Flash are required to be downloaded (the Partition table and init pairing_data are not required).
4. OTA process is triggered by PC application.
5. OTA finishes. The firmware of Headset is updated.
6. Re-flash the Dongle with Gaming application, which is the end of the OTA process.

2.2 SSB

The Second Stage Bootloader (SSB) is automatically bootstrapped by the (ROM) first stage bootloader.

You can store multiple firmware in Flash according to your requirements and inform SSB which firmware to boot.

For example, considering Headset board has no USB port in the **LPC54114 BLE Audio System**, developers store at least three firmware in advance, including

- SSB: to decide which firmware to boot
- OTA firmware: to receive new firmware
- application firmware: actual application, including specific Headset functions

Taking the current demos as an example, the following describes the functions of SSB:

1. Set the VTOR to the application vector table address.
2. Set stack pointers to the application stack pointer.
3. Jump to the application (PC now points to application).

NOTE

In the current design, the NXH3670 is used to transfer data over the air and program LPC54114 through the SPI interface with the SSB code.

Define `NO_CRP` to ensure that the program starts at the specified location, if using SSB to boot application (the ignoring of this step may result in the `0x02FC` offset in the application).

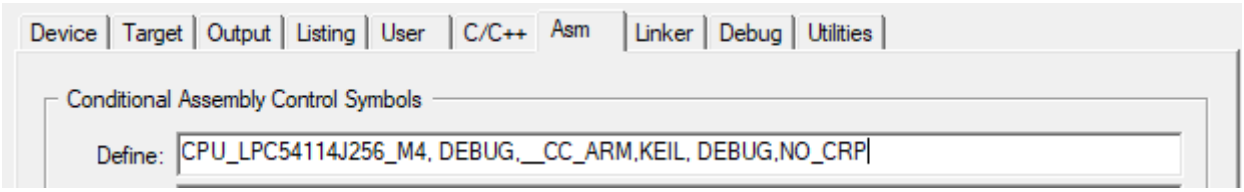

Figure 2. `NO_CRP` settings

2.3 Partition table

The Dongle, OTA_Dongle, Headset and OTA_Headset applications and their locations are required to be mapped in Flash. This mapping is present in the Partition Table stored at a fixed offset in the Flash memory of the Host controller. For more information, refer to HAPI OTA.

The followings are important to note:

1. The Sector size is different between LPC54114 and KL27.
   - LPC54114: 8 Sectors (0-7), and the size of per Sector is 32 KB.
   - KL27: 256 Sectors (0-255), and the size of per Sector is 1 KB.
2. The current software design of the OTA process contains the following three scenarios.
   - The **Active flag** of Partition Table and PD of Dongle is reassigned with new value to inform SSB which firmware to boot.
   - If the `KL_APP` is set to **Sector0 (0x00000000 – 0x0000 7FFF)**, the original data is copied to SRAM during OTA. You can change the data and then flush the changed data back to **Sector0** with FLASHIAP API.
   - The interrupted vectors are stored in **Sector0**.

   All the data can be cleaned up when there is an unexpected power-off or other uncertainty reason.

Therefore, in the software design of LPC54114, **SSB** and **ota_app**, which will never be changed, are put on **Sector0** and **Partition Table** and **PD** on **Sector7**.

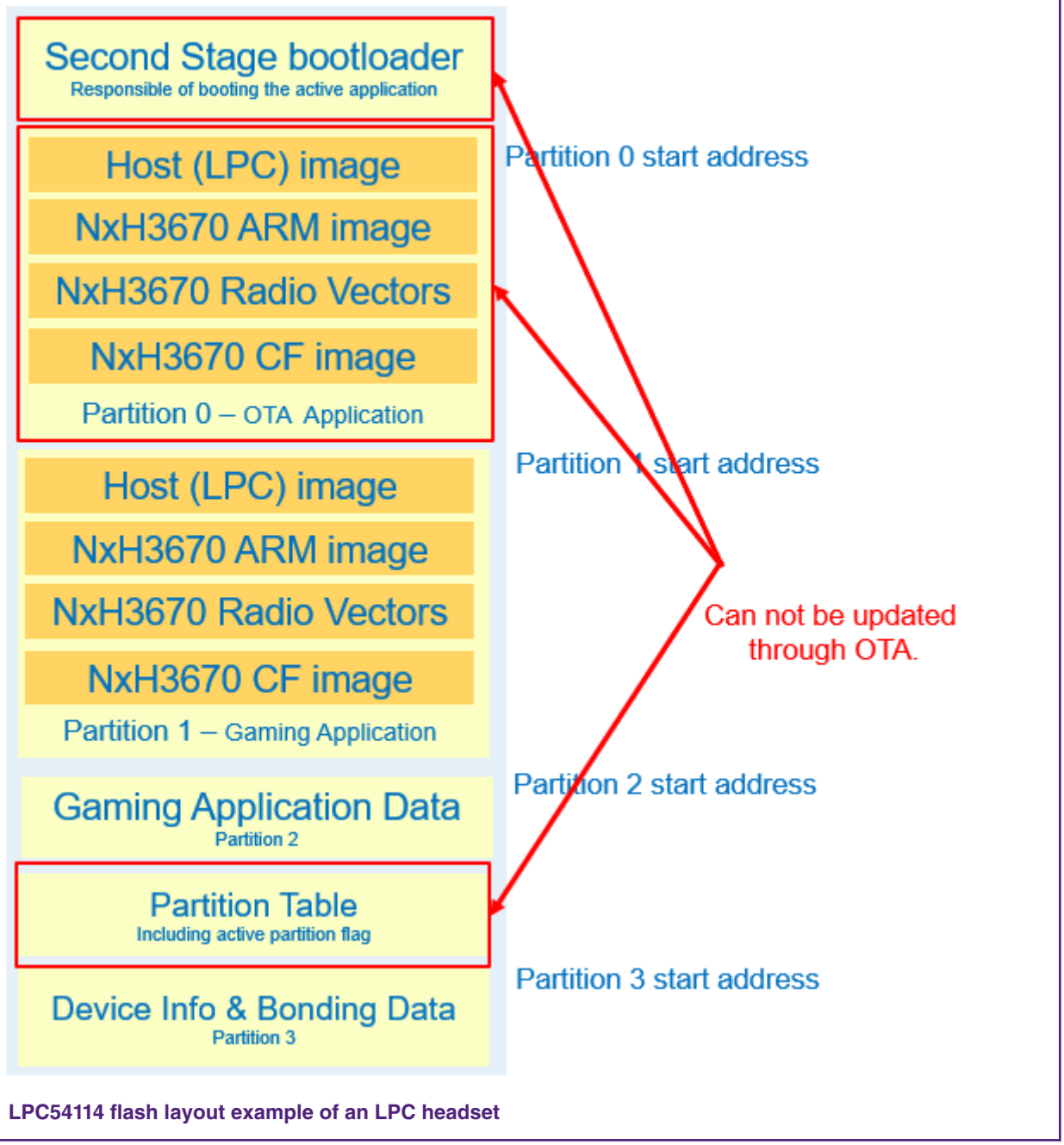

Figure 3. LPC54114 flash layout example of an LPC headset

- **Partition 0** is the OTA application, and it contains the Host Controller (LPC54114) firmware and NxH3670 Arm Image.

  For **OTA_Dongle**, only `ota_app` and `NxH_Binary` (`Arm.phOtaDongle.ihex.eep`) are required to be flashed.

  — `rfmac` (`rfmac.eep`) is added in the NxH image.

  — `CF` (`phStereoInterleavedAsrcTx.eep`) is not required or used.

- **Partition 1** is the Gaming Application, and it contains the firmware for the Host Controller (LPC54114) firmware, the NxH3670 Arm image, the NxH3670 Audio RadioVectors, and NxH3670 CoolFlux image.
- **Partition 2** contains Application data meant as general-purpose data storage for the Gaming application. Currently, **Partition 2** is not used.

- **Partition 3** contains the Device info and Bonding data. Device information contains BLE specific attributes that need to be present for the air interface to work. Bonding data makes sure that Dongle and Headset automatically reconnect. Bonding data is only relevant for the Dongle.

3 Using flashtool

This document lists the operation steps to use `.BAT` to update firmware easily and quickly. For more specific introduction of Flashtool, refer to the **HAPI OTA** and tools sections in NXH3670_SDK_Gaming_G3.0.

3.1 Modificataion

1. `ota_update_headset.bat`

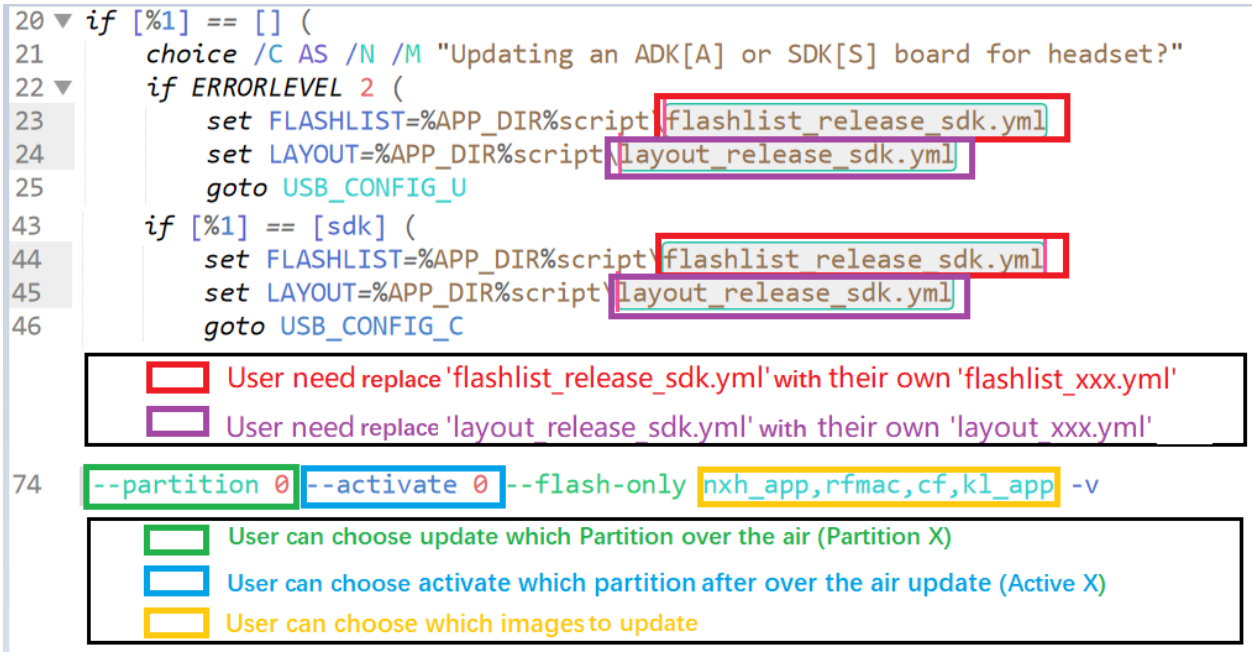

Figure 4. Modifying `ota_update_headset.bat`

With JLink, how can we convert `.yml` of Partition table to `.BIN` that will be download to Flash?

a. Open a command line interface.

b. Go to the `NXH3670_SDK_Gaming_G3.0` folder.

c. Run `flash_scripts\flashtool.cmd` -> `dev table.bin` -> `connection export` -> `layout kinetis_democode \apps\kl_dongle\script\layout_debug_sdk.yml`.

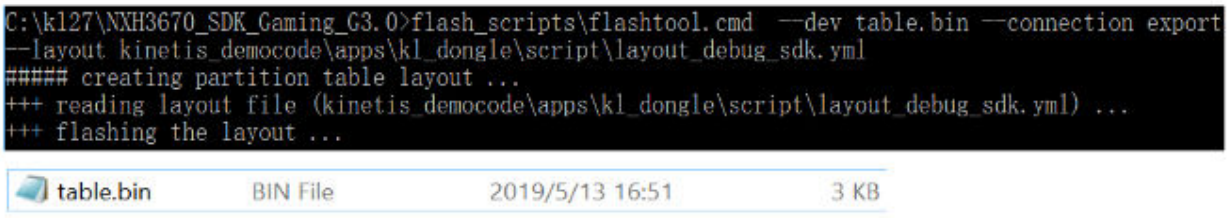

Figure 5. Converting `.yml` to `.bin`

However, first 2560 (`0Xa00`) bytes of this `table.bin` are all `0x00`. This document provides two methods.

- Make sure that `table.bin` is flashed before flashing the SSB located in `0x00`. Otherwise, the `table.bin` will overwrite the SSB. So for OTA, user have to port the `kl_ssb` application or flash SSB file as well.
- Or, you can delete 2560 (`0xA00`) bytes of this `table.bin` and then download the changed `table.bin` to Partition table address.(In our software, we put it to `0x3f400`).

2. `flashlist_release_sdk.yml`

```
binaries:
    - name: ssb
      files:
        - kinetis_democode/apps/kl_ssb/sdk/release/kl_ssb_sdk.bin
      address: 0x0
    - name: kl_app
      files:
        - kinetis_democode/apps/kl_headset/sdk/release/kl_headset_sdk.bin.eep
      offset_index: 0
      partition: { name: "app", type: firmware }
      headroom: -1
```

User can replace 'kl_ssb_sdk.bin' with their own SSB.bin

User can replace this location with their own location for firmware

User can replace 'kl_headset_sdk.bin.eep' with their own '.EEP' file

Figure 6. Modifying `flashlist_release_sdk.yml`

`flashlist_release_sdk.yml` (`kl_headset`) lists the binaries and offset_index of Partition used to operate OTA. Figure 6 shows the example to update `kl_headset_sdk.bin.eep` to `offset_index_0` in the current Partition.

3. `layout_release_sdk.yml`

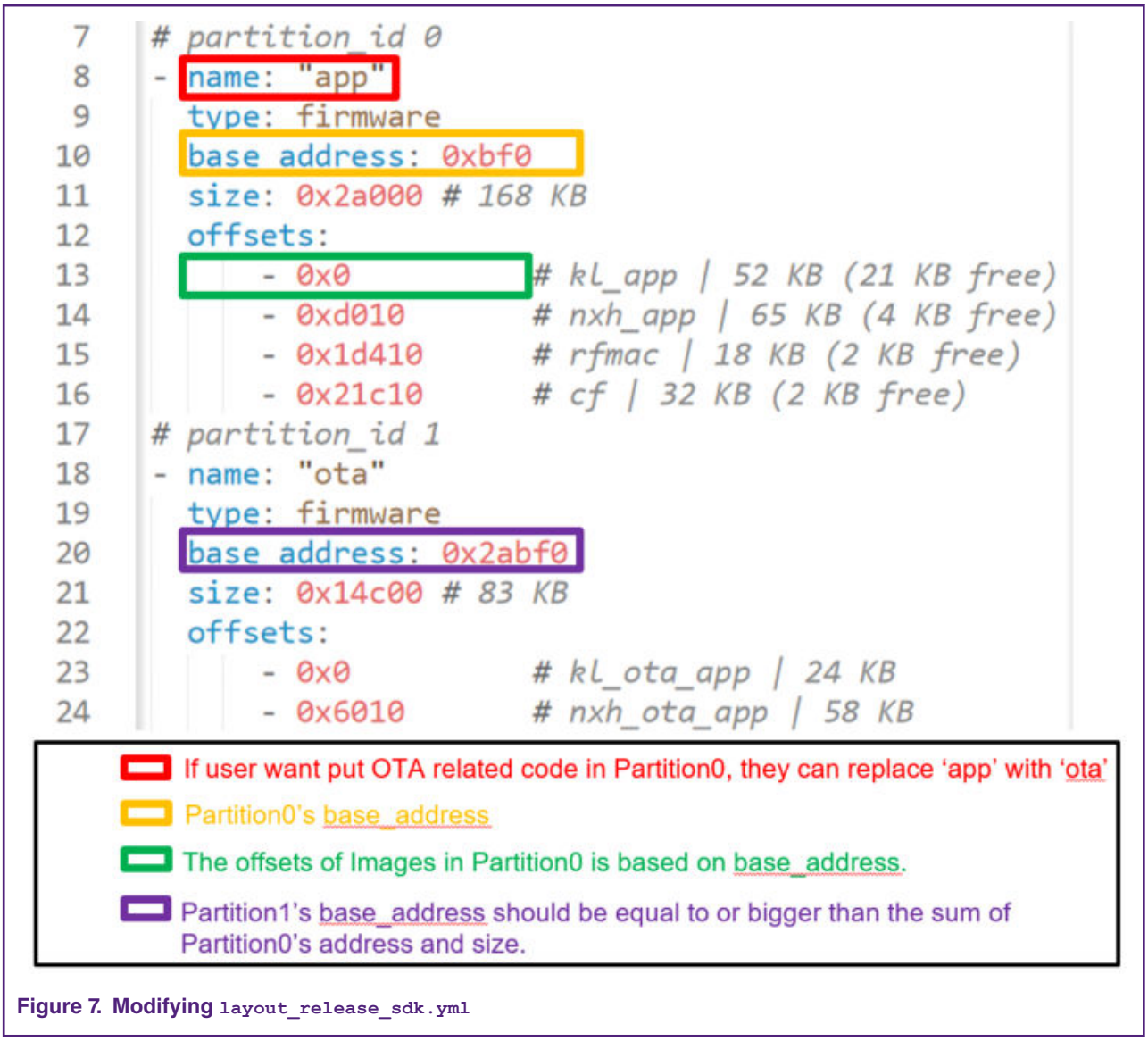

Figure 7. Modifying `layout_release_sdk.yml`

You can design your own `layout_release_sdk.yml` to meet the use of Flash. In the software design, MCU reads NXH_Binaries from specified location and then transfer data to NXH3670 through the SPI interface. Make sure the design of Flash layout is correct.

How can we convert `.BIN` to `.EEP` used in OTA process?

Use the `…\tools\to_eep.cmd` and type in `-i XXX.bin -o XXX.bin.eep`.

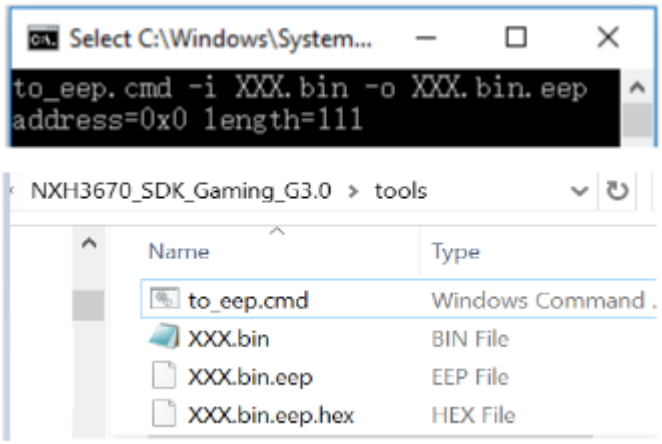

Figure 8. Converting `.BIN` to `.EEP`

The following two items are important;

- To keep `ota` Partition the first in memory, in the `layout_release_sdk.yml` file, keep in the order of `ota`, `app`, …. WIthout this order, a `Partition Table.bin` cannot be output or the Partition Table is not correct.
- Make sure that the addition value of `base_address of Partition`and `its size` is smaller than `base_address of the next Partition`.

3.2 Test process

After `ota_update_headset.bat`, `flashlist_release_sdk.yml`, and `layout_release_sdk.yml` are correctly modified, OTA can start.

This document lists steps when **Dongle** and **Headset** have been paired.

1. Download **OTA_Dongle** and make sure PC can recognize it as a **USB Serial Device**.

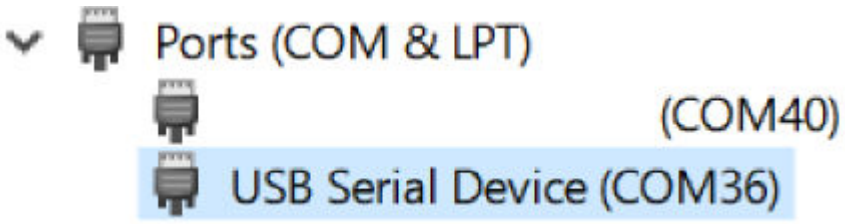

Figure 9. VCOM USB serial device

As shown in Figure 9, the PC recognizes it as **COM36**.

2. This document lists two cases.
   - **Case 1**: `app` is running instead of `ota_app`.
     - In the **Release mode**, the `Active_flag` of APP_Partition is **1** (`app`), which indicates that userw need to send commandw to switch `Active_Partition` to `OTA_Partition`: **0** (`ota_app`).
     - In the **app** case, if `phOtaHeadset.ihex.eep` is used instead of `phGamingRx.ihex.eep`, it indicates that users have boot and start NXH3670 as OTA function (user can use OTA related tool to communicate with Dongle board with firmware `phOtaHeadset.ihex.eep`), and there is no need to switch remote Active_Partition. The `hci_table` of `app` do not have OTA related code, so it cannot be used for OTA .

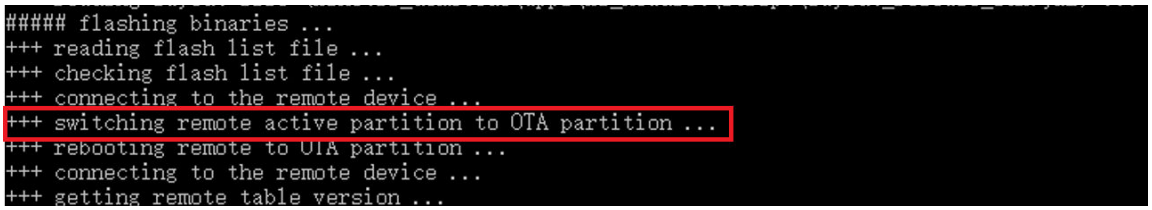

Figure 10. switch operation

- **Case 2**: `ota_app` is running and `NXH_Binary` is `phOtaHeadset.ihex.eep`
  - In the **Debug mode**, `Active_flag of OTA_Partition` is **1**, which indicates that the code is ready for OTA process and there is no need to switch remote Active_Partition.

In this step, this document assumes that Case 2 is used.

a. Open a command line interface.

b. Go to the `flash_scripts` folder.

c. Type in `ota_demo_sdk.bat` (it may be changed or renamed), board (SDK board is used for the test, so type in **S**), and USB port name (**COM36**).

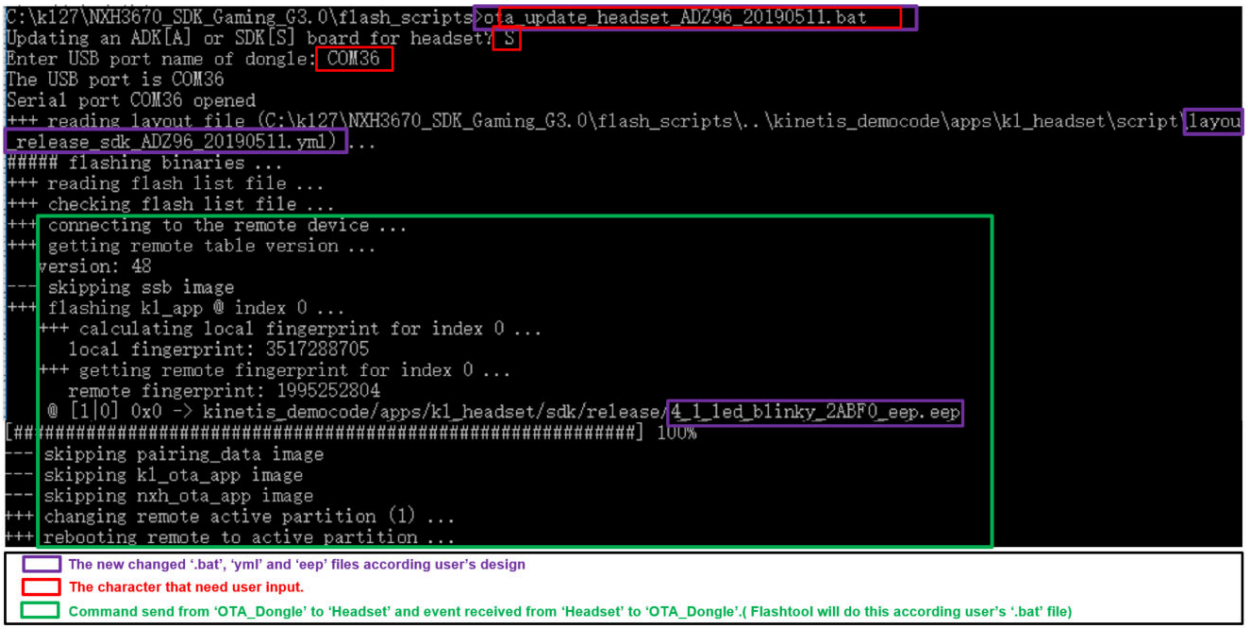

Figure 11. CMD of OTA process

As shown in Figure 11, `[##..##] 100%` indictaes the update progress.

Now, all the OTA work is completed.

3. LOG information

`OTA_Headset_Debug_mode` provides the LOG information and it can be downloaded to view OTA progress better.

Figure 12. OTA_Headset WriteToPartition event

4 Software design

In order to describe the software design clearly, some programs are attached for user as reference.

4.1 Code of SSB

```
enum _vector_table_entries { kInitialSP = 0,kInitialPC };

uint32_t *appVectorTable = NULL;
uint32_t applicationAddress = 0;
uint32_t stackPointer = 0;

appVectorTable = (uint32_t *)(entry.startAddress + entry.imageOffsets[0] +
NVMMGR_EEP_INITIAL_HEADER_SIZE);
applicationAddress = appVectorTable[kInitialPC];
stackPointer = appVectorTable[kInitialSP];

JumpToApplication(applicationAddress, stackPointer);
```

...

```
void JumpToApplication(uint32_t applicationAddress, uint32_t stackPointer)
{
    /* Static variables are needed as we need to ensure the values we are using are not stored on the
previous stack */
    static uint32_t s_stackPointer = 0;
    s_stackPointer = stackPointer;
    static void (*farewellBootloader)(void) = 0;
    farewellBootloader = (void (*)(void))applicationAddress;

    /* Set the VTOR to the application vector table address */
    SCB->VTOR = applicationAddress;

    /* Set stack pointers to the application stack pointer */
    __set_MSP(s_stackPointer);
    __set_PSP(s_stackPointer);

    /* Jump to the application */
    farewellBootloader();
}
```

`startup_LPC54114_cm4.s` files can be modified to implement SSB feature.

```
bootValidApp    PROC
      EXPORT  bootValidApp
        LDR     r1, [r0, #0]      ; Get app stack pointer
```

```
    MOV     sp,r1
    LDR     r1, [r0, #4]    ;
    BX      r1              ; Get app reset vector
     ENDP                   ; PC now point to App_Firmware
```

4.2 Code of OTA receive

To let user understand the whole receive process of OTA easily, this section provides a `event handler` in the `OTA_Headset` code: `HCI_VS_WRITE_TO_PARTITION_SUB_EVENT` to introduce how to write firmware to Flash.

Assuming Dongle board is running **OTA_Dongle** demo and Headset board is running **OTA_Heatset** demo, the NXH3670 of Headset can receive event from Dongle and transmit event to Host Controller (LPC54114) through the SPI interface.

1. The NXH3670 of Headset receives `HCI_VS_WRITE_TO_PARTITION_SUB_EVENT` (0Xe1) from the NXH3670 of Dongle and then runs `HCI_EvtWriteToPartitionHandler`.

   ```
    {
        .evtCode = HCI_VS_EVENT_CODE,
        .subEvtCode = HCI_VS_WRITE_TO_PARTITION_SUB_EVENT,
        .evtHandler = HCI_EvtWriteToPartitionHandler,
        .evtParmsLen = HCI_UNDEFINED_PARAMETER_LENGTH,
     },
   ```

2. The NXH3670 of Headset writes the data to the requested partition with offset. The following lists some APIs.

   - To write outside of the current cached sector, copy all data in the current sector.

     ```
     ReadFromFlash(s_Context.cacheBuf, SECTOR_SIZE_IN_BYTES, s_Context.cachedSectorAddr)
     ```

     The sector size of LPC54114 is 32 KB, so a 32 KB cacheBuf is required to be defined in code design.

   - Users can modify Cache with the data sent from NXH3670 of Dongle board.

     ```
     memcpy(&s_Context.cacheBuf[cacheOffset], data, cpyLen);
     ```

   - When the data of one packet is copied to cacheBuf, you can program Sector by using Flash write API.

     ```
     ProgramSector(s_Context.cacheBuf, SECTOR_SIZE_IN_BYTES, s_Context.cachedSectorAddr);
     ```

3. The NXH3670 of Headset notifies that the operation is success to NXH3670 with a command.

   ```
   HCI_CmdDataWritenToPartition(&req);
   HCI_SendCmdBlocking(&req)
   ```

4.3 Code of OTA send

To let user understand the whole send process of OTA easily, this section uses Pseudo codes to introduce how **OTA_Dongle** sends firmware to **OTA_Headset**.

```
void UsbVcomDataReceived_Cb(uint32_t length, uint8_t *data)
{
    switch (opcode) {
        case HCI_CMD_VS_CONNECT_OPCODE: {
            …
            HCI_CmdPrepareConnect(&hciReq, &connectParams);
            ….
            break;}
```

```
        default: {
            ….
            HCI_CmdPrepareHostGenericCmd(&hciReq, data, length);
            ….
            break;}
    }
```

NOTE

- Case `HCI_CMD_VS_CONNECT_OPCODE`

  This command indicates that **OTA_Dongle** wants to connect **OTA_Headset**.

- Default CMD

  OTA_Dongle will send any other command to **OTA_Headset** with. Actually, the MCU of **OTA_Headset** is responsible for writing these data to Flash.

5 Conclusions

With the Flashtool and files in **NXH3670_SDK_Gaming_G3.0**, you can update programs or make changes on your design needs. The firmware update speed via OTA is about 1 KB per second.

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, QorIQ, QorIQ 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, and UMEMS 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. 2019. **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: September 2019
Document identifier: AN12594

arm