AN12649
OTA 流程介绍

版本 0 — 2020 年 1 月

应用笔记

目录

1 导言

K32L2B 是基于增强型 Cortex-M0 +（CM0 +）内核的高度集成的，市场领先的超低功耗 32 位微控制器。此系列包含以下功能：

- 内核时钟高达 48 MHz，总线时钟高达 24 MHz。
- Flash 最大为 256 KB，RAM 大小为 32 KB。
- 两个 SPI 模块。
- 两个（I2C）模块。
- 一个 FlexIO 模块。

Over-The-Air (OTA)是在不使用物理线缆的情况下更新固件的过程，因此可以将它应用于当前低功耗蓝牙音频系统的解决方案。当产品准备好并已经发售时，OTA 可以用来更新带有新功能的固件。

- 从客户的角度来看，OTA 是方便的，因为耳机不需要连接到 PC 上。
- 从制造商的角度来看，OTA 降低了 BOM 成本，因为不需要 USB 接口。

本文档提供 OTA 操作步骤，以支持希望使用 NXH3670_SDK_Gaming_G3.0 工具轻松更新 K32L2B 内存中的固件的用户。

有关更具体的 OTA 介绍，请参阅 KL27 的 HAPI OTA。

2 概念

2.1 OTA 更新过程

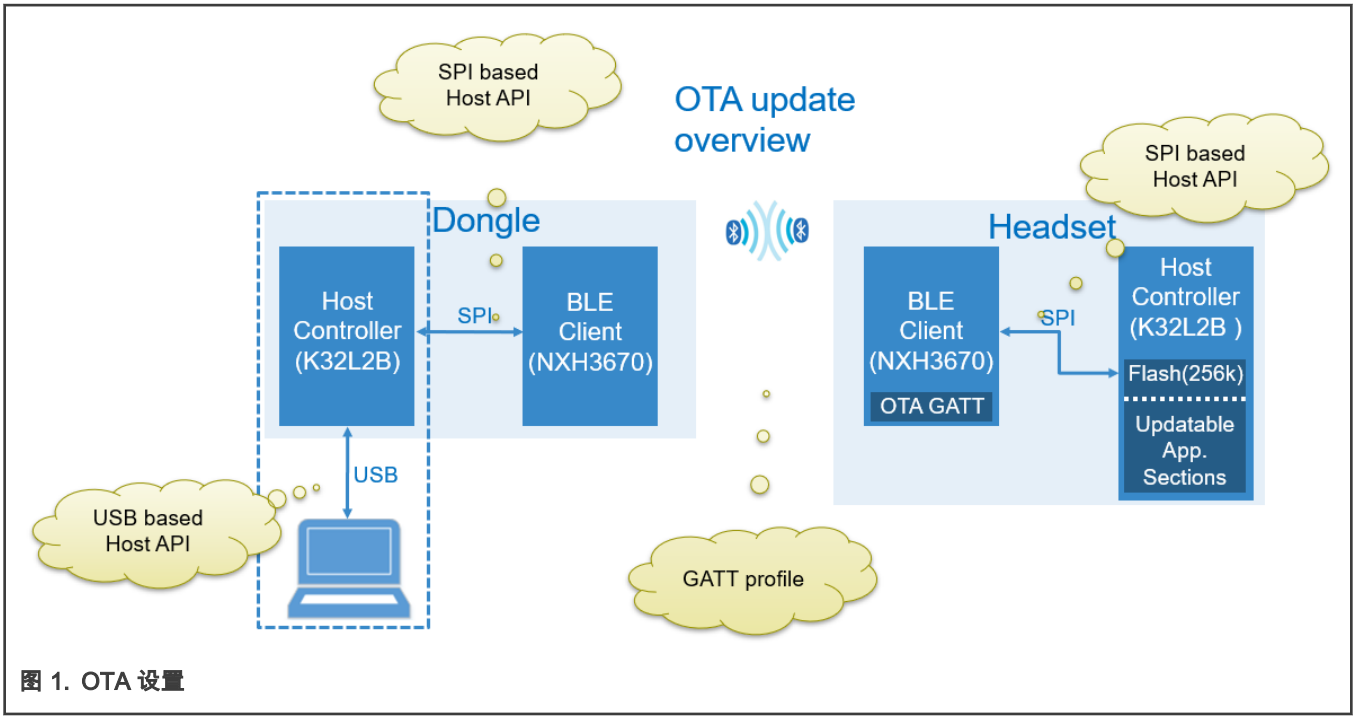

图 1. OTA 设置

此过程需要用到一个 Dongle，耳机，PC 和一根连接 Dongle 和 PC 的 USB 线。

基于 K32L2B+NxH3670 的典型应用场景描述如下：

1. 通过 USB 接口编程 Dongle 和耳机。
2. 等待 Dongle 和耳机配对完成。
   - 耳机的配对数据(PD)独立于 Dongle 的配对数据。

     一旦启动，耳机将从预先编写的内存中检索 PD，且不会存储额外的 PD。
   - Dongle 的配对数据依赖于耳机的配对数据。

     一旦启动，Dongle 检索配对数据并与耳机配对，然后将耳机设备信息存储在 Dongle 的 PD 部分（例如，分区 3：设备信息和绑定数据）。
3. 将 OTA_Dongle 应用程序下载到 Dongle 中，这是 OTA 过程的开始。
   - OTA_Dongle 可以作为 VCOM 通过 USB 在 PC 和 K32L2B 之间传输数据。
   - 在重新编程之前，应确保配对数据不被擦除。（Dongle 具有获取耳机的设备信息的功能，但 OTA_Dongle 没有这个功能）。 两个 NXH 设备首先配对，然后连接，因此如果用户删除了存储在 Dongle 的闪存中的配对数据，它们就不能建立连接。
   - 在调试模式下，代码占用了很多空间。 耳机端没有足够的空间同时存储 Headset 和 OTA_Headset 的二进制数据。因此，用户可以用 OTA_Headset 应用程序重新刷新耳机板来测试 OTA 功能。 用户需要选择“Release”模式，以确保 Flash 有足够的空间存储固件。
4. 由 PC 应用程序发起 OTA 进程。
5. OTA 完成，耳机固件更新完成。
6. 重新将游戏应用程序下载到 Dongle 中。

2.2 第二级引导程序（SSB）

SSB 由第一阶段引导加载程序（ROM Bootloader）自动引导。

用户可以根据自己的要求在 Flash 中存储多个固件，并通知 SSB 启动哪些固件。

例如，考虑到耳机板在 K32L2B 低功耗蓝牙音频系统中没有 USB 端口，开发人员应至少提前存储三个固件，包括：

- SSB：决定启动哪个固件。
- OTA 固件：接收新固件。
- 应用固件：一个实际的应用程序，包括特定的耳机功能。

以当前工程为例，SSB 函数包括：

1. 将 VTOR 设置为应用程序的中断向量表地址。
2. 将堆栈指针设置为应用程序的堆栈指针。
3. 跳转到应用程序（PC 寄存器指向的地址）。

2.3 分区表

需要将 Dongle、OTA_Dongle、Headset 和 OTA_Headset 应用程序及其位置映射到 Flash 中。此映射关系存在于主机 Flash 的固定偏移量的分区表中。有关更多信息，请参阅 HAPI OTA。

图 2 显示分区和偏移量。

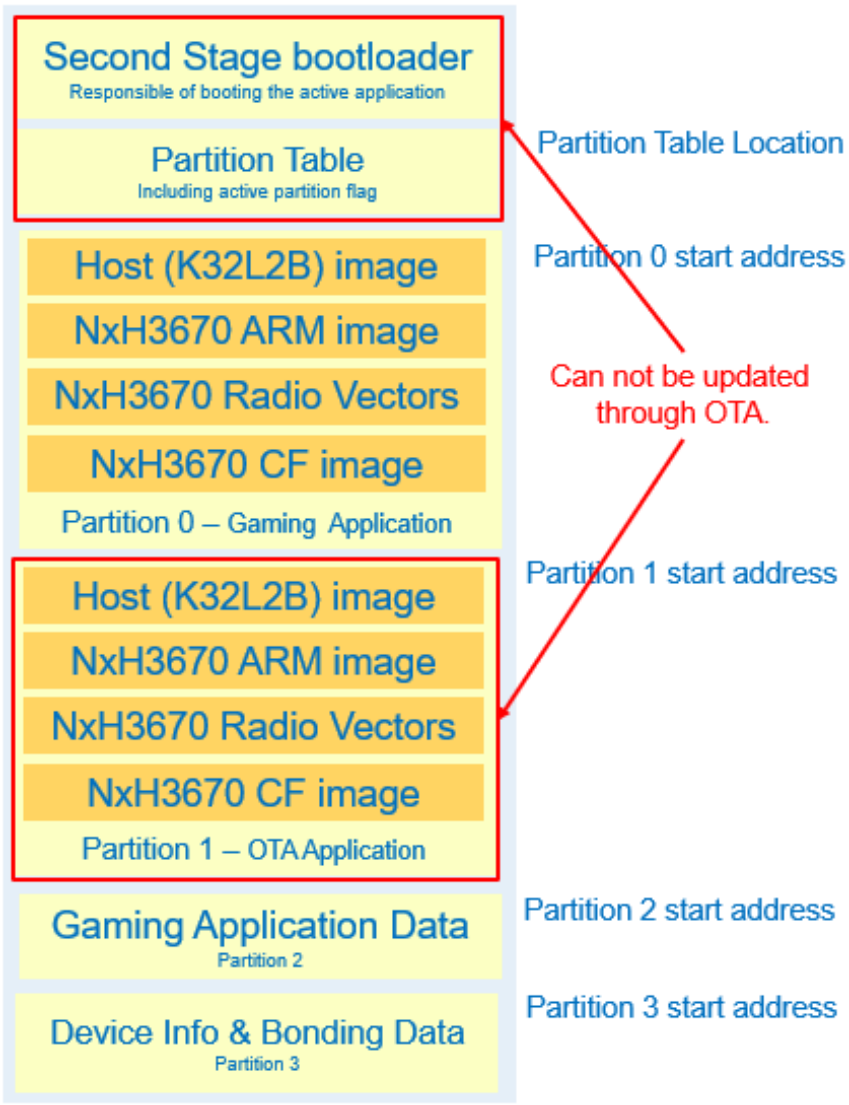

图 2. K32L2B 耳机端的 Flash 分区

- 分区 0 是游戏应用程序。 它包含 K32L2B 主机控制器固件、NxH3670 ARM Image、NxH3670 Audio Radio Vectors 和 NxH3670 CoolFlux Image。
- 分区 1 是 OTA 应用程序。 它包含主机控制器（K32L2B）固件和 NxH3670 ARM Image。
- 对于 OTA_Dongle，用户只需要刷新 `ota_app` 和 `NxH_Binary(ARM.phOtaDongle.ihex.eep)`。
  - `rfmac(rfmac.eep)` 已经添加到 Nxh 映像中。
  - 不需要 `CF(phStereoInterleavedAsrcTx.eep)`。
- 分区 2 包含应用程序数据。它用来存储游戏应用程序数据，目前尚未使用。
- 分区 3 包含设备信息和绑定数据。设备信息包含低功耗蓝牙特定属性，这些属性用来使空中通道生效。绑定数据确保 Dongle 和耳机自动重新连接，绑定数据仅与 Dongle 相关。

3 使用 FlashTool

本文档列出了如何使用.BAT 文件去方便且快速的更新固件的操作步骤。有关 Flashtool 的更多信息，请参阅 `NXH3670_SDK_Gaming_G3.0` 中的 HAPI OTA 和工具章节。

3.1 修改说明

1. `ota_update_headset.bat`

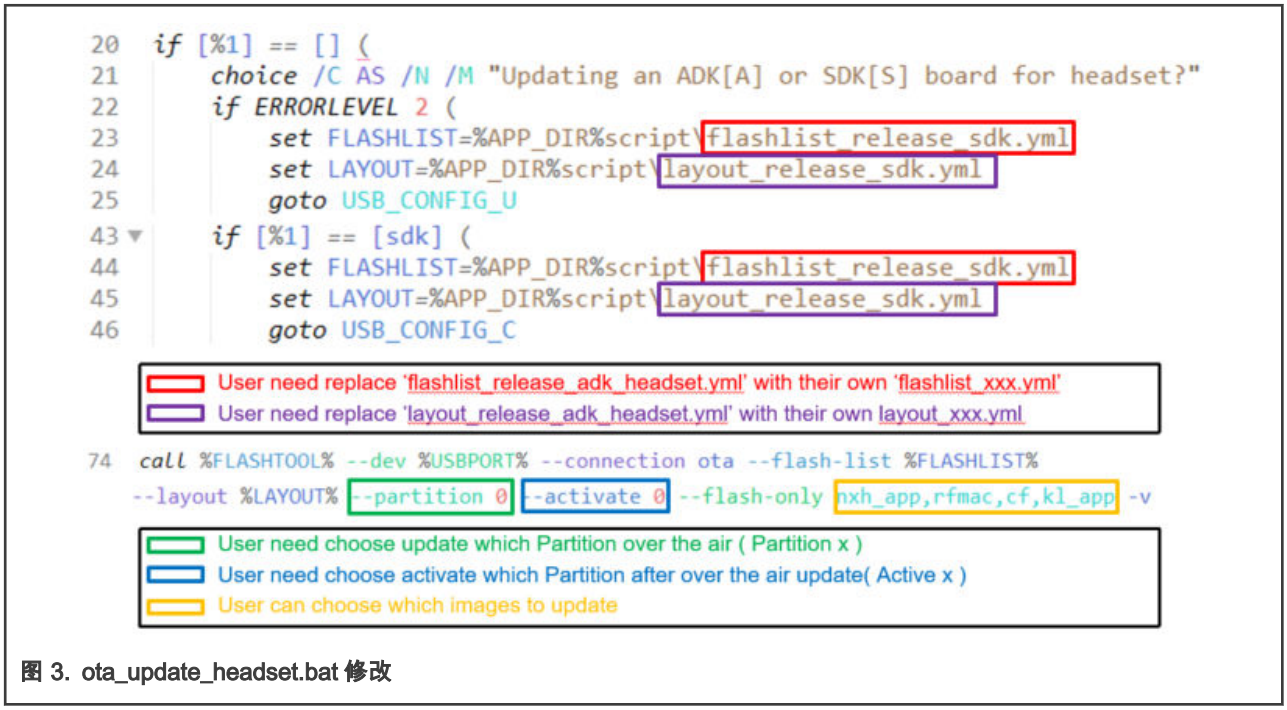

图 3. ota_update_headset.bat 修改

使用命令行窗口，执行以下步骤将分区表中的.yml 文件转换为.BIN 文件。

a. 打开命令行界面。

b. 转到 NXH3670_SDK_Gaming_G3.0 文件夹。

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

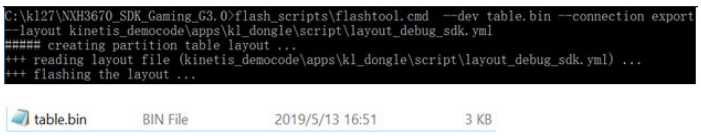

图 4. 将.yml 转换为 bin

但是，这个 `table.bin` 的前 2560(0xA00)字节都是 0x00。本文档介绍了两种处理方法。

- 在下载位于 0x00 的 SSB 之前，请确保 `table.bin` 已被下载。否则，`table.bin` 将覆盖 SSB。因此，对于 OTA 应用来说，用户还必须移植 kl_ssb 应用程序或下载 SSB 文件。
- 或者你可以删除此 `table.bin` 的 2560(0xA00)字节，然后将更改后的 `table.bin` 下载到分区表地址（在我们的软件中，分区表的地址为 0x3f400）。

2. `flashlist_release_sdk.yml`

图 5. flashlist_release_sdk.yml 修改

`flashlist_release_sdk.yml(kl_headset)`列出了要用于操作 OTA 的分区的二进制文件名和 offset_index。在此示例中，用户希望将 `kl_headset_sdk.bin.eep` 到下载到当前分区的 `offset_index_0` 中。

3. `layout_release_sdk.yml`

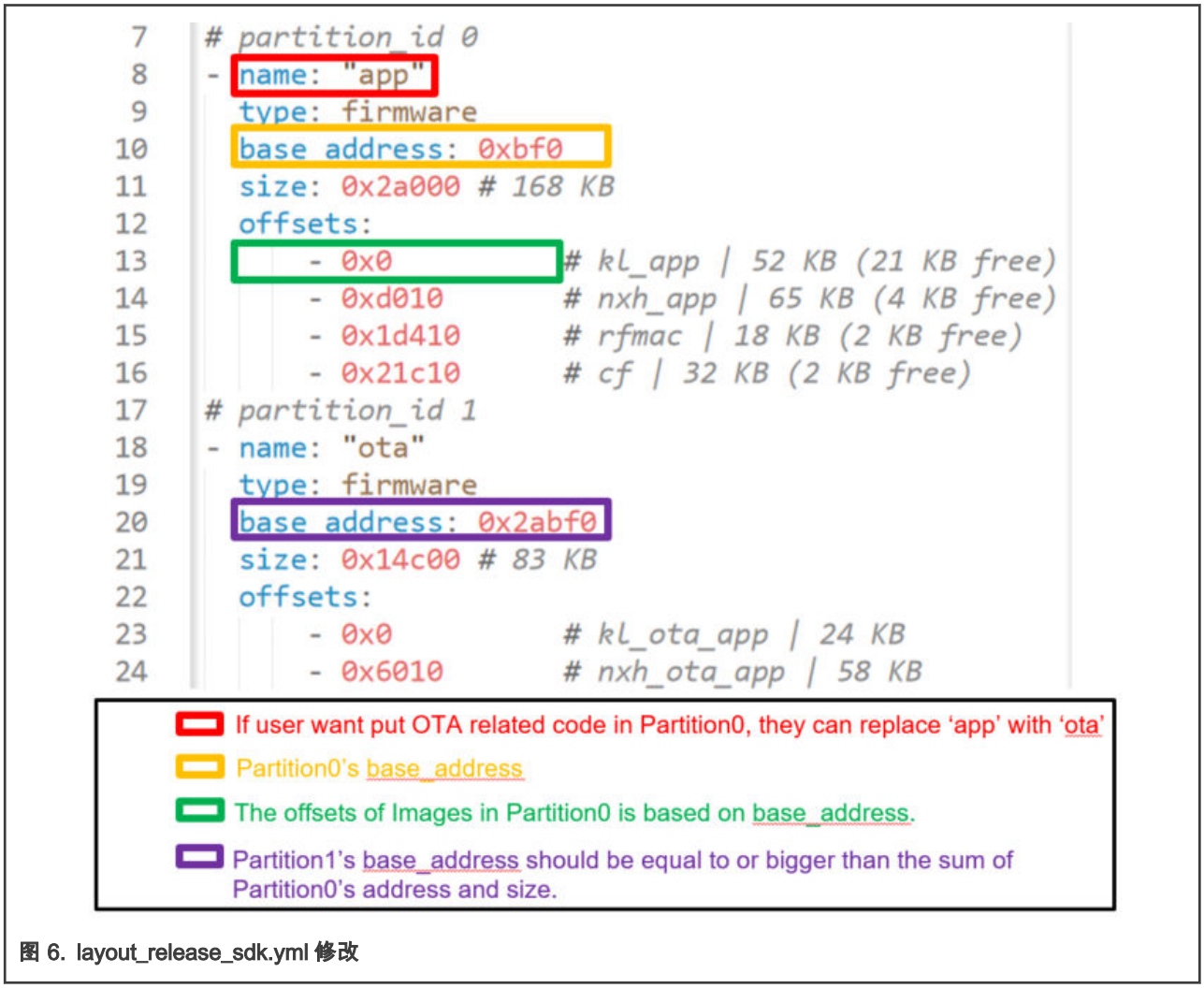

图 6. layout_release_sdk.yml 修改

用户可以设计自己的 layout_release_sdk.yml 以满足 Flash 的使用。在软件设计中，MCU 从指定位置读取 NXH_Binaries，然后通过 SPI 接口将数据传输到 NXH3670,所以请确保 Flash 布局的设计是正确的。

执行以下步骤将`.BIN`转换为 在 OTA 过程中使用的`.EEP`。

- 通过输入`-iXXX.bin-o XXX.bin.eep`来使用`...\tools\to_eep.cmd`。

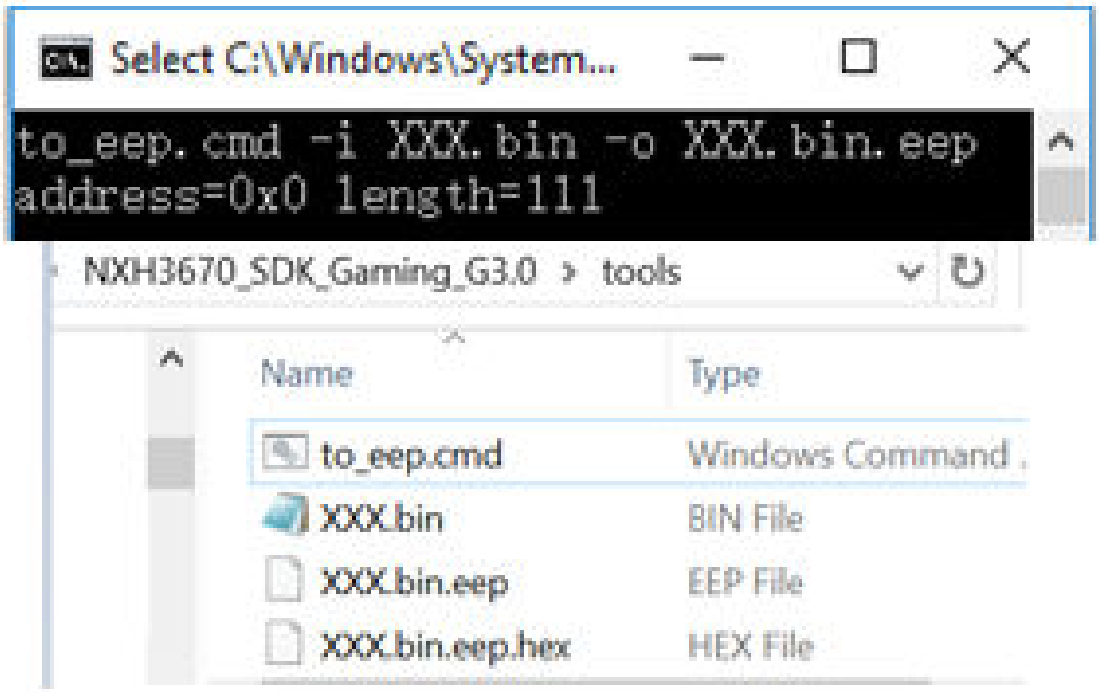

图 7. 转换 .BIN 到 .EEP

3.2 测试过程

当用户正确的修改`ota_update_headset.bat`、`flashlist_release_sdk.yml`和`layout_release_sdk.yml`时，OTA 可以工作。

假设 Dongle 和耳机已经成功配对，请按照下面的步骤进行。

1. 下载 OTA_Dongle，并确保 PC 可以识别为 USB Serial Devce。

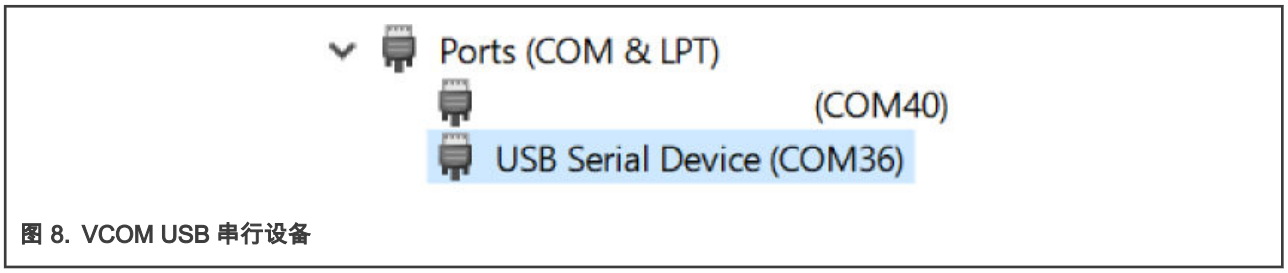

图 8. VCOM USB 串行设备

   如 图 8 所示 PC 识别它为 COM36。

2. 本文档列举两种情况：

   a. 案例一：用户正在运行 app 而不是 ota_app。

      在 Release 模式中，`APP_Partition`的`Active_flag`当前为 0（分区 0-游戏应用程序），这表明用户需要发送命令将`Active_Partition`从 0 切换到 1(分区 1-OTA 应用程序)。

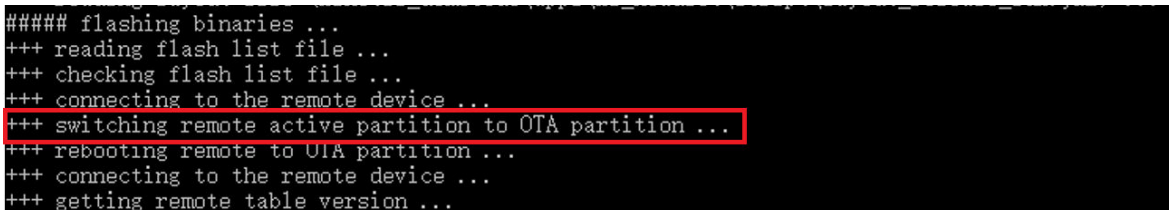

图 9. 需要切换操作

如果用户在应用程序案例中使用 phOtaHeadset.ihex.eep 而不是 phGamingRx.ihex.eep，则意味着他们已经使 NXH3670 运行了 OTA 功能（用户可以使用 OTA 相关工具与固件为 phOtaHeadset.ihex .eep 的 Dongle 板进行通信），此时不需要切换远程 Active_Partition。但是，应用程序的 hci_table 没有 OTA 相关的代码，因此不能用于操作 OTA。

b. 案例二：用户正在运行 ota_app，NXH_Binary 为 phOtaHeadset.ihex.eep。

- 在 Debug 模式下，OTA_Partition 的 Active_flag 是 1（分区 1-OTA 应用程序），表示代码已准备好用于 OTA 进程，不需要切换远程 Active_Partition。

示例是假设用户正在使用案例一。

打开命令行界面，进入 flash_scripts 文件夹。输入：

- `ota_demo_sdk.bat`（用户可以更改它并重命名它）
- 板子（本文档使用 SDK 板进行测试，因此输入 S）
- USB 端口名称（COM36）

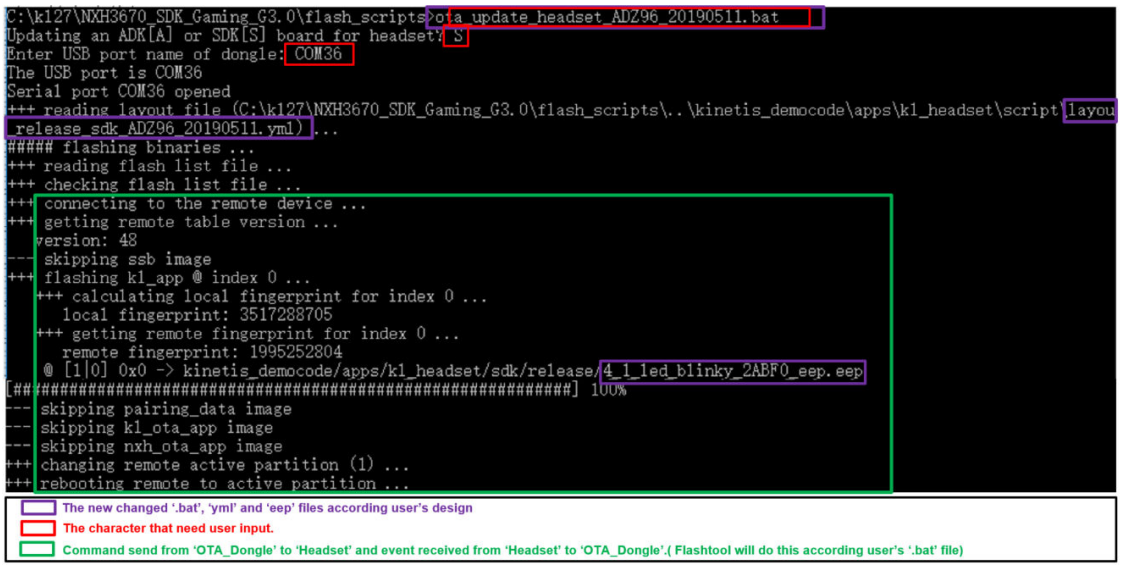

图 10. OTA 过程的 CMD

如 图 10 所示，[#.#]100%表示更新进度。

3. LOG 信息

为了更好地查看 OTA 进度，用户可以下载 OTA_Headset_Debug_mode，它可以提供如下的 LOG 信息。

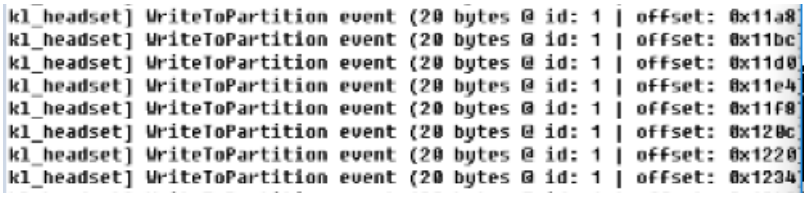

图 11. OTA_Headset

4 软件设计

为了清楚地描述软件设计，本文档附上了一些程序供用户参考。

4.1 SSB 代码

```
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();
        }
bootValidApp    PROC
        EXPORT  bootValidApp
        LDR    r1, [r0, #0]    ; Get app stack pointer
        MOV     sp,r1        ;
        LDR    r1, [r0, #4]    ; Get app reset vector
        BX    r1        ; PC now point to App_Firmware
        ENDP
```

4.2 OTA 接收代码

为了让用户轻松理解 OTA 接收过程，本节在 `OTA_Headset` 代码中提供了一个事件处理程序：`HCI_VS_WRITE_TO_PARTITION_SUB_EVENT`，介绍如何将固件写入 Flash

假设 Dongle 板正在运行 OTA_Dongle 程序，而耳机板正在运行 OTA_Heatset 程序，耳机端的 NXH3670 可以从 Dongle 接收事件，并通过 SPI 接口将事件传输到主机控制器（K32L2B）。

1. 耳机端的 NXH3670 接收从 Dongle 端的 NXH3670 发送的 `HCI_VS_WRITE_TO_PARTITION_SUB_EVENT(0Xe1)`，然后运行 `HCI_EvtWriteToPartitionHandler`。

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

2. 将数据写入请求的分区。 本文档列出了一些 API，如下所示：
   - 在对当前扇区进行写入操作之前，需要将当前扇区的所有数据拷贝至缓存扇区中。

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

   - 用户可以用 Dongle 板的 NXH3670 发送的数据修改缓存扇区。

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

   - 当一个数据包被复制到缓存扇区后，用户可以使用写 FlashAPI 编程扇区。

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

3. 到 NXH3670 的操作成功，命令如下：

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

4.3 OTA 发送代码

为了让用户轻松理解 OTA 发送过程，本节使用伪代码来介绍 OTA_Dongle 如何向 OTA_Headset 发送固件。

```
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;}
}
```

注意

- **案例** `HCI_CMD_VS_CONNECT_OPCODE`

  这个 CMD 表明 OTA_Dongle 想要连接 OTA_Headset。

- **默认** CMD

  OTA_Dongle 将使用 NXH3670 将任何其他 CMD 发送给 OTA_Headset。实际上，耳机端的 MCU 负责将这些数据写入 Flash。

5 结论

用户可以使用 Flashtool 和 `NXH3670_SDK_Gaming_G3` 中的文件来实现程序更新。可能需要根据设计需求进行一些修改。通过 OTA 的固件更新速度约为每秒 1KB。

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: 2020 年 1 月

Document identifier: AN12649

arm