AN12647

带有 NxH3670 的 K32L2B USB 适配器（Dongle）

版本 0 — 2020 年 1 月

应用笔记

目录

1 简介

1.1 概述

Dongle 通过 USB 接口连接到 PC 机上，它用于建立与耳机之间的无线音频连接。功能包括：

- 发送：将音频流从 PC 传输到耳机。
- 接收：接收耳机端发送至 PC 的控制信号和音频数据。
- 空中下载（OTA）：作为 VCOM 设备，将固件文件从 PC 传输到正在运行 OTA_Headset 固件的耳机。

为使读者系统地了解 K32L2B 蓝牙低功耗（Bluetooth LE）音频系统中的 Dongle，本文档介绍了硬件设计和软件架构。

1.2 参考文献

表 1. 参考文献

参考文献	描述
K32L2B Bluetooth LE Audio System	K32L2B 低功耗蓝牙音频系统简介
K32L2B Headset	带有 NXH3670 的 K32L2B 耳机
K32L2B OTA	K32L2B 低功耗蓝牙系统 OTA 操作步骤
K32L2B Emulating the I²S Bus	用 FlexIO 模块模拟 I²S 总线主控器

2 系统概述

2.1 框图

本演示板可以支持 Dongle 和耳机的配置。

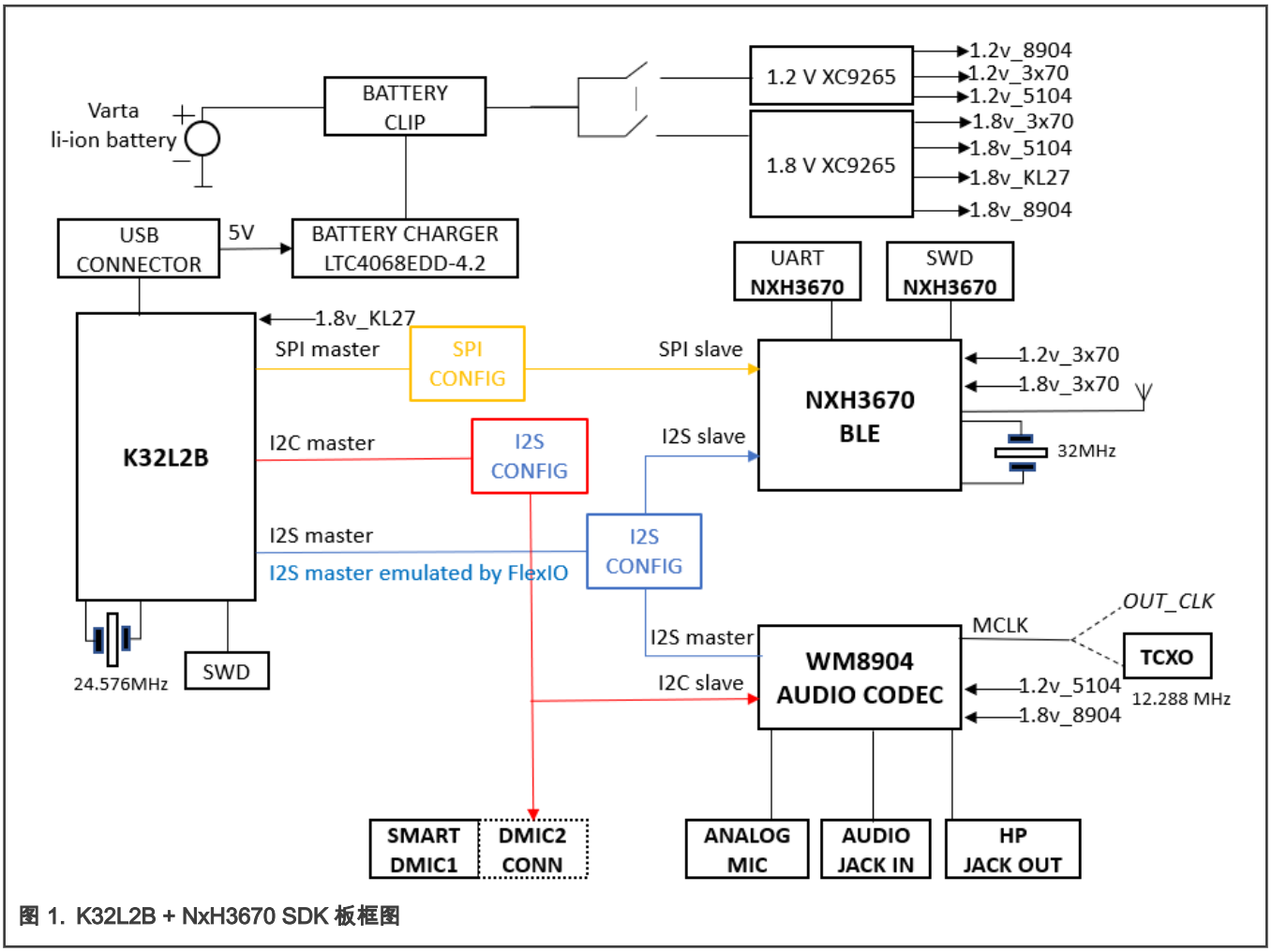

图 1. K32L2B + NxH3670 SDK 板框图

如 图 1 所示，**SPI CONFIG**、**I2C CONFIG** 和 **I2S CONFIG** 指出了通信接口的主机选择。例如，对于 **I2S CONFIG**，在 Dongle端选择 **K32L2B** 作为 **I²S 主控器**，而在耳机端选择 **WM8904** 作为 **I²S 主控器**。

注意

考虑到 K32L2B 不包含 I²S 外设，用户可以配置 FlexIO 外设以生成所有必需的 I²S 总线信号。

图 2 展示了 K32L2B_Dongle 的框图。

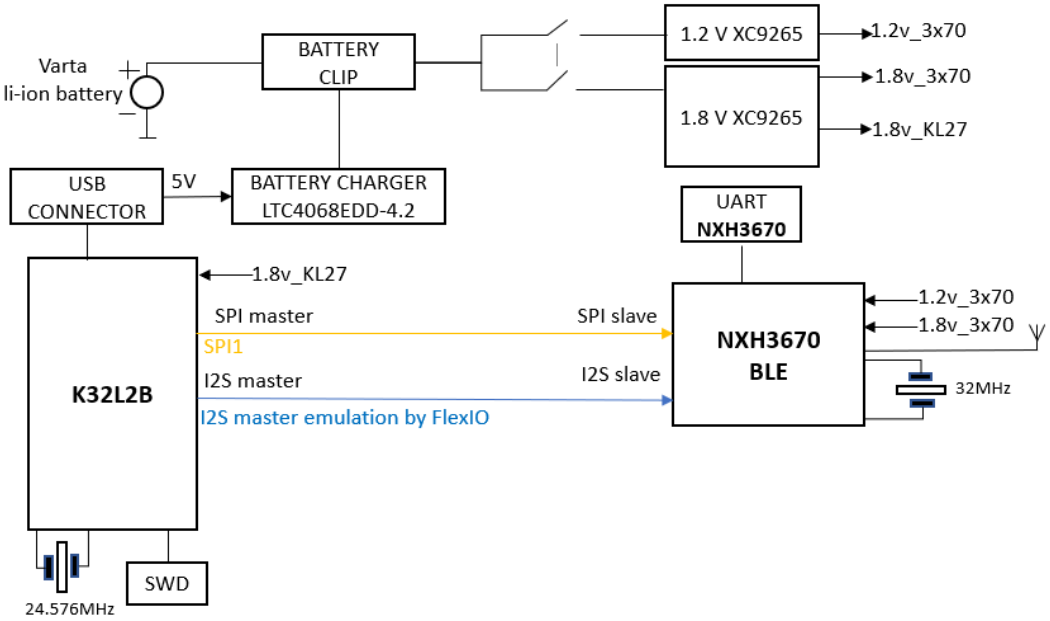

图 2. Dongle 配置框图

如 图 2 所示，该系统包括：

- 主机控制器（K32L2B），用于运行 Dongle 和 OTA_Dongle 例程。
- NXH3670 通过 SPI 接口与 K32L2B 通信，并通过 FlexIO 外设模拟的 I^2S 总线信号传输音频流。

2.2 USB Dongle 软件架构

图 3 和 图 4 展示了软件架构。

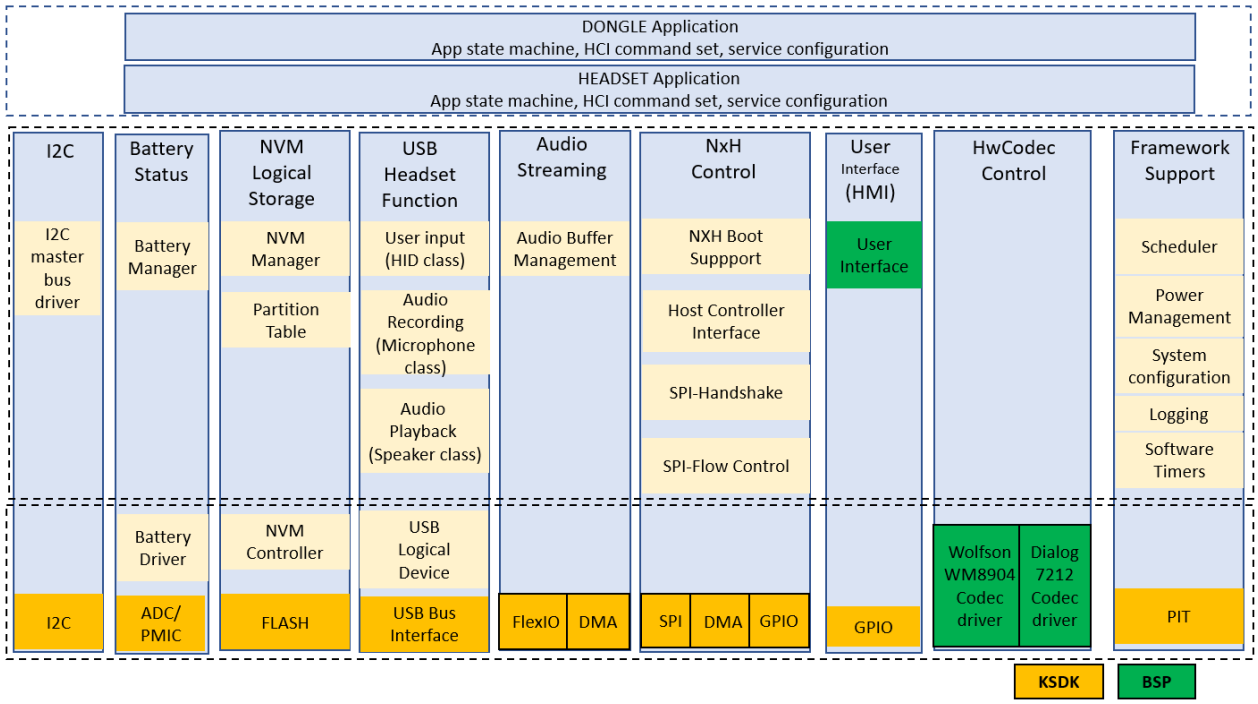

图 3. 应用架构

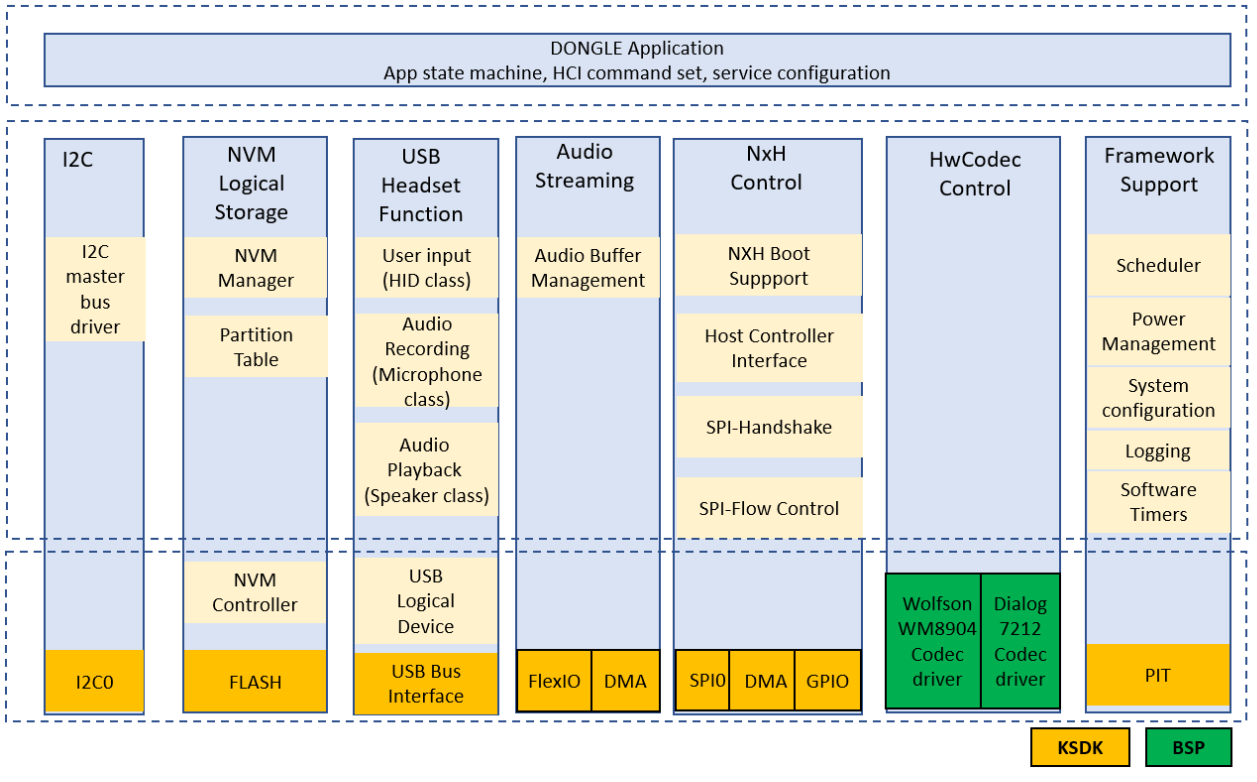

图 4. Dongle 应用架构

该架构包含 **NVM 服务、USB 服务、音频服务、NXH 服务**和**用户界面（UI）服务**。 本文档列出了以下功能：

1. **NVM** 服务：用于读取分区表。

2. NxH 控制：使用 SPI 接口去启动 NxH3670,并实现 NxH3670 和 K32L2B 之间的数据传输。
3. UI: 是指用于音量控制、播放和暂停的按键。
4. 音频服务：将音频数据传输到 FlexIO 的 SHIFTBUF0。

注意

使用 DMA 通道将音频数据从环形缓存区直接移动到 FlexIO 的 **SHIFTBUF0**，而无需软件干预。

5. USB 控制器：USB 被配置为用户音频接口（UAC）。

图 5 展示了音频的传输过程。

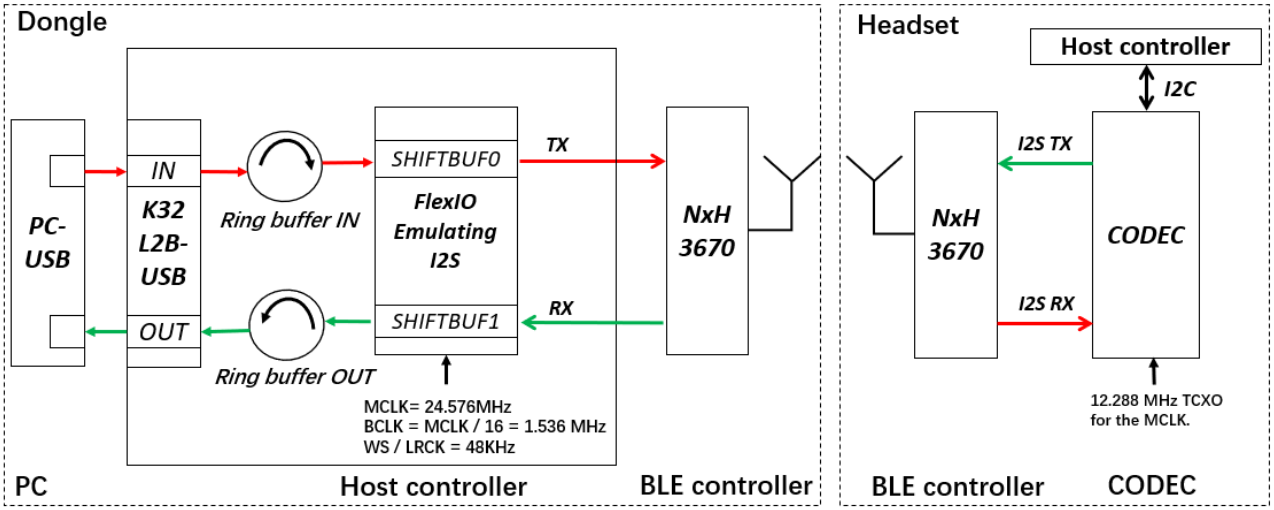

图 5. 音频路径

- →

  播放（前向通道）：音频路径是从 PC 到耳机。

  一旦传输完成，主机的 USB 控制器将产生一个中断。

  — 应用程序会将 USB 协议栈中的传输数据复制到循环缓存区（输入缓存区）中。该缓存区将对音频数据进行排序，直到其缓存区容量的一部分被填满为止，例如 50%。

  — 应用程序将启用 DMA 通道，将音频数据从循环缓存区传输到 FlexIO 的 **SHIFTBUF0**，而无需软件干预。

  — Dongle 端的主机控制器通过 FlexIO 外设模拟的 I^2S 信号与 NxH3670 相连。然后，NxH3670 将音频数据通过无线的方式传输至耳机端的 NxH3670。

  — 耳机端的低功耗蓝牙控制器通过 I^2S 与 CODEC 连接。将接收到的 I^2S 数据传输至 CODEC。

- ←

  录音（后向通道）：音频路径是从耳机到 PC。

  音频通过 **LINE IN** 或 DMIC 输入到耳机端的 CODEC。然后，NxH3670 将接收到的音频数据传输至 Dongle 端的 NxH3670，而无需主机控制器干预（当前，CODEC 是 I^2S 的主机）。

  — DMA 通道会将接收到的音频数据从 FlexIO 外设的 **SHIFTBUF1** 传输到循环缓存区。

  — 应用程序会将音频数据从循环缓存区拷贝到 USB 协议栈。

本文档仅介绍 Dongle 部分的音频传输过程。有关“耳机”部分的更多信息，请参阅《带有 NXH3670 的 K32L2B USB 耳机》（文档 AN12648）。

3 USB Dongle 的组成

3.1 K32L2B

3.1.1 主机控制器（K32L2B）

该器件是一款基于增强型 Cortex-M0 +（CM0 +）核平台的高度集成的、市场领先的超低功耗 32 位微控制器。带有 NXH3670 的 K32L2B USB Dongle 包含以下功能：

- 内核时钟最高为 48 MHz，总线时钟最高为 24 MHz。
- 内存选项最大可提供 256 KB Flash 和 32 KB RAM。
- 足够宽的工作电压，范围为 1.71 至 3.6 V，具有功能齐全的 Flash 编程/擦除/读取操作。
- 两个支持 16 位数据长度的 SPI 模块。
- 两个内部集成电路（I^2C）模块。
- 一个 FlexIO 模块。

3.1.2 时钟

1. 板上使用一个参考时钟。
   - 与 NxH3670 连接的 32 MHz 晶振。
2. FlexIO 模块充当产生所有所需信号的 I^2S 总线主机。
   - BCLK 为 1.536 MHz。
   - Word Select（WS）/Left-Right 时钟（LRCK）为 48 KHz。

   图 6 展示了正确配置 FlexIO 外设后的时钟信息。

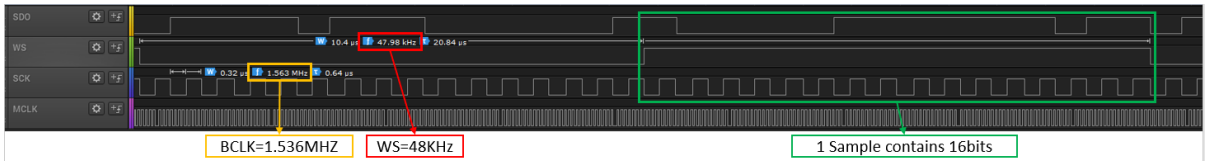

图 6. 音频传输过程中的时钟信息

3.1.3 引脚连接

表 2 列出了 K32L2B 和其他组件的引脚连接。

表 2. 引脚连接

功能	跳线	名字	跳线	名字
	K32L2B Dongle	KL2X_I2S_MASTER	NXH3670	BLE_I2S_SLAVE
FlexIO 模拟的 I^2S（与 MCU 连接）	J2-8 (PIN PTD6)	KL2X_SDI	J12_1 (I2S_CONFIG)	BLE_SDO
	J1-6 (PIN PTD3)	KL2X_SDO	J12_3 (I2S_CONFIG)	BLE_SDI
	J2-12 (PIN PTD5)	KL2X_WS	J12_5 (I2S_CONFIG)	BLE_WS

下页继续...

表 2. 引脚连接（续上页）

功能	跳线	名字	跳线	名字
	K32L2B Dongle	KL2X_I2S_MASTER	NXH3670	BLE_I2S_SLAVE
	J2-6 (PIN PTD4)	KL2X_SCK	J12_7 (I2S_CONFIG)	BLE_SCK
NXH 握手	J1_2 (PIN PTA1)	BLE_SPIS_INTN	J16_9 (BLE_SPI)	SWM4 (-INTN)
	J1_8 (PIN PTA12)	BLE_SPIS_SRQ	J16_11 (BLE_SPI)	SRQ
SPI (SPI0)	J1-11 (PIN PTC7)	BLE_SPIS_MISO	J16_1 (BLE_SPI)	SW0
	J1-9 (PIN PTC6)	BLE_SPIS_MOSI	J16_3 (BLE_SPI)	SW1
	J1-15 (PIN PTC5)	BLE_SPIS_SCLK	J16_5 (BLE_SPI)	SW2
	J1-7 (PIN PTC4)	BLE_SPIS_SSN	J16_7 (BLE_SPI)	SW3
NXH 复位	J1_4 (PIN PTA2)	BLE_RESETN	J20_5 (BLE_SWD)	POR_RESETN

3.1.4 原理图

1. 音频传输
   - I^2S

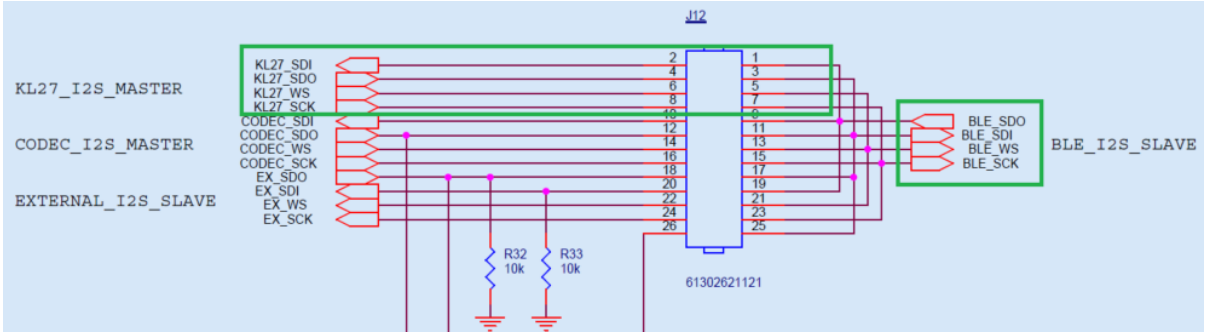

图 7. I^2S 接口连接

在 Dongle 部分，主机控制器（K32L2B）通过 FlexIO 外设模拟的 I^2S 总线信号将数据直接传输到 NXH3670。

2. NXH3670
   - NXH 握手

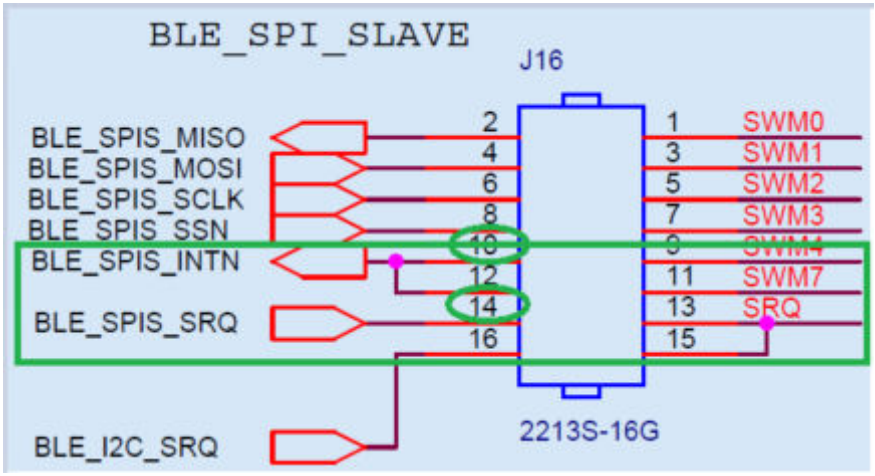

图 8. 握手引脚（SRQ 和 INTN）连接

- SPI

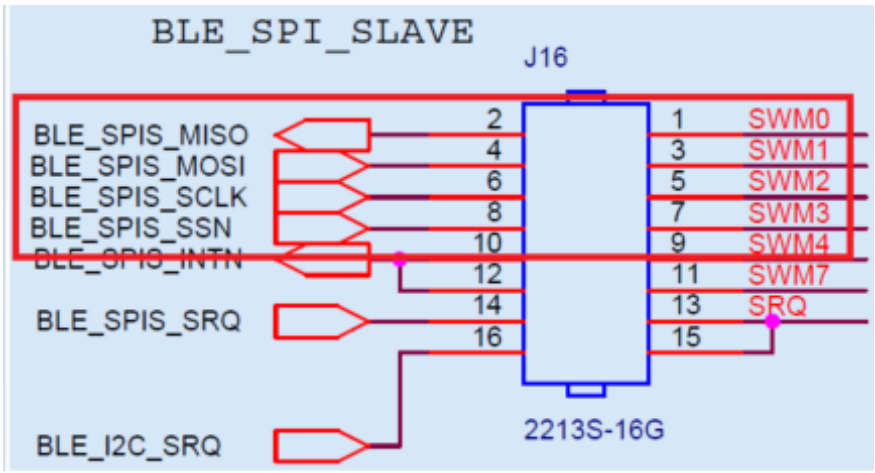

图 9. SPI 接口连接

- 上电复位（POR）

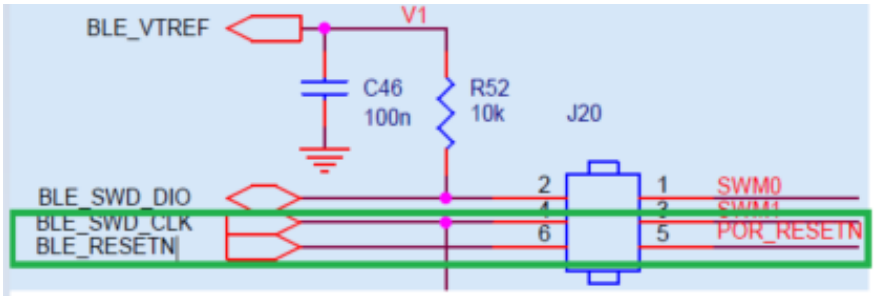

图 10. POR 引脚连接

3.1.5 引脚配置

- SPI

  - 接口：SPI0
  - 引脚：CS（PTC4），SCK（PTC5），MISO（PTC7），MOSI（PTC6）
  - 极性：高电平有效的 SPI 时钟（空闲时为低电平）。
  - 时期：SPSCK 的第一个边沿为采样点。
  - 波特率：SPI 的波特率值配置为 8000000。
- 用于模拟 I^2S 的 FlexIO 引脚
  - TXD: PTD3
  - RXD: PTD6
  - BCLK: PTD4
  - FS: PTD5
- NxH3670 引脚
  - INIT（PTA1），配置为数字输入。
  - SRQ（PTA12），配置为数字输出。
  - POR（PTA2），配置为数字输出。

3.2 NXH3670

3.2.1 低功耗蓝牙

NxH3670 是低功耗蓝牙（Bluetooth LE）设备。它是一款内置 MCU 的单芯片超低功耗 2.4 GHz 收发器，目标市场为耳机、听筒中的无线音频流。

功能包括：

1. 主要特点
   - 支持高质量、低延迟（<20 ms）的无线音频流。
   - 集成的无线音频流解决方案
     - 集成的 Arm Cortex -M0 处理器
     - 集成的 CoolFlux DSP 和 HW 加速器，用于音频处理
   - 超低功耗运行：
     - 前向通道为立体声后向通道为单声道工作时功耗为 7.5 mW
   - 封装为凸块模具 < 7.25 mm^2
   - 典型电源电压：1.2 V
2. 专有音频流协议

   NxH3670 设备可以支持标准的低功耗蓝牙协议和 NXP 的专有音频协议。此音频流协议被配置为支持 Dongle 设备和耳机之间的音频流，音频配置如下：
   - 前向音频路径
     - 立体声
     - 48 KHz 采样率，16 位分辨率
     - 音频带宽 > 20 KHz
     - SBC HQ 编解码器
     - 延迟 < 20 ms

- 后向音频路径
  - 单声道
  - 16 KHz 采样率，16 位分辨率
  - 音频带宽 > 6 KHz
  - G.722 编解码器

在音频流传输期间，耳机端的功耗仅为 7.2 mW，从而延长了播放时间并减小了电池尺寸。在音频流传输期间，音频源和音频接收器之间可以同时进行双向数据连接，并且吞吐量高达 8 kbps。

图 11 展示了 Dongle 和 NXH3670 耳机之间的连接过程。

a. 下载并启动 NxH3670 Image。
   - 在引导阶段，主机控制器通过 SPI 接口将 Image 从 flash/eeprom 加载到 NXH3670。
   - 在启动阶段，主机控制器需要与 NXH3670 握手。然后，软件将会在 HCI 层注册一个事件表，用于处理从 NXH3670 发送的事件。

b. 配对。

   Dongle 和耳机上的 NXH3670 将相互配对。例如，Dongle 将从耳机端恢复配对数据（Persistent Data）。

c. 连接。

   Dongle 和耳机上的 NXH3670 将相互连接，如果连接成功，可以彼此之间传输数据。例如，Dongle 可以将音频流发送到耳机。

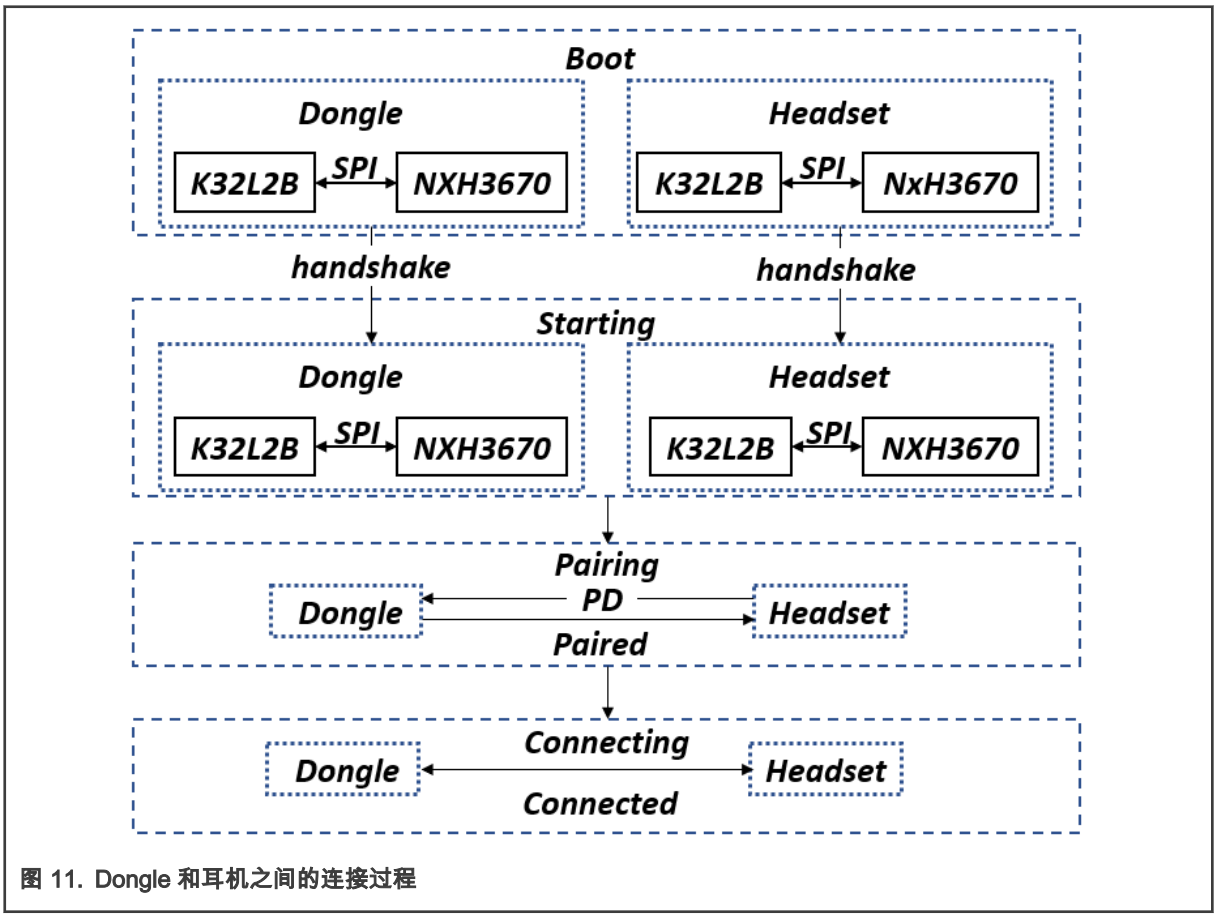

图 11. Dongle 和耳机之间的连接过程

3.2.2 启动

3.2.2.1 NXH3670UK 启动加载程序

引导程序的最重要任务是准备 NXH3670UK 以启动用户应用程序。典型的启动加载程序的运行流程为：

1. 配置设备。
2. 加载内存。默认情况下，NXH3670UK 以从机模式启动：SPI 从机接口。
3. 进入活跃模式。

3.2.2.2 分区表

由于 NxH3670 无法永久的存储数据，因此使用主机控制器的 Flash 进行存储。

该参考应用程序具有将内存划分为逻辑分区的功能。

可以读取或写入此类分区中的数据包括：固件二进制数据或应用程序配置数据。

layout_release_sdk.yml

```
# number of partitions: 3
# table address: 0xa00
# max used size (including ssb and table): 256 KB
active_partition: 0
layout_version: 0x30
entries:
    # partition_id 0
    - name: "app"
      type: firmware
      base_address: 0xbf0
      size: 0x3ec00 # 251 KB
      offsets:
          - 0x0          # kl_app | 130 KB (95 KB free)
          - 0x20810      # nxh_app | 65 KB
          - 0x30c10      # rfmac | 16 KB
          - 0x34c10      # cf | 39 KB
    # partition_id 1
    - name: "app_data"
      type: appdata
      base_address: 0x3f800
      size: 0x400 # 1 KB
      offsets:
          - 0x0          # app_data | 1 KB
    # partition_id 2
    - name: "pairing_data"
      type: appdata
      base_address: 0x3fc00
      size: 0x400 # 1 KB
      offsets:
          - 0x0          # pairing_data | 1 KB
```

图 12. Dongle Rlease 模式的分区表

如 图 12 所示，partition_id 0 包含四个 Image，kl_app 、nxh_app 、rfmac 和 cf。例如，nxh_app 的偏移量为 0x20810，这表示该 Image 将被下载到 0x21400(0xbf0+0x20810)。

用户可以设计自己的分区表，以下两个注意事项很重要：

1. 如果用户希望 Partition1: app_data 作为内存中的第一个分区，请在 layout_release_sdk.yml 文件中，将顺序保持为：app_data ，app ,....

   如果用户不遵循该规则，则该工具将不会输出 Partition Table.bin。

2. 用户必须确保 base_address of Partition + size of partition0 的大小小于 base_address of the Partition1 。

3.2.2.3 NVM

非易失性内存（NVM）是一种在断电期间保持存储数据的存储技术。Flash 是一种使用 NOR 型闪存技术的 NVM。

K32L2B 的 NVM 可用于保存 NXH3670 的固件。 以 Dongle 为例，用户需要预先在 NVM 中存储 phGamingTx.ihex.eep、phStereoInterleavedAsrcTx.eep 和 rfmac.eep ，这大约需要 120 K。

3.2.2.4 EEP

1. 定义

   出于安全原因，NVM Image 在不同级别（即模块级别和总体级别）都包含 CRC 和签名。该功能有助于检测损坏的 Image，并中止加载和执行有潜在风险的指令。

表 3. EEP 文件的格式 - 单个 Image（LSB 优先）

	1 byte	1 byte	1 byte	1 byte
SIGNATURE	0xCA	0xFE	0xBA	0xBE
HEADER	Image Length			Type
	Destination Address			
	CheckSum			
Image Length Size	Program Image			
HEADER	Image Length = 0			Type
	Destination Address			
	CheckSum			

表 4. EEP 文件的格式 - 多个 Image（LSB 优先）

	1 byte	1 byte	1 byte	1 byte
SIGNATURE	0xCA	0xFE	0xBA	0xBE
HEADER	Image Length			Type
	Destination Address			
	CheckSum			

下页继续..

表 4. EEP 文件的格式 - 多个 Image（LSB 优先）（续上页）

	1 byte	1 byte	1 byte	1 byte
Image Length Size	Program Image			
HEADER	Image Length = 0			Type
	Destination Address			
	CheckSum			
Image Length Size	Program Image			
HEADER	Image Length = 0			Type
	Destination Address			
	CheckSum			

表 3 和 表 4 中列出的所有字段均以小端模式存储。有效 Image 必须以 32 位签名 0xBEBAFECA 开头。签名后，可以显示一个或多个 Image。每个 Image 都有一个文件头部。

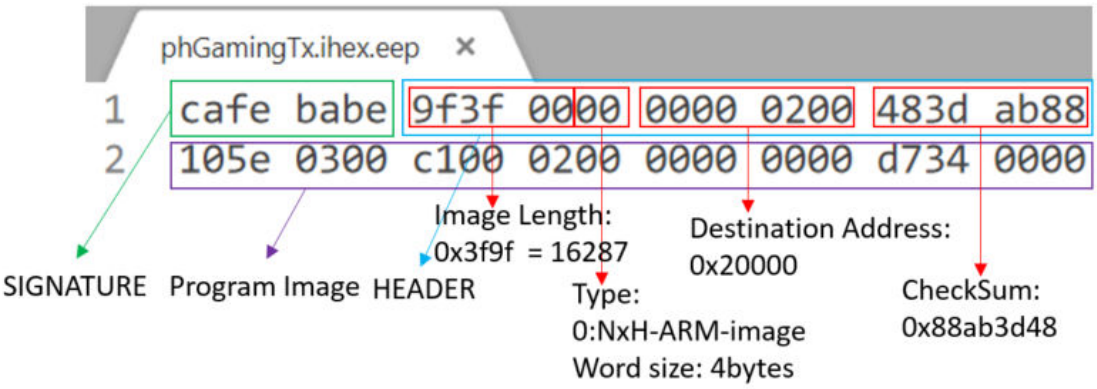

图 13. phGamingTx.ihex.eep 分析 - 第一个 Image

如 图 13 所示，Image 长度为 16287，类型 ID 为 0。这表示主机控制器将通过 SPI 向 NXH3670 发送 65148（16287 * 4）字节。

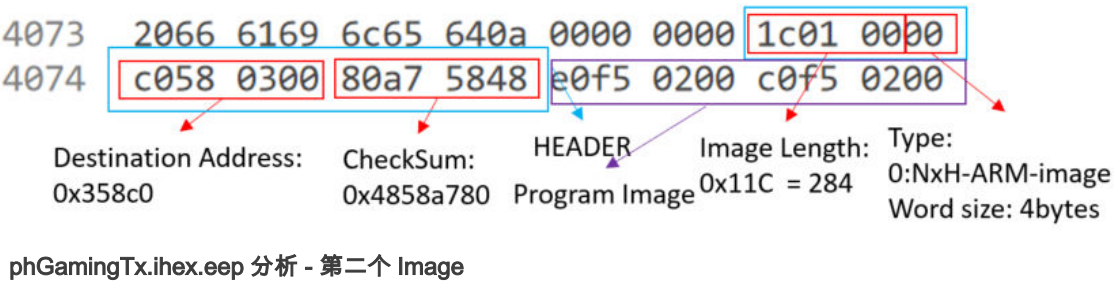

图 14. phGamingTx.ihex.eep 分析 - 第二个 Image

如 图 14 所示，Image 长度为 284，类型 ID 为 0。这表示主机控制器将通过 SPI 向 NXH3670 发送 1136（284 * 4）字节。

2. 将 `.EEF` 文件下载到 K32L2B3

本文档提供了两种方法来将 .EEF 文件下载到 K32L2B3。

a. 将 .EEF 文件转换为 HEX 缓存区。

- Winhex
- __attribute__ ((section (.ARM.__at_address)))

该方法有助于将 NXH3670 相关固件存储为数组，因为 OTA 操作仅重写了主机控制器的应用程序，而没有重写 NXH3670。

b. 将 .BIN 文件转换为 .EEP 文件。

- SDK 包在发行版中具有一个名为 to_eep.cmd 的工具。

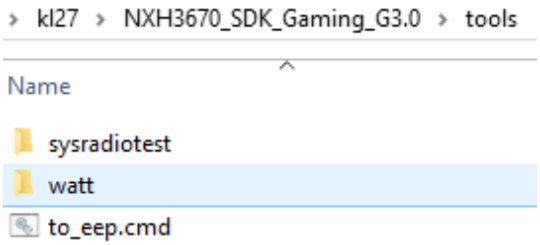

图 15. SDK 包中的 to_eep.cmd

- 输入

```
to_eep.cmd -i spi_dma_b2b_transfer_master.bin -o spi_dma_b2b_transfer_master.eep
```

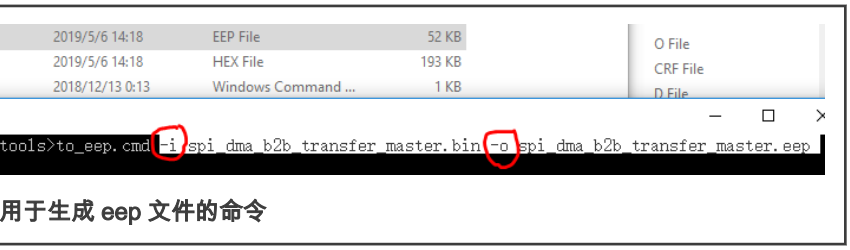

图 16. to_eep.cmd 中使用的用于生成 eep 文件的命令

- 如 图 17 所示，文件包括了签名和头文件。

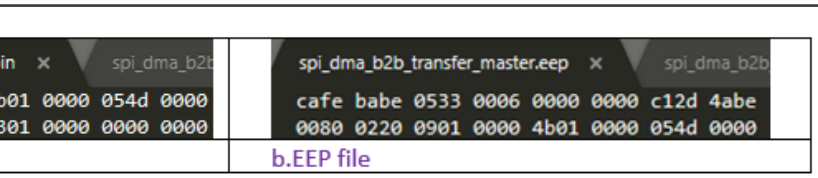

图 17. .BIN 文件和 .EEP 文件

此方法有助于存储主机控制器的应用程序。例如，用户可以使用 CRC 和签名将应用程序的 .BIN 文件转换为 .EEP.BIN 文件，这对于 OTA 处理很有用。

3.2.2.5 NXH3670 主机接口：SPI

1. SPI 总线

对于 NXH3670，引导程序会将其配置为 SPI 从机接口，并假定主机为 SPI 主设备。SPI 从机操作模式配置如下：

- SPI 从属四线模式连接：MOSI，MISO，SCK，SSEL
- SPI 从机最大通信速度：8 MHz
- SPI 从机模式：模式 0（CPHA = 0，CPOL = 0）

工作模式：时钟和相位选择。

SPI 接口通常允许配置时钟相位和极性。如 表 5 和 图 18 所示，CPOL 和 CPHA 由 SPI 控制寄存器 1（`SPIx_C1`）配置。

表 5. SPI 模式总结

CPOL	CPHA	SPI 模式	描述	SCK 休息状态	SCK 数据改变边沿	SCK 数据取样边沿
0	0	0	SPI 在传输的第一个时钟转换（当时钟从静止状态改变时）捕获串行数据。数据在下降沿更新。	低电平	下降沿	上升沿
0	1	1	SPI 在传输的第一个时钟转换（当时钟从静止状态改变时）更改串行数据。在下降沿捕获数据。	低电平	上升沿	下降沿
1	0	2	与 SCK 反转的模式 0 相同。	高电平	上升沿	下降沿
1	1	3	与 SCK 反转的模式 1 相同。	高电平	下降沿	上升沿

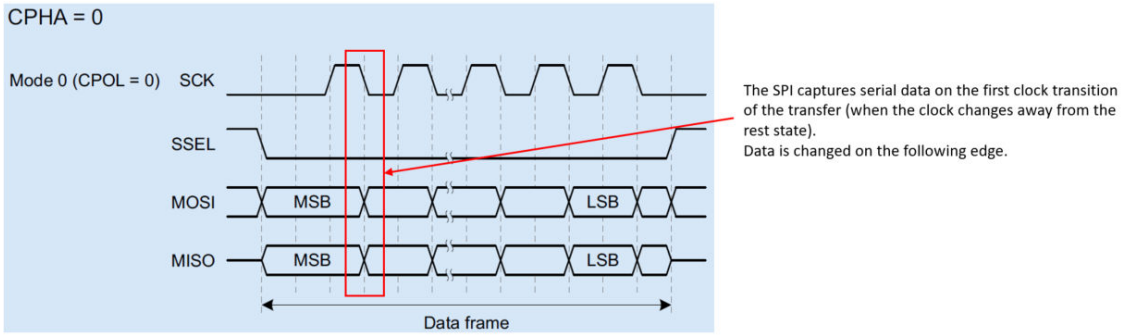

图 18. SPI 基本工作模式：mode0

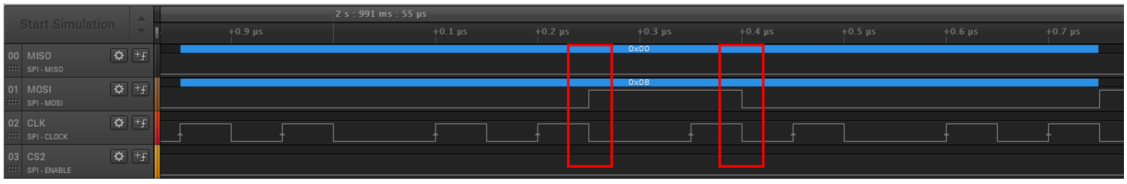

图 19. mode0 时的逻辑分析仪示例

2. SPI 流控

在低功耗蓝牙音频系统中，SPI 传输必须符合 图 20 所示的结构。

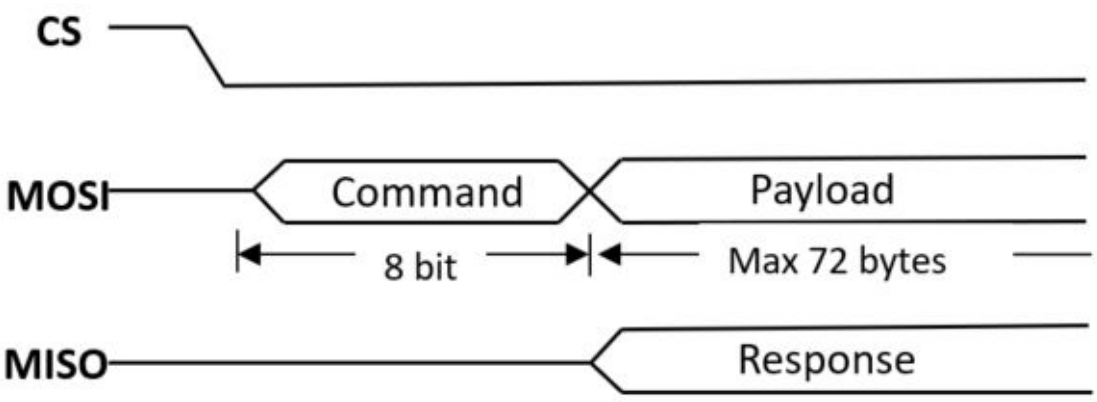

图 20. SPI 从机–传输格式

表 6. SPI 从机–支持的指令

指令	Opcode	描述
写指令	0b010xxxxx	将有效数据写入 NxH3670 SPI 从机。
读指令	0b110xxxxx	从 NxH3670 SPI 从机读取 Pending 数据。
读状态	0b101xxxxx	读取状态字节。
读扩展状态	0b111xxxxx	读取扩展状态字节。

例如，写指令由软件设计中的 `#define SPI_WRITE_CMD (0x40u)` 定义。为了使用户易于理解 SPI 传输，本文列出了逻辑分析仪的信号，如 图 21 所示。

图 21. 传输模式

3.2.2.6 HCI 指令格式

1. HCI 指令格式

   HCI 指令嵌入在 SPI 写入指令的 SPI payload 字段中（详见 SPI 流控）。HCI 指令的开头必须与 SPI 传输的开头对齐。所有指令和事件均需要按照 Bluetooth HCI 供应商的特定指令格式，如 图 2 所示。

Bit 0	8	16	24	31
HCI Packet type (command = 0x01)		Opcode		ParamLength
Param0		Param2		Param...
...				
Parameter N				

图 22. HCI 指令格式

指令以下面内容开头：

- 命令包字节，固定值为 `0x01`。
- 唯一的 16 位操作码，用于标识 HCI 指令。
- 参数长度为后面参数的长度（以字节为单位），也允许零值。
- 实际参数（可选）。 这些实际参数可以存在 8 位、16 位、24 位和其他参数。

指令处理器必须正确解释字节序列。

命令中的 HCI 操作码和可选参数字节始终是 LSB 在前。

SPI 从机—逻辑分析仪的传输格式如 图 23 所示。

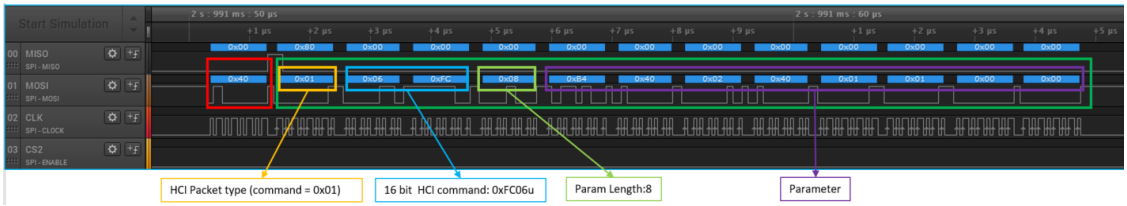

图 23. SPI 从机—逻辑分析仪的传输格式

2. HCI 事件格式

   指令的结果作为 HCI 事件发送回来。一旦 HCI 控制器将某些内容发送回主机，它就会将该事件排入队列，主机会检索队列中的事件。

   HCI 事件嵌入在 SPI 读取指令的 SPI payload 字段中。

Bit 0	8	16	24	31
HCI Packet type (event= 0x04)		Event code	Param Length	Param 0
Param 1		Param 2	Param 3	Param...
...				
Parameter N				

图 24. HCI 事件格式

图 25 展示了逻辑分析仪的 HCI 事件格式。

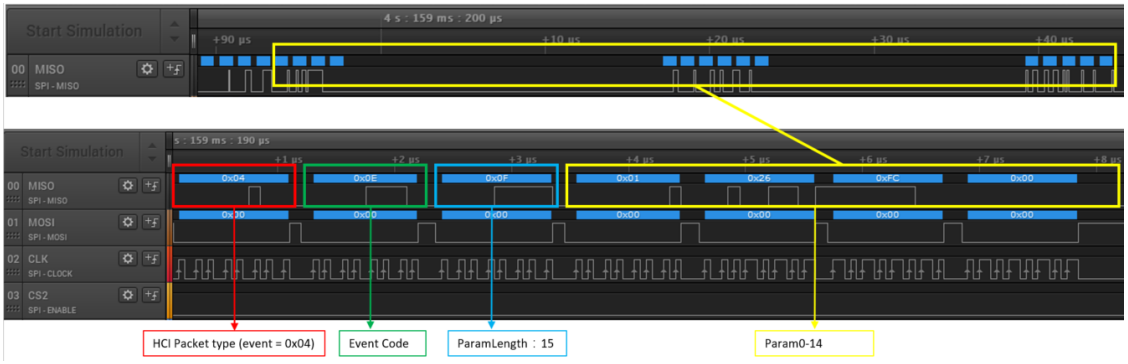

图 25. 逻辑分析仪的 HCI 事件格式

3. HCI 指令传输

图 26 展示了如何将 HCI 指令发送到 NxH3670。

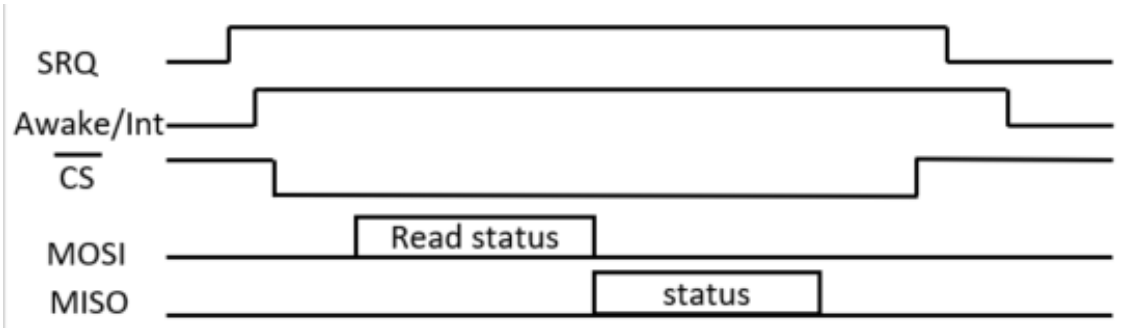

图 26. HCI 指令传输–读取状态

在开始 SPI 传输之前，必须确保 NxH3670 处于唤醒状态并且 SPI 总线可用。通过声明 SRQ 线路并等待 awake / int 信号的确认，可以完成该步骤。

虽然主机知道 NxH3670 处于唤醒状态，但仍必须检查 NxH3670 是否准备好接收新的 SPI 数据。可以使用 SPI 读取状态指令检索此信息。

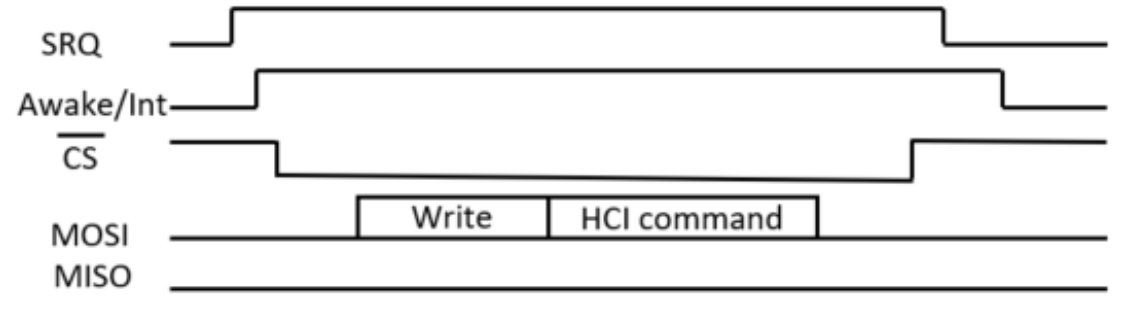

图 27. HCI 指令传输–写入指令

用户可以看到信号的变化，包括 CS、MOSI 和 MISO，如 图 28 所示。

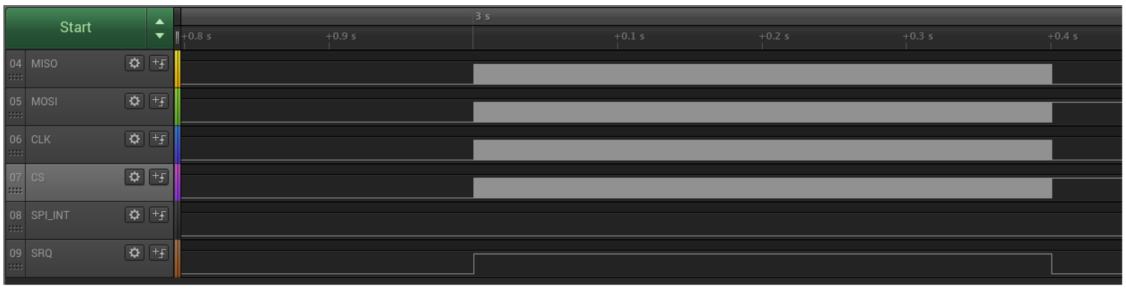

图 28. NXH3670 的启动过程

4. HCI 事件传输

   NxH3670 使用软件队列来存储多个 HCI 事件。如果 SPI 读取缓存区为空，则将最早的事件移至 SPI 读取缓存区，并声明 SPI 等待数据信号。

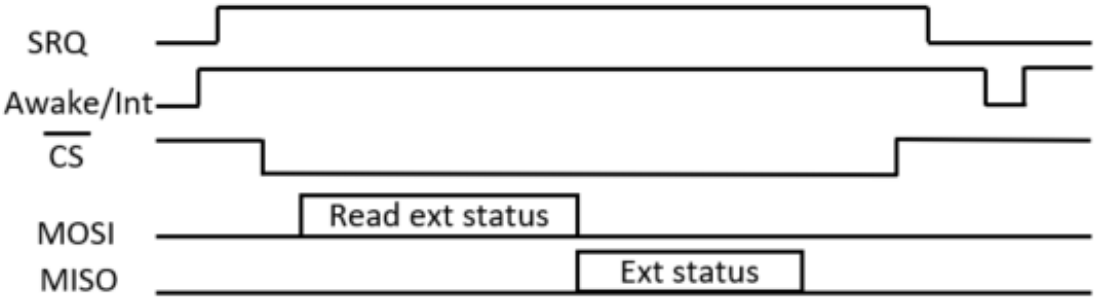

图 29. HCI 事件传输–读取扩展状态

NxH3670 通过声明 awake/int 信号来指示待处理数据。主机可以检索扩展状态以检查有多少数据待处理。 SRQ 信号被置为无效，awake/int 被置为无效。由于尚未读取实际的待处理数据，因此需要再次声明 awake/int 信号。

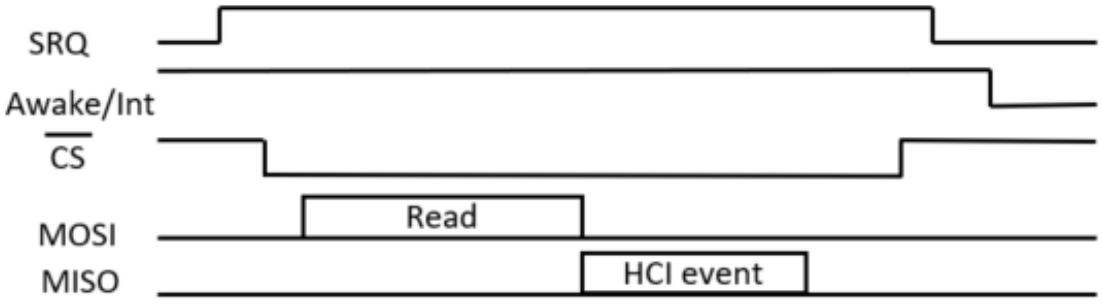

图 30. HCI 事件传输–读取

当主机知道有多少字节待处理时，可以启动 SPI 读取指令。NxH3670 将序列化的 HCI 事件发送回主机。

NxH3670 一次仅传送一个 HCI 事件。如果在软件队列中还有其他待处理的 HCI 事件被挂起，则它将最早出现的 HCI 事件再次移至 SPI 缓存区，并且重新开始上述序列。

如果主机读取速度不够快，由于缓存溢出，HCI 事件可能会丢失。

3.3 握手

SPI 握手协议使用两个物理硬件信号实现三个逻辑信号。

3.3.1 逻辑信号

1. 服务请求信号

   主机使用此信号来请求 NxH3670 的服务。当 NxH3670 检测到该信号时，可以通过唤醒信号指示它已准备好来处理服务请求。

2. 唤醒信号

   NxH3670 使用此信号表示它已唤醒。NxH3670 仅在 SRQ 信号有效时才有效，而不是在每次唤醒时都有效。

3. 待处理数据信号

   当 NxH3670 具有待处理数据时，它将向主机发出该信号。

3.3.2 物理信号

这三个逻辑信号映射到两个物理信号上，以减少所需的引脚数。

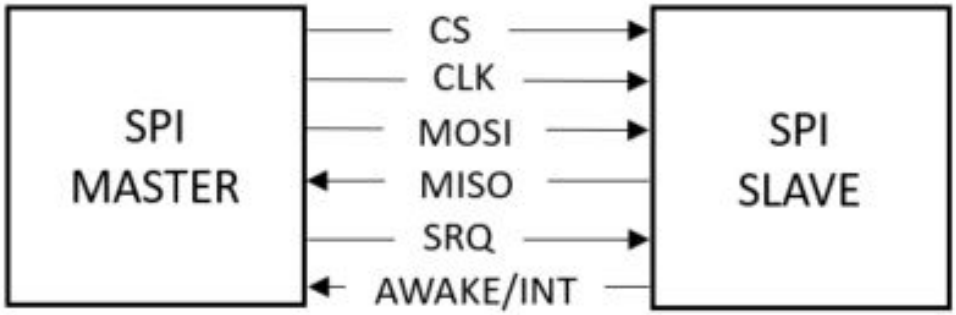

图 31. SPI 的握手物理信号

表 7. 物理信号到逻辑信号的映射

逻辑信号	SRQ 物理信号	AWAKE/INT 物理信号
服务请求信号	声明	不需关注
唤醒信号	声明	声明
待处理数据信号	未声明	声明

注意
待处理数据信号映射到 INT 物理信号。

可能会发生以下情况：

1. 主机启动 SPI 传输。
2. NxH3670 请求 SPI 传输。
3. 主机启动，并且 NxH3670 同时请求 SPI 传输。

为了使用户容易地了解握手过程，本文介绍了关于握手过程的逻辑分析仪的信号。

当 NxH3670 具有待处理数据时，它将生成以下序列以报告待处理数据。

图 32. NxH3670 请求 SPI 传输

只要有待处理的数据，NxH3670 就会保持唤醒状态。主机必须尽快读取未处理数据以节省功耗。

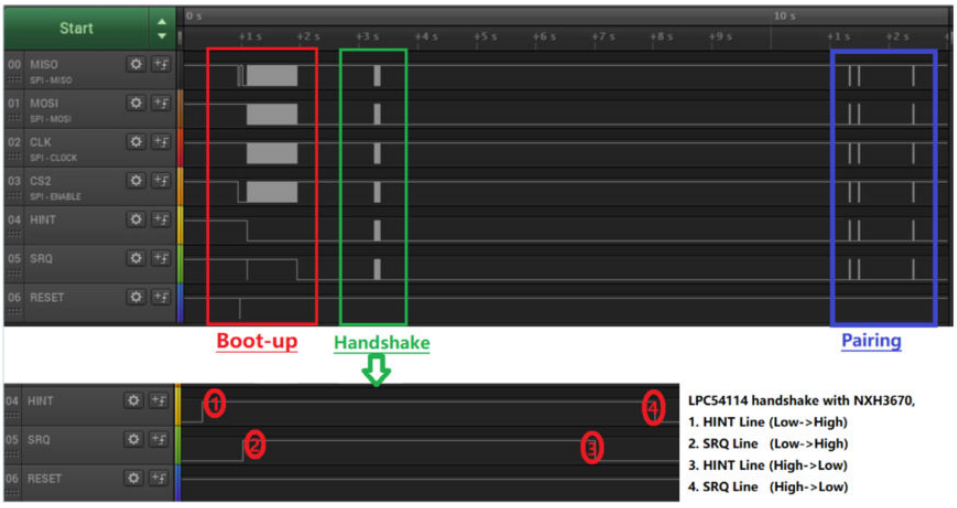

图 33. NxH3670 请求 SPI 传输

3.4 开始

用户可以使用 USB 线将 J13（FRDM-K32L2B）与 PC 连接，用于供电或下载固件。

4 总结

本文档介绍了低功耗蓝牙音频系统中 K32L2B_Dongle 的硬件设计和软件架构。可以作为用户构建自己方案的参考。

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: AN12647

arm