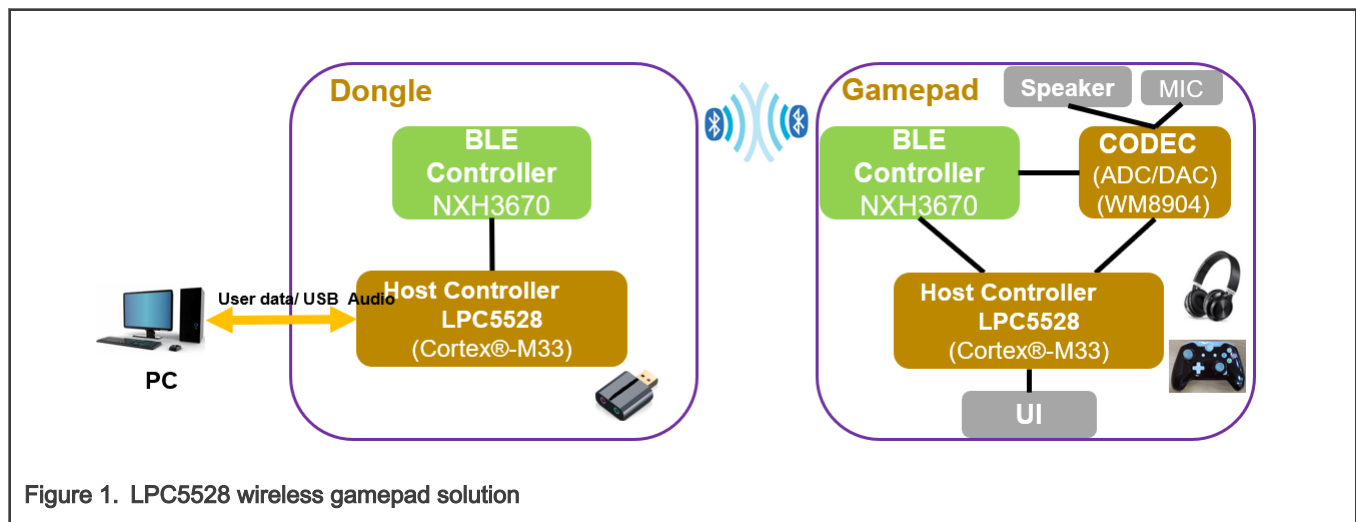


1 Introduction

This document introduces the wireless gamepad solution based on LPC5528 and NxH3670. It provides some necessary information to guide developers how to use the software and hardware resources provided by NXP to develop wireless gamepad or wireless headset. The design materials of this solution will be released on NXP website together with this document. Figure 1 shows the system block diagram of this solution.

Contents

- 1 Introduction.....1**
- 2 Getting started.....1**
 - 2.1 Hardware.....1
 - 2.2 Software.....4
- 3 Firmware development.....6**
 - 3.1 Flash layout.....6
 - 3.2 GUI tools.....10
 - 3.3 Project debug.....14
- 4 Gamepad usage.....17**
 - 4.1 Wireless mode.....18
 - 4.2 Wired mode.....21
 - 4.3 LED state.....21
 - 4.4 Volume control.....21
 - 4.5 Mode switch.....21
 - 4.6 Software reset.....21
 - 4.7 Low-power mode.....21
- 5 Useful document.....21**
- 6 Reference.....22**
- 7 Revision history.....22**



2 Getting started

This section introduces the hardware and software resources required by this solution.

2.1 Hardware

This solution requires two boards: LPC5528 gamepad main board and LPC5528 dongle board.



2.1.1 LPC5528 gamepad main board and dongle board features

The gamepad main board supports the following features:

- LPC5528JEV98 MCU (Arm[®] Cortex[®]-M33 core, 150 MHz clock, 512 KB Flash, 256 KB RAM, 7 × 7 mm² VFBGA98 package)
- NxH3670UK (Arm Cortex-M0 core, Ultra-low Power, Low Latency Audio for Wireless Gaming Headphone)
- WM8904 Codec
- Battery charge
- 3.5 mm headphone jack
- High Speed USB interface
- SWD interface
- Reset button, two user buttons
- Joysticks and buttons

The dongle board supports the following features:

- LPC5528JBD64 MCU (Arm Cortex-M33 core, 150 MHz clock, 512 KB Flash, 256 KB RAM, HTQFP package)
- NxH3670UK (Arm Cortex-M0 core, Ultra-low Power, Low Latency Audio for Wireless Gaming Headphone)
- High Speed USB interface
- SWD interface
- Reset button, ISP button

LPC5528 gamepad main board is as shown in [Figure 2](#) and [Figure 3](#).

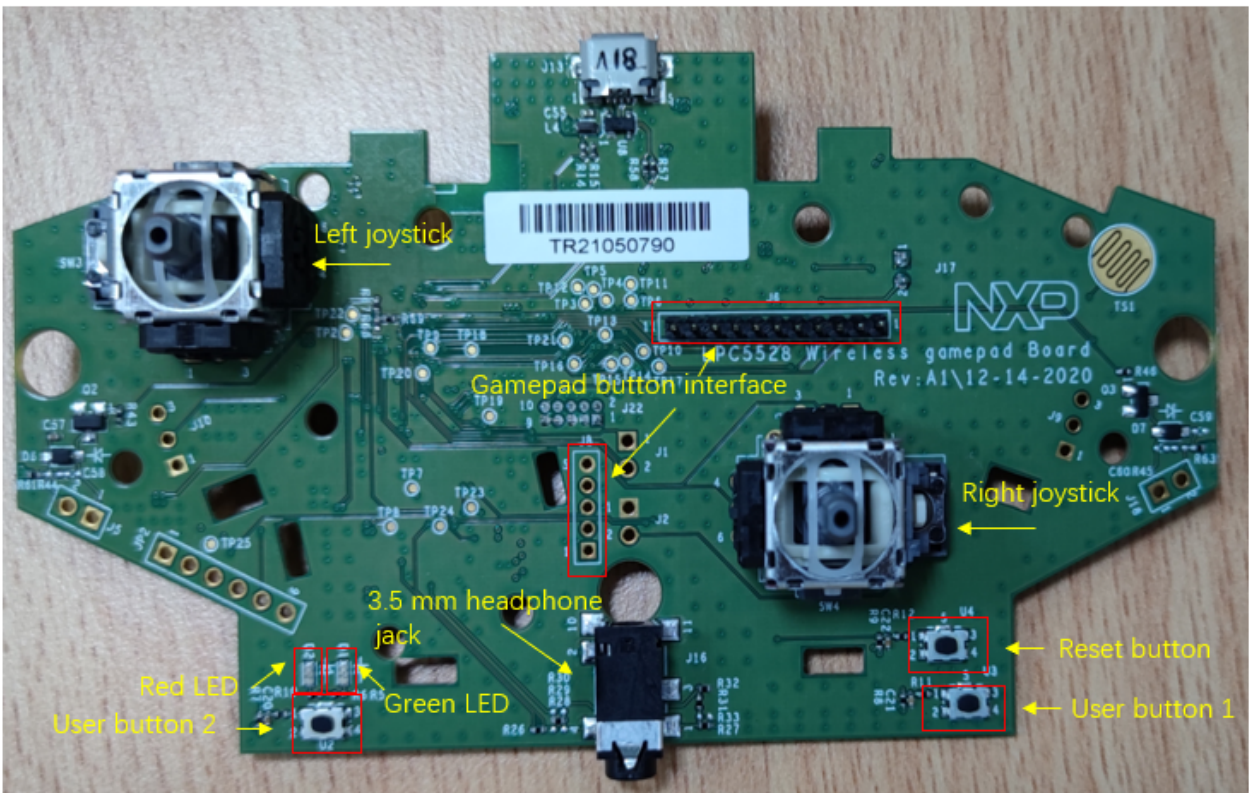


Figure 2. Front view of gamepad main board

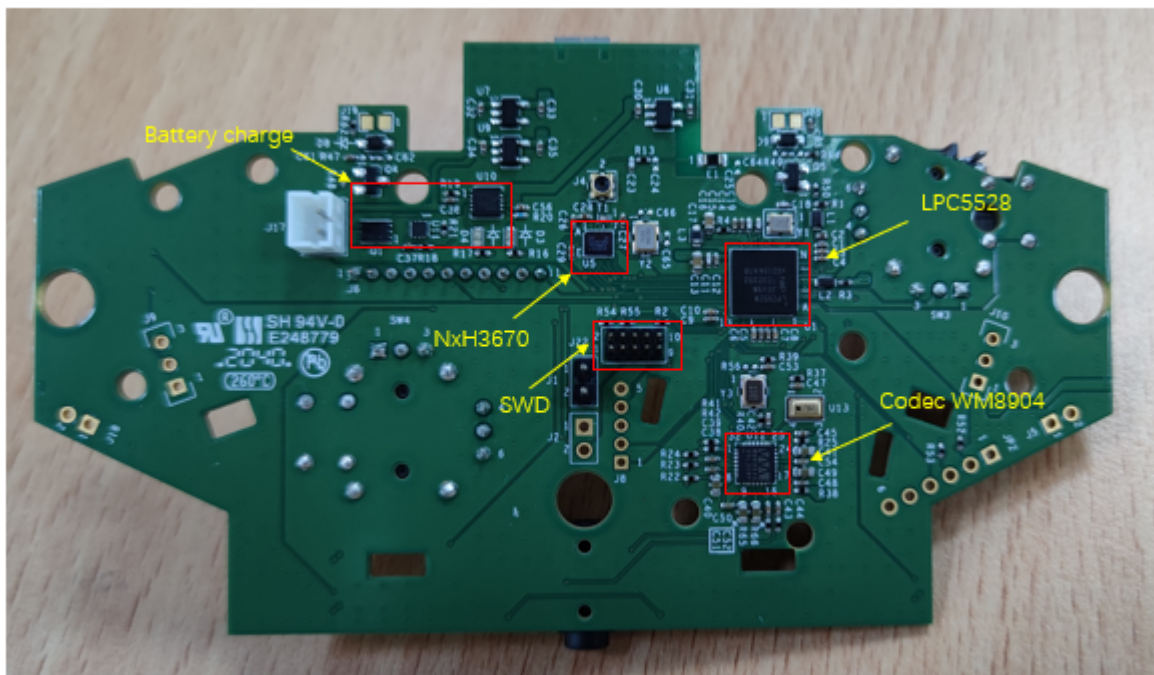


Figure 3. Back view of gamepad main board

LPC5528 dongle board is as shown in [Figure 4](#).

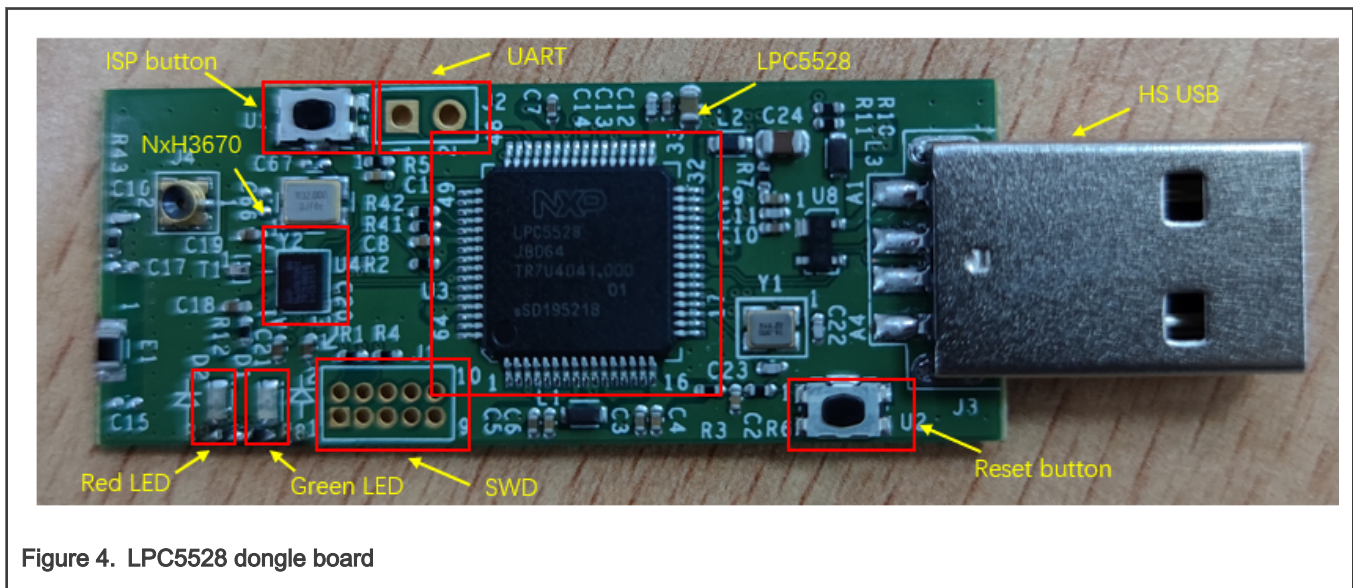


Figure 4. LPC5528 dongle board

2.2 Software

This solution is developed based on NxH3670 SDK Rev5.2 and LPC5528 SDK v2.7.1. NXP provides the NxH3670 SDK board and the corresponding software development kit. The NxH3670 SDK can be downloaded from [NxH3670 SDK Gaming Package](#). The original host controller on the NxH3670 SDK board is KL27. But in this solution, we use high performance LPC5528 instead of KL27. This LPC5528-based solution allows enough room for users to expand their own functions.

2.2.1 Original NxH3670 SDK

The structure of original NxH3670 SDK Rev5.2 is as shown in [Figure 5](#).

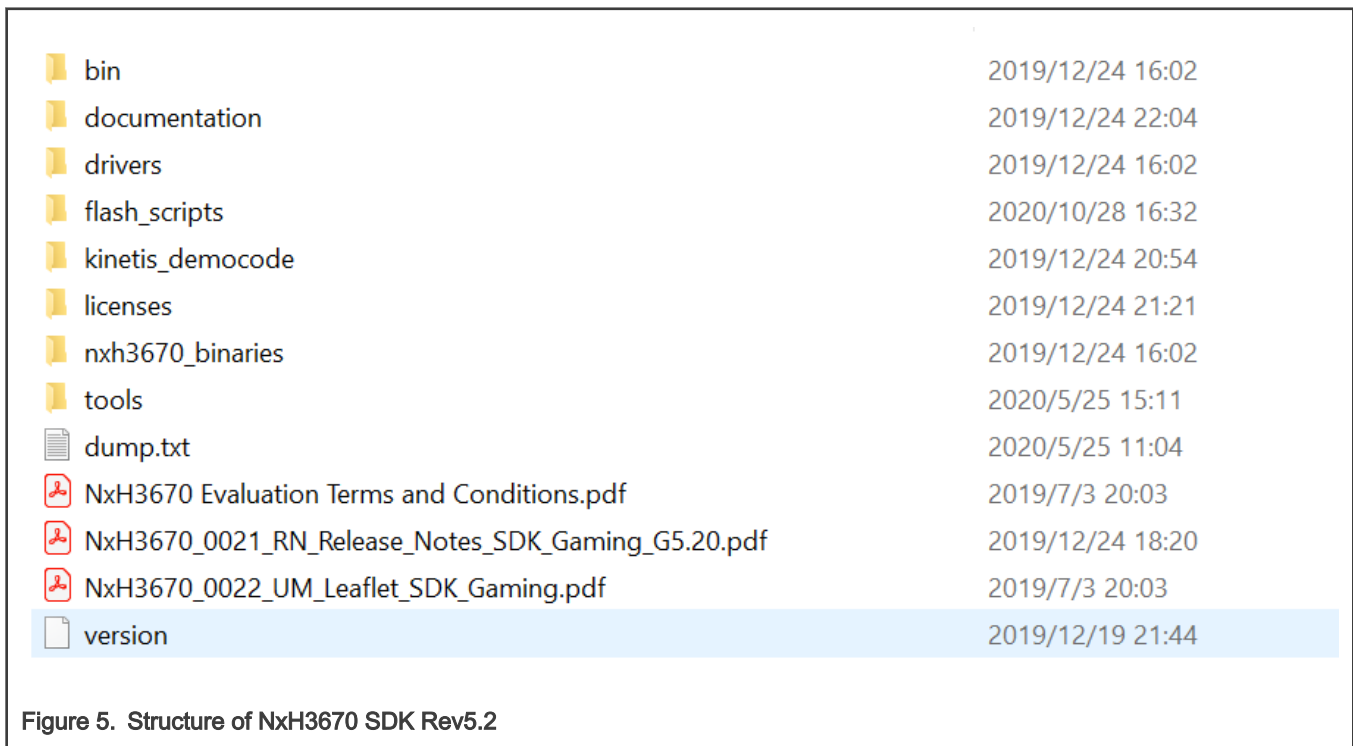


Figure 5. Structure of NxH3670 SDK Rev5.2

The content in the SDK package is as follows:

- **bin:** Some executable files necessary for the execution of GUI tools, as shown in [Figure 6](#).

flashtool.exe	2019/12/19 22:10
persist_datagen.exe	2019/12/19 22:10
testsuite_cli.exe	2019/12/19 22:10
to_eep.exe	2019/12/19 22:10
vcom_cli.exe	2019/12/19 22:10

Figure 6. Files in the bin directory of NxH3670 SDK

- **flash_scripts:** Flash scripts are used to generate dongle and headset device information and pairing data.
- **document:** Some useful document for developers.

















 NxH3670_0027_UM_HAPI_SysRadioTest.pdf
 NxH3670_0029_AN_Performance.pdf
 NxH3670_0033_AN_Pre_Production_Steps.pdf
 NxH3670_AN11953_HAPI_Bootloader.pdf
 NxH3670_AN12244_RadioIntegration.pdf
 NxH3670_AN12280_Reference_Application_Software_Architecture.pdf
 NxH3670_AN12360_Gaming.pdf
 NxH3670_AN12361_OTA_Firmware_Upgrade.pdf
 NxH3670_AN12362_TestSuite.pdf
 NXH3670_HAPI_overview.pdf
 NxH3670_UM11148_HAPI_Gaming.pdf
 NxH3670_UM11149_Host_Interface.pdf
 NxH3670_UM11198_FlashTool.pdf
 NxH3670_UM11203_HAPI_OTA.pdf
 NxH3670-0032_UM_HAPI_VCOM_Rev0.6.pdf
 NxH3670UK_Datasheet.pdf

Figure 7. Documentations provided in the NxH3670 SDK package

- **kinetis_democode:** Including the app, ota_app, and SSB projects required by the headset and dongle.
- **nxh3670_binaries:** The firmware of NxH3670 required by dongle and headset.

- **tools:** Some useful tools include Nhx3670 Flash Tool and VCOM Tool. For the details of tool usage, see *NxH3670 FlashTool* (document UM11198) and *NxH3670 HAPI VCOM*.

2.2.2 LPC5528_NxH3670 SDK

Since this solution is developed based on the Nhx3670 SDK and the KL27 host controller is replaced with LPC5528, it is necessary to replace the KL27 SDK with the LPC5528 SDK. In order to improve the development efficiency, this solution reuses the file structure in [Figure 5](#) and replaces `kinetis_democode` with `lpc_democode`. The development tool is Keil v5.28 IDE. The content of the modified Nhx3670 SDK for LPC5528, `LPC5528_NxH3670 SDK`, is as shown in [Figure 8](#).

bin	2020/11/13 14:47	File folder
documentation	2020/11/16 9:31	File folder
drivers	2020/11/16 9:33	File folder
flash_scripts	2020/11/13 14:47	File folder
licenses	2020/11/16 9:31	File folder
lpc_democode	2020/11/13 14:47	File folder
nxh3670_binaries	2020/11/13 14:48	File folder
tools	2020/11/13 19:26	File folder

Figure 8. LPC5528_NxH3670 SDK Rev 1.0

Compared with original Nhx3670 SDK, the main differences are as follows:

- Use `lpc_democode` to replace `kinetis_democode`. `lpc_democode` contains the Secondary Stage Bootloader (SSB), applications (`app`), and ota applications (`ota_app`) of the dongle and gamepad.

lpc5528_dongle	2020/11/13 15:49	File folder
lpc5528_dongle_ota	2020/11/13 15:48	File folder
lpc5528_dongle_ssb	2020/11/13 14:54	File folder
lpc5528_gamepad	2020/11/13 15:47	File folder
lpc5528_gamepad_ota	2020/11/13 15:47	File folder
lpc5528_gamepad_ssb	2020/11/13 14:48	File folder

Figure 9. LPC5528_NxH3670 demo code

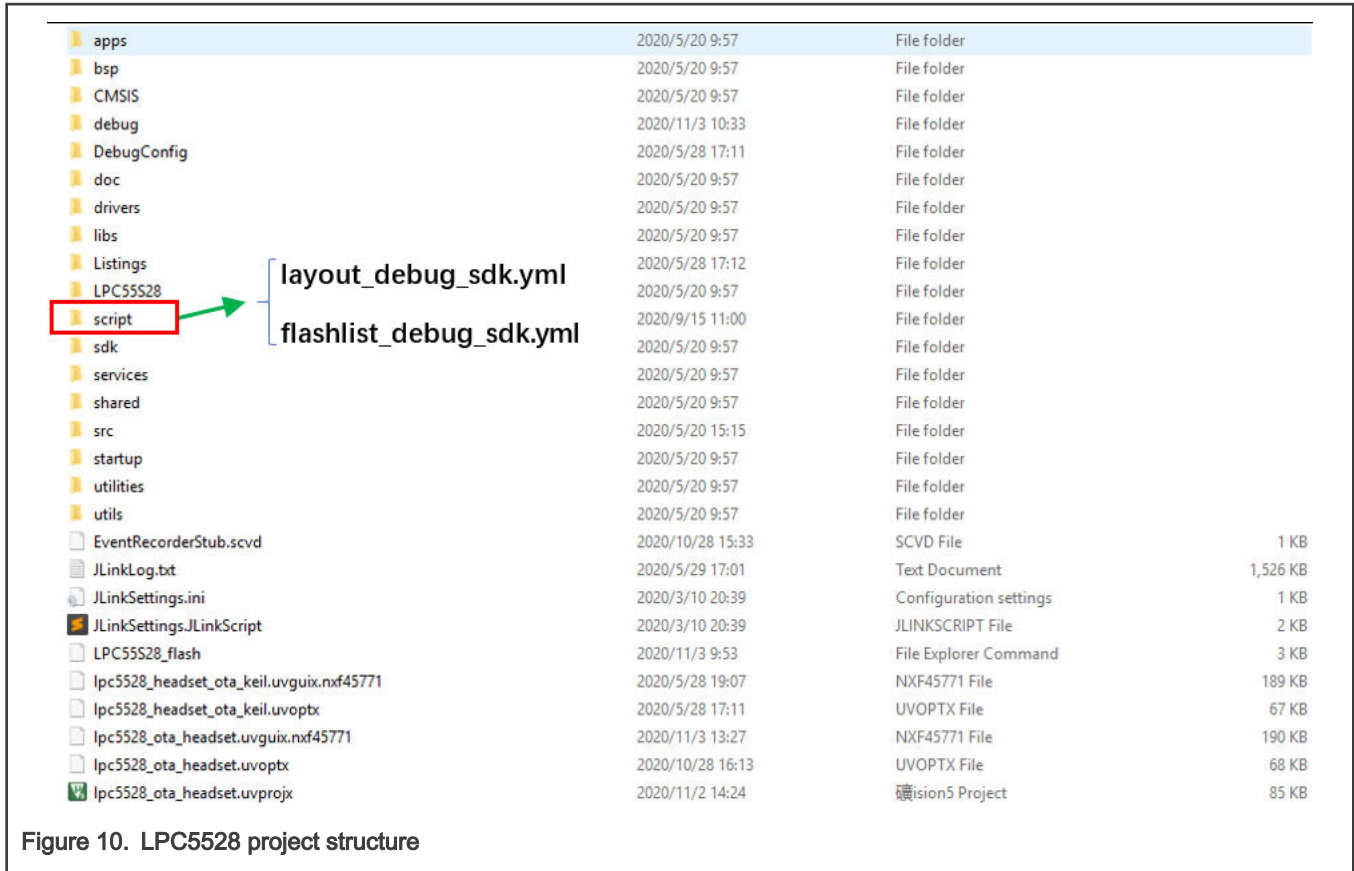
- **tools:** Add a LPC5528_NxH3670 Flash Tool. See [LPC5528_NxH3670 Flash tool](#) for the details of tool usage.

3 Firmware development

This section introduces how to use LPC5528_NxH3670 SDK to develop the wireless gamepad. Before starting to debug the project, developers need to understand the Flash layout and GUI tools usage.

3.1 Flash layout

This section introduces two very important files, `layout_debug_sdk.yml` (layout file) and `flashlist_debug_sdk.yml` (flashlist file). These two files are stored in the script folder of each project directory, as shown in [Figure 10](#).



Layout file and flashlist file are very important files. When using LPC5528_NxH3670 Flash tool and NxH3670 Flash tool for firmware download and OTA upgrade, the correct .yml file must be provided.

3.1.1 layout_debug_sdk.yml

Before analyzing the layout file, developers need to understand the concept of partition table. The partition table is actually a binary file, which stores the name, attribute, address, size and other information of each firmware. Before booting app or ota_app, SSB also need to obtain the address of the firmware to be booted from the partition table. The content of the partition table is determined by the layout file. The format of the layout file is as shown in Figure 11.

```

1 # number of partitions: 6
2 # max used size (including ssb and table): 512 KB
3 active_partition: 0
4 layout_version: 0x52
5 app_type: 0 # 0--gamepad, 1-- dongle
6 partition_table_addr: 0x7000
7 entries:
8 # partition_id 0
9 - name: "app"
10 type: firmware
11 base_address: 0x81f0 #0xbf0
12 size: 0x41c00 # (263 KB) 0x41c00 + 0x1400(persdata) = 0x43000
13 offsets:
14 - 0x0 # k1_app | 143 KB (0x8000 - 0x28E00, 52 KB free)
15 - 0x23c10 # nxh_app | 70 KB (4 KB free)
16 - 0x35410 # rfmac | 18 KB (2 KB free)
17 - 0x39c10 # cf | 32 KB (2 KB free)
18 # partition_id 1
19 - name: "ota"
20 type: firmware
21 base_address: 0x501f0
22 size: 0x22d00 # 140k KB
23 offsets:
24 - 0x0 # lpc_ota_app | 80 KB
25 - 0x13e10 # nxh_ota_app | 60 KB
26 # partition_id 2
27 - name: "app_data"
28 type: appdata
29 base_address: 0x4a800
30 size: 0x400 # 1 KB
31 offsets:
32 - 0x0 # app_data | 1 KB
33 # partition_id 3
34 - name: "nxh_pers_data"
35 type: appdata
36 base_address: 0x4ac00
37 size: 0x400 # 1 KB
38 offsets:
39 - 0x0 # nxh_pers_data | 1 KB
40 # partition_id 4
41 - name: "app_data_factory"
42 type: factoryreset
43 base_address: 0x4b000
44 size: 0x400 # 1 KB
45 offsets:
46 - 0x0 # app_data_factory | 1 KB
47 #partition_id 5
48 - name: "nxh_pers_factory"
49 type: factoryreset
50 base_address: 0x4b400
51 size: 0x400 # 1 KB
52 offsets:
53 - 0x0 # nxh_persistent_data_factory | 1 KB
54

```

The real starting address of the app image is 0x8200, 0x81f0-0x81ff is a 16-byte image header

The real starting address of the ota_app image is 0x50200, 0x501f0-0x501ff is a 16-byte image header

Figure 11. Format of the layout file

The page size of LPC5528's flash is 512 bytes, but as shown in Figure 11, the starting addresses of app and ota_app are not aligned with 512 bytes. They are actually the start addresses that contain the 16-byte image header. The real firmware starting addresses are 0x8200 and 0x50200 respectively. Since LPC5528's Flash sector size is 32 KB, in order to manage the storage space of app and ota_app more conveniently, app and ota_app are stored in different sectors.

The address in the linker file of the Keil project needs to meet 512 bytes alignment and at least 16 bytes of space is reserved for the imager header, so 512 bytes are reserved for the imager header and the first 496 bytes are filled with 0. Therefore, 0x8000-0x81FF and 0x50000-0x501FF are composed of 496 bytes, 0x00, and 16 bytes image header.

The flash partition corresponding to this layout file is as shown in [Figure 12](#).

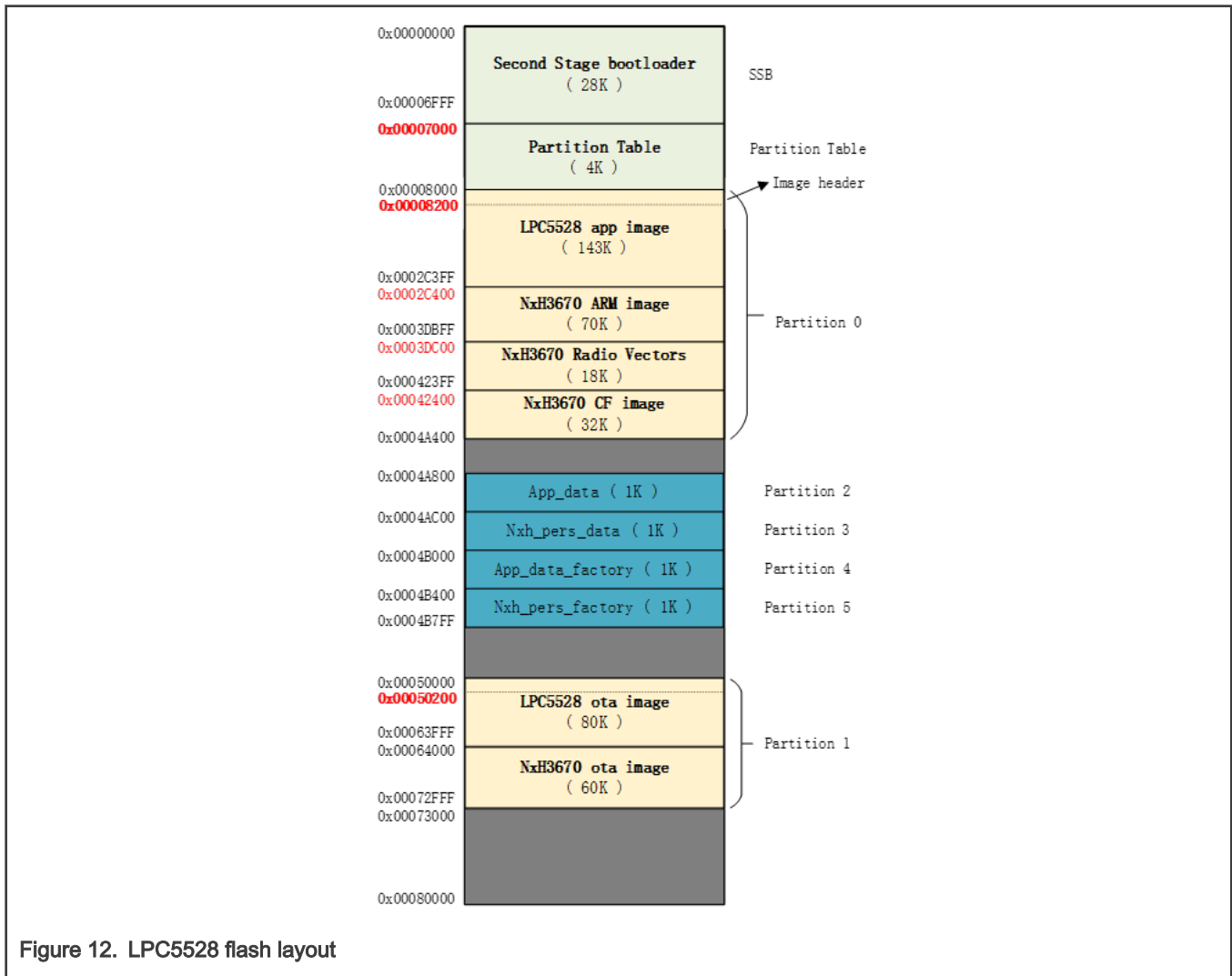


Figure 12. LPC5528 flash layout

3.1.2 Flashlist_debug_sdk.yml

The content of flashlist file is as shown in [Figure 13](#).

```
1  binaries:
2    - name: ssb
3      files:
4        - lpc_democode/apps/lpc5528_gamepad_ssb/debug/lpc5528_gamepad_ssb.bin
5        address: 0x0
6    - name: lpc_app
7      files:
8        - lpc_democode/apps/lpc5528_gamepad/debug/lpc5528_gamepad.bin.eep
9        offset_index: 0
10       partition: { name: "app", type: firmware }
11       headroom: -1
12    - name: nxh_app
13      files:
14        - nxh3670_binaries/arm/phGamingRx.ihex.eep
15        offset_index: 1
16        partition: { name: "app", type: firmware }
17        headroom: 4096
18    - name: rfmac
19      files:
20        - nxh3670_binaries/rfmac/rfmac.eep
21        offset_index: 2
22        partition: { name: "app", type: firmware }
23        headroom: 2048
24    - name: cf
25      files:
26        - nxh3670_binaries/cf/phStereoInterleavedAsrcRx.eep
27        offset_index: 3
28        partition: { name: "app", type: firmware }
29        headroom: 2048
30    - name: lpc_ota_app
31      files:
32        - lpc_democode/apps/lpc5528_gamepad_ota/debug/lpc5528_gamepad_ota.bin.eep
33        offset_index: 0
34        partition: { name: "ota", type: firmware }
35    - name: nxh_ota_app
36      files:
37        - nxh3670_binaries/arm/phOtaHeadset.ihex.eep
38        offset_index: 1
39        partition: { name: "ota", type: firmware }
```

Figure 13. flashlist_debug_sdk.yml

The flashlist file provides the names and relative paths of some necessary firmware. In addition to downloading the firmware in [Figure 13](#), you also need to download the partition table and device information (NxH_pers_data) to the specified address of LPC5528.

3.2 GUI tools

In order to facilitate developers to debug and update programs, the LPC5528_NxH3670 SDK also provides three tools, namely NxH3670 Flash tool, LPC5528_NxH3670 Flash tool and VCOM tool. Among them, NxH3670 Flash tool and VCOM tool come from original NxH3670 SDK Rev5.2.

3.2.1 NxH3670 Flash tool

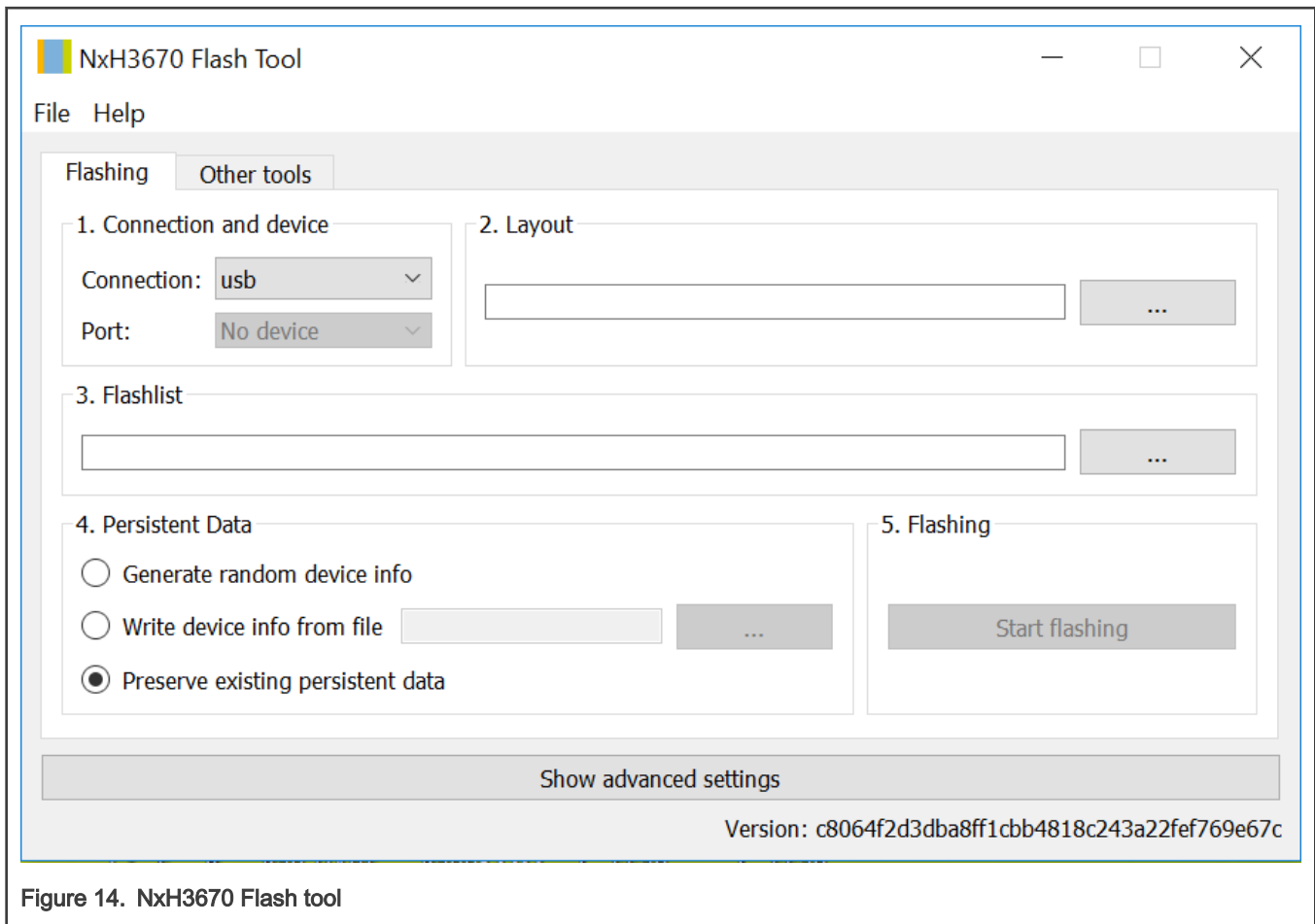


Figure 14. NxH3670 Flash tool

There are two main functions supported by this tool:

- Host controller program download
- OTA upgrade

When using this tool to download the host controller program, the tool currently does not support the program download of LPC5528, so an LPC5528 NxH3670 Flash tool GUI is re-developed in this solution.

3.2.2 LPC5528_NxH3670 Flash tool

The GUI interface of LPC5528_NxH3670 Flash tool is as shown in [Figure 15](#).

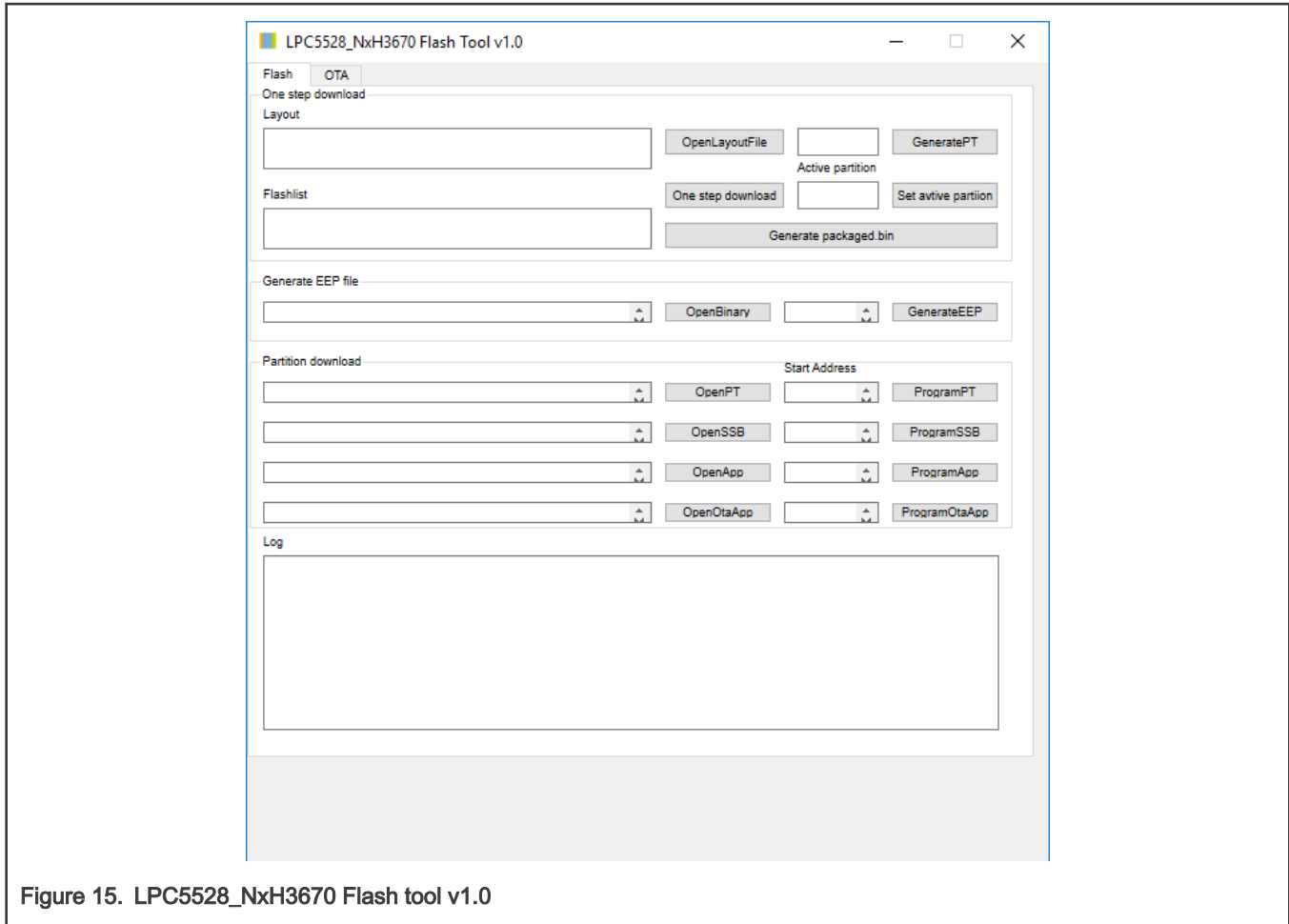


Figure 15. LPC5528_NxH3670 Flash tool v1.0

This tool implements the following nine functions. Except **Generate partition table**, **Generate EEP**, and **Generate packaged.bin**, when using other functions, LPC5528 need to enter the ISP mode.

3.2.2.1 Generate partition table

This function is used to generate partition table.

1. Click the **OpenLayoutFile** button to select the correct layout file.
2. Click the **GeneratePT** button to generate a partition table file and it will be saved in same directory.

3.2.2.2 Set active partition

This function is used to set active partition.

1. Select the correct layout file and enter a valid partition value in the **Active partition** box.
2. Click the **Set active partition** button. The GUI tool will first read out the partition table that already exists in the flash and modify the active partition value, and then write the new partition table to the flash.

3.2.2.3 Generate EEP

This function is used to generate an EEP file with an image header.

1. Click the **OpenBinary** button and select the *bin* file.
2. Click the **GenerateEEP** button to generate the EEP file and save it in the same directory. In addition to using this function to generate EEP files, you can also use the after-build command in the Keil IDE.

For the details, see [Generate binary and EPP file](#).

3.2.2.4 Program partition table

This function is used to download the partition table.

1. Click the **OpenPT** button.
2. Select the partition table file and enter the download address in the **Start Address** window.
3. Click the **ProgramPT** button to download the partition table.

3.2.2.5 Program SSB

This function is used to download SSB firmware.

1. Click the **OpenSSB** button.
2. Select the SSB file and enter the download address in the **Start Address** window.
3. Click the **ProgramSSB** button to download the SSB.

3.2.2.6 Program App

This function is used to download App firmware.

1. Click the **OpenApp** button.
2. Select the *app.eep* file and enter the download address in the **Start Address** window.
3. Click the **ProgramApp** button to download the app firmware.

3.2.2.7 Program Ota App

This function is used to download the *ota_app* firmware.

1. Click the **OpenOtaApp** button.
2. Select the *ota_app* file and enter the download address in the **Start Address** window.
3. Click the **ProgramOtaApp** button to download the *ota_app* firmware.

3.2.2.8 Generate packaged binary

This function is used to generate packaged binary.

1. Select a layout file, and the GUI will automatically select the flashlist file in this directory.
2. Click the **Generate packaged.bin** button. The GUI parses the selected layout file and generates a partition table binary, and then merges the SSB, *app.eep*, *ota_app.eep*, partition table, and device information files into one *_packaged.bin* file according to the firmware path provided in the *flashlist* file.

3.2.2.9 One step download

This function is used for one-step download.

1. Click the **Generate packaged.bin** button to generate a *_packaged.bin* file containing multiple firmware.
2. Click the **One step download** button to download multiple firmware to the address specified by the layout file.

If a *packaged.bin* has been generated and does not need to be updated, you can directly click the **One step download** button to download the complete firmware to the flash.

3.2.3 VCOM tool

This solution also provides a NxH3670 VCOM tool, and its GUI interface is as shown in [Figure 16](#).

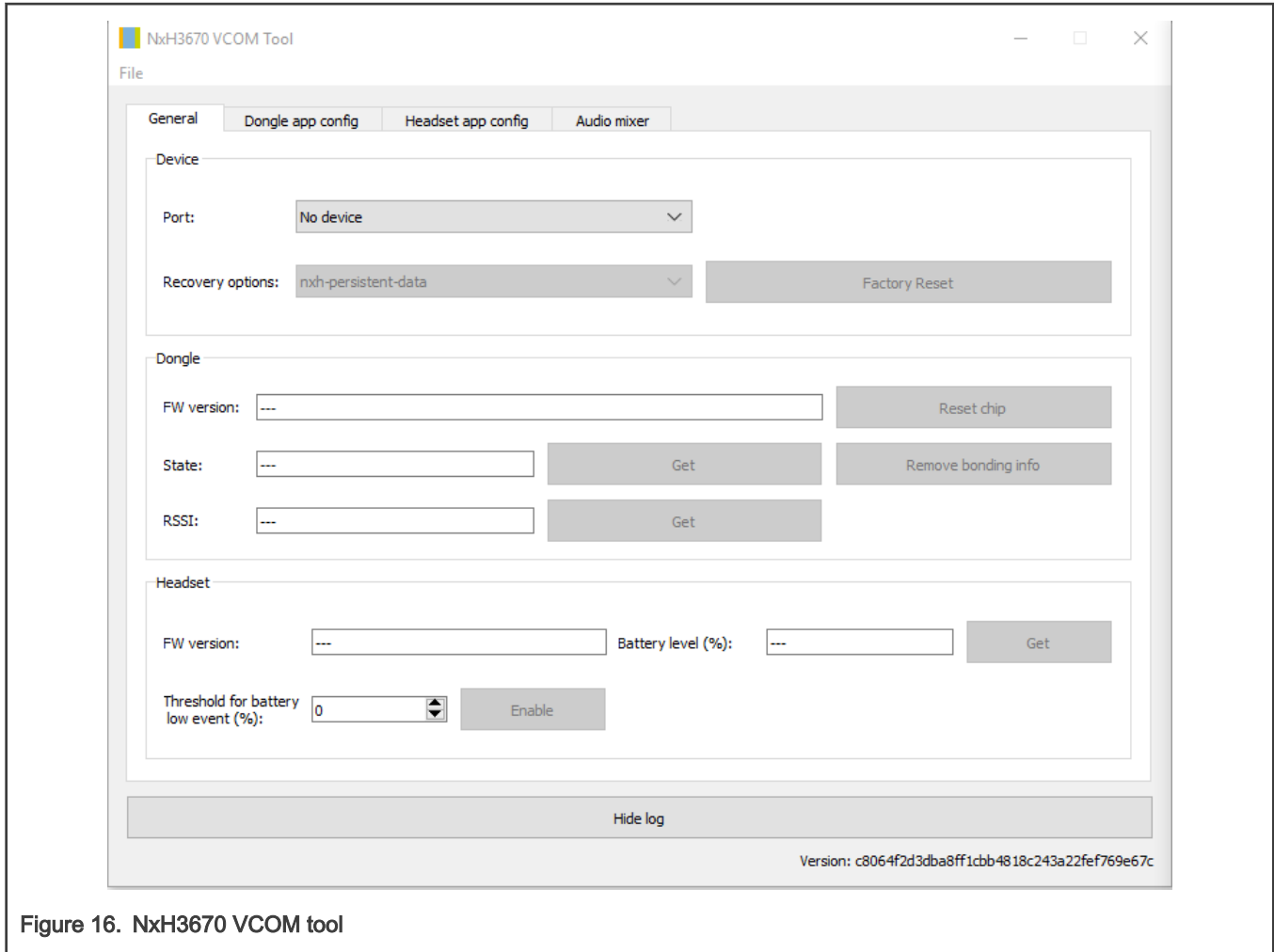


Figure 16. NxH3670 VCOM tool

For the usage of NxH3670 VCOM tool, see *NxH3670-0032_UM_HAPI_VCOM_Rev0.6.pdf* in NXH3670 SDK Rev5.2.

3.3 Project debug

This section describes how to debug each project in the *lpc_democode* directory.

3.3.1 Configuration of Keil projects

When using Keil for SSB, app and *ota_app* development, some special project configurations are required.

3.3.1.1 Generate binary and EPP file

When developing app and *ota_app*, you need to configure after build command to generate *eep* files. The steps are as follows:

1. Open the *lpc5528_gamepad* project or *ota_app* project in the *lpc_democode/apps* directory.
2. Click the



button, open the project option.

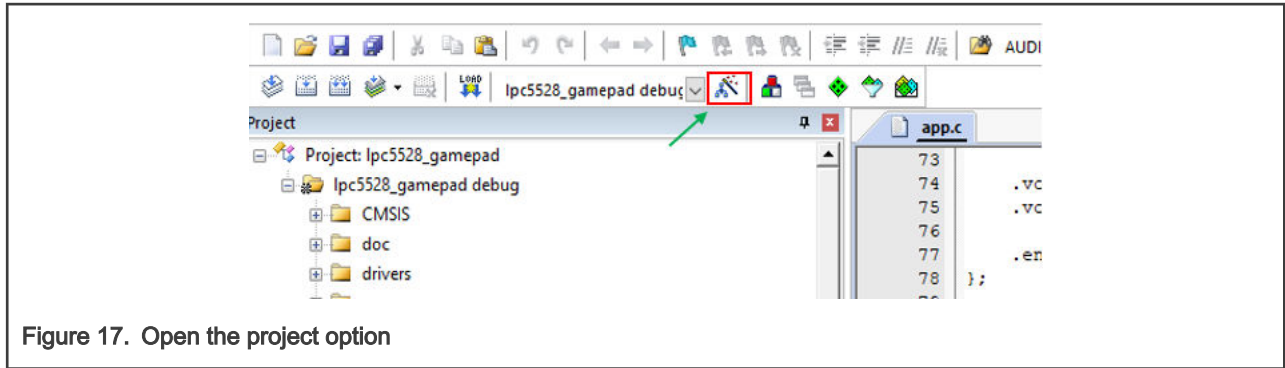


Figure 17. Open the project option

3. Switch to the user window and add two commands to the **After build/Rebuild** column, as shown in [Figure 18](#). These two commands will be executed automatically every time the project is compiled.

The first command is to generate a *.bin* file and the second command is to generate a *.eep* file.

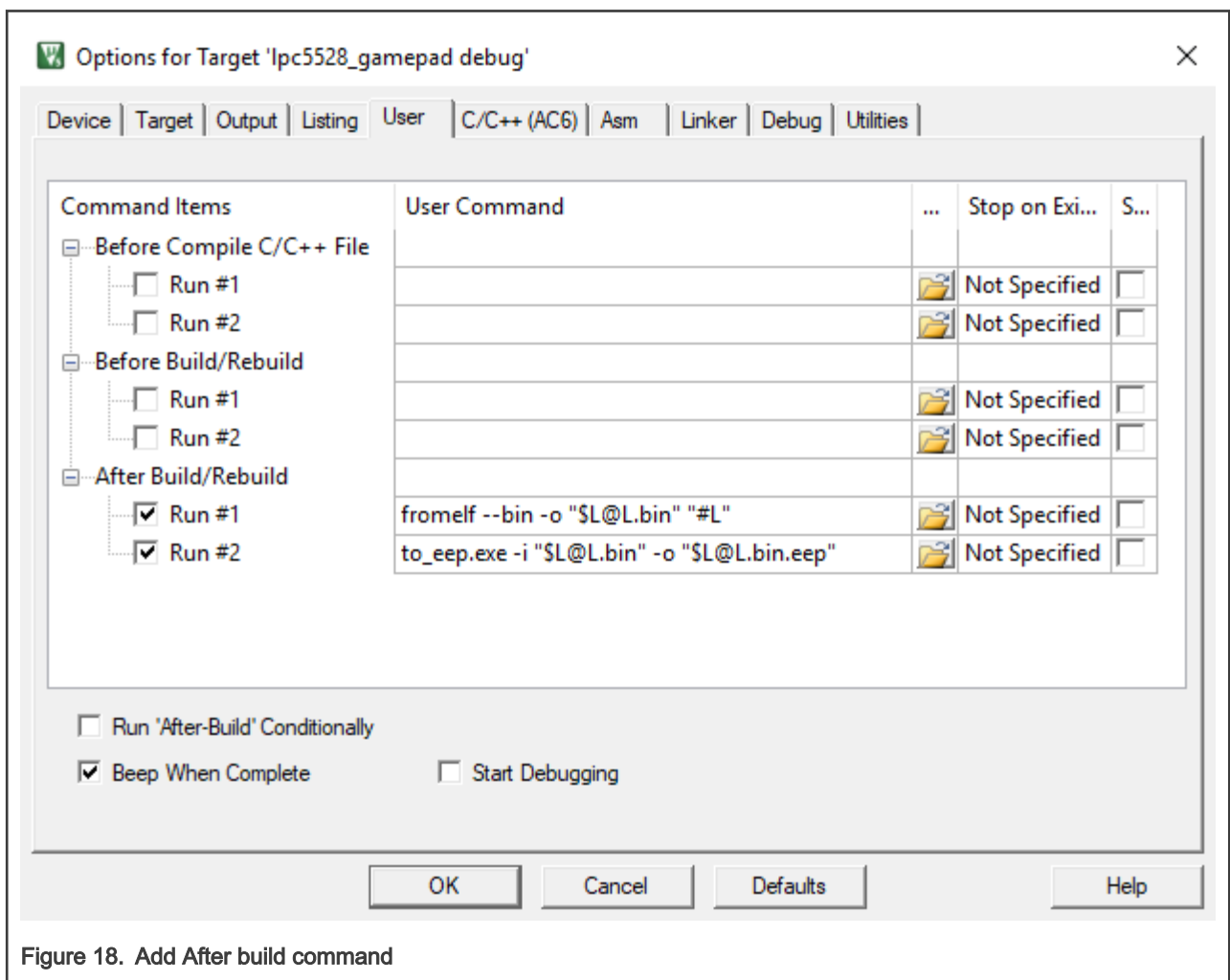


Figure 18. Add After build command

The second command needs to use the *to_eep.exe* file, which is stored in the */bin/windows* directory of the LPC5528_NxH3670 SDK package. Before executing this command, you need to copy *to_eep.exe* to the Keil installation directory, *.../ARM/ARMCLANG/bin*. The phenomenon after execution is as shown in [Figure 19](#).

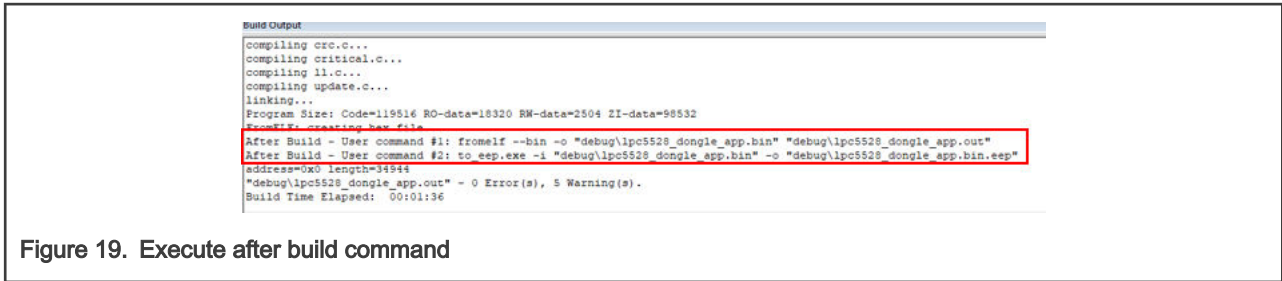


Figure 19. Execute after build command

The *eep* file actually adds a 16-byte image header to the head of the binary file, as shown in Figure 20.

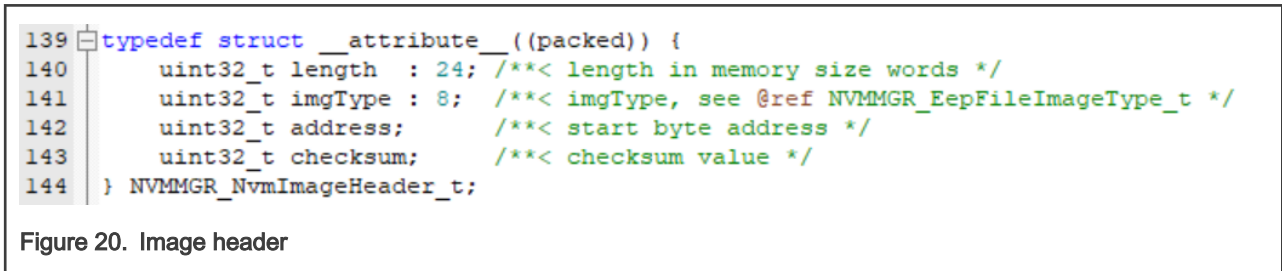


Figure 20. Image header

What is downloaded to LPC5528 through LPC5528_NxH3670 Flash tool is the *.eep* file. When using Keil to debug the project directly, the binary file without the image header is downloaded.

As shown in Figure 20, the image header has a total of 16 bytes, including Image length, type, starting address and checksum. When performing an OTA upgrade, the dongle will read the header of the image to be upgraded in the remote (Gamepad) flash and compare it with the new image header. If the image headers are the same, they are considered the same image, and the dongle will skip the upgrade of this image, this image will not be sent to the gamepad. If they are different, a new image will be sent to the remote to replace the old image.

3.3.1.2 Sector erase

The flash of the LPC5528 needs to contain multiple firmware, so when debugging a single project, you need to set the download function to **Sectors Erase**. The contents of other firmware are retained when a single program is downloaded. The steps are as follows:

1. Follow [Step 1](#) and [Step 2](#) in [Generate binary and EPP file](#).
2. Switch to the **Debug** window and click **Settings -> Flash Download**. Select **Erase Sectors** and click **OK** to save the configuration.

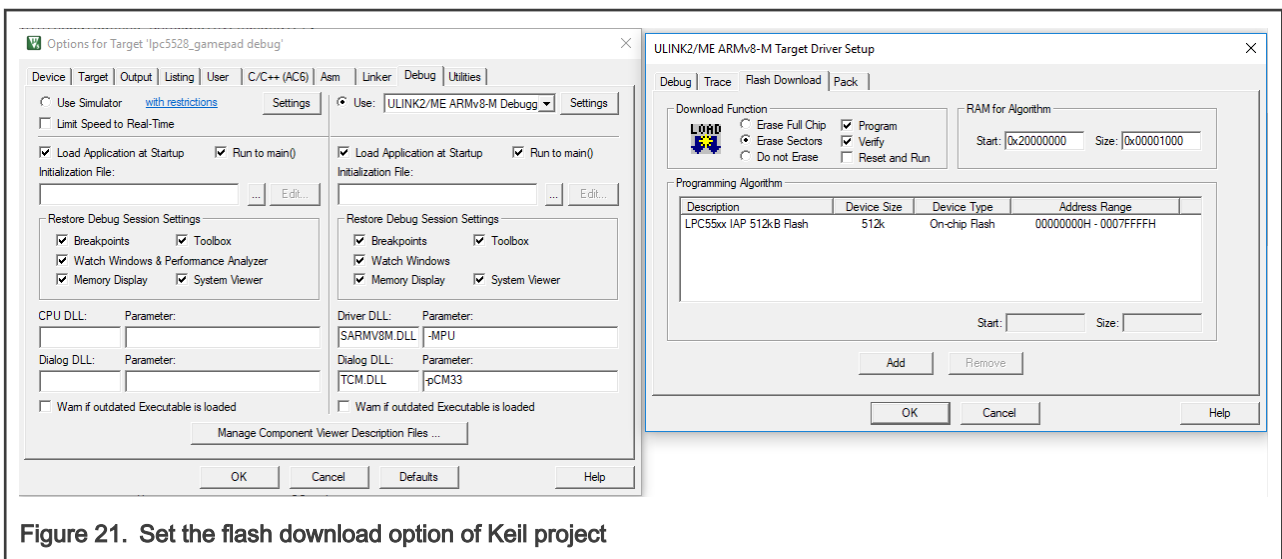


Figure 21. Set the flash download option of Keil project

3.3.2 Debugging single project

Before debugging the app or `ota_app` separately, make sure that the SSB and partition table have been downloaded in the LPC5528 flash. SSB is the first code executed after MCU reset, it will read the partition table to obtain the active partition value and then jump to the corresponding partition.

3.3.2.1 Download SSB

There are three ways to download SSB:

- Open the SSB project, compile the SSB project, and click the **Download** button in the project to directly download the SSB to `0x00` in the flash.
- Use the **Program SSB** function in the LPC5528_NxH3670 Flash tool to download the SSB firmware. For details, see [Program SSB](#).
- Use the **One step download** function to download SSB. For details, see [One step download](#).

3.3.2.2 Download Partition table

There are two ways to download the partition table:

- Use the **Program PT** function to download the Partition table. For details, see [Program partition table](#).
- Use the **One step download** function to download the partition table. For details, see [One step download](#).

After downloading the SSB and partition table, you can debug the app or `ota_app` project separately.

- When debugging the app project, set `active_partition` in the layout file to **0**.
- When debugging `ota_app`, set `active_partition` to **1**.

Once modifying the layout file, you need to regenerate the partition table and download the new partition table to the flash.

4 Gamepad usage

After downloading the firmware of the dongle and gamepad using the **One step download** function, reset the two boards and make both boards run in app mode. By default, the gamepad and dongle work in the wireless mode. They will automatically complete the pairing process. You can switch the working mode by pressing a specific button and adjust the volume. The names of the buttons are as shown in [Figure 22](#).



Figure 22. Button name of gamepad

4.1 Wireless mode

Plug the USB dongle into the PC, the red and green led will light up, and a composite USB device will be recognized on the PC, as shown in [Figure 23](#).

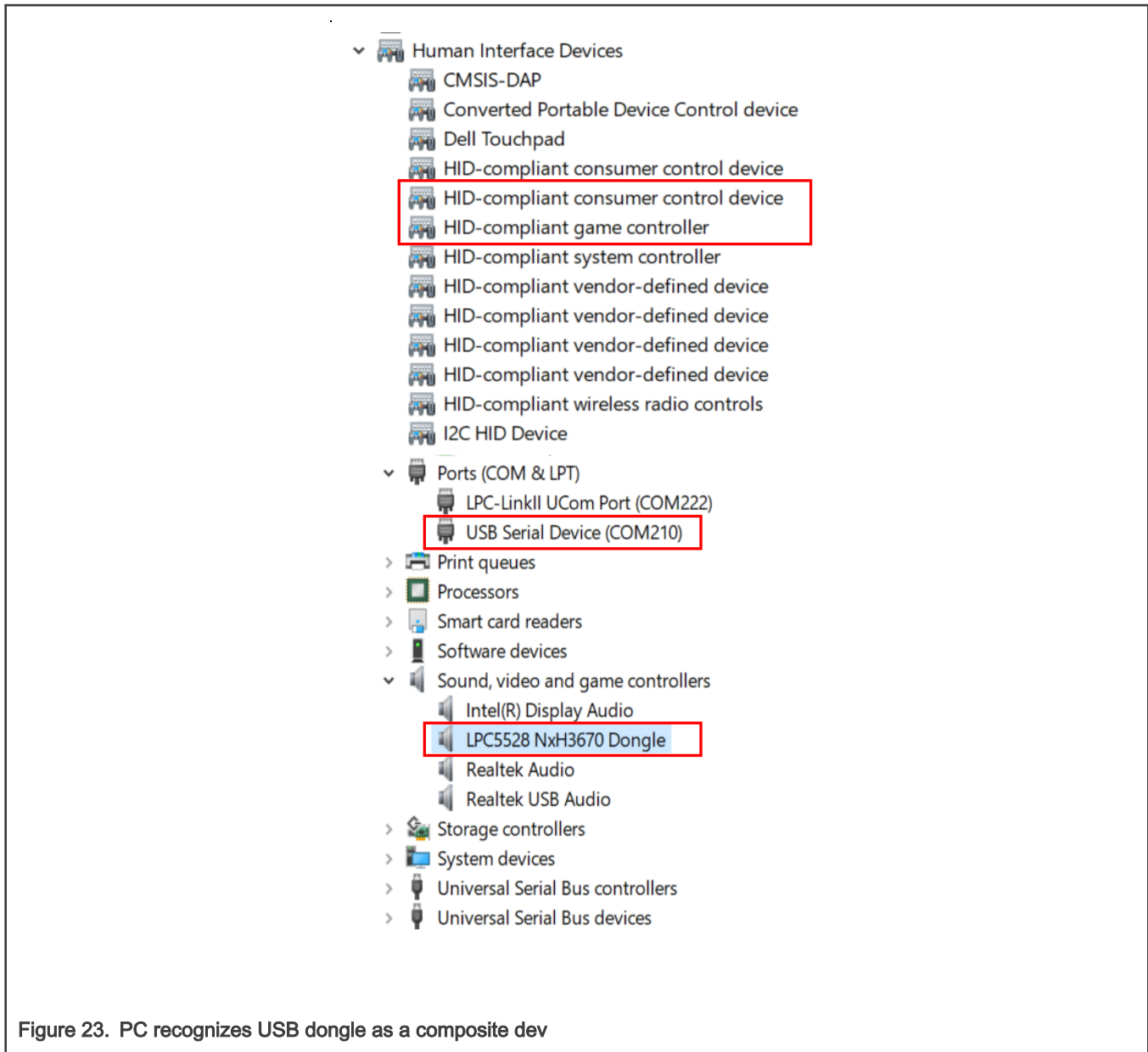


Figure 23. PC recognizes USB dongle as a composite dev

As shown in Figure 23, the USB dongle is a composite device integrating USB CDC + USB audio + USB HID + USB vendor specific class.

- Check the status of the red and green LED on the gamepad main board. If the red and green LED are OFF, it indicates that gamepad is in a low-power state or the lithium battery is low.
- Press the **START** button to try to wake up the gamepad in the low-power mode. If the red and green LED are ON, it indicates that the gamepad wakes up from the low-power mode and enters the normal run mode. By default, the gamepad works in the wireless mode. It will actively initiate the pairing process with the dongle. If the pairing is successful, the red LED on the dongle and gamepad main board will both be OFF. If the red and green led on the gamepad main board are still OFF after pressing the **START** button, it indicates that the gamepad's lithium battery is low. You need to use a micro USB cable to connect the PC and the gamepad, using the PC to charge the lithium battery. After the gamepad is started normally, the PC will recognize a composite USB device, as shown in Figure 24. Then gamepad will automatically initiate the pairing process with dongle. If the pairing is successful, the red LED on the dongle and gamepad main board will both be OFF. If the pairing process fails, the red LED on the gamepad is always ON and the gamepad is switched to the wired mode.

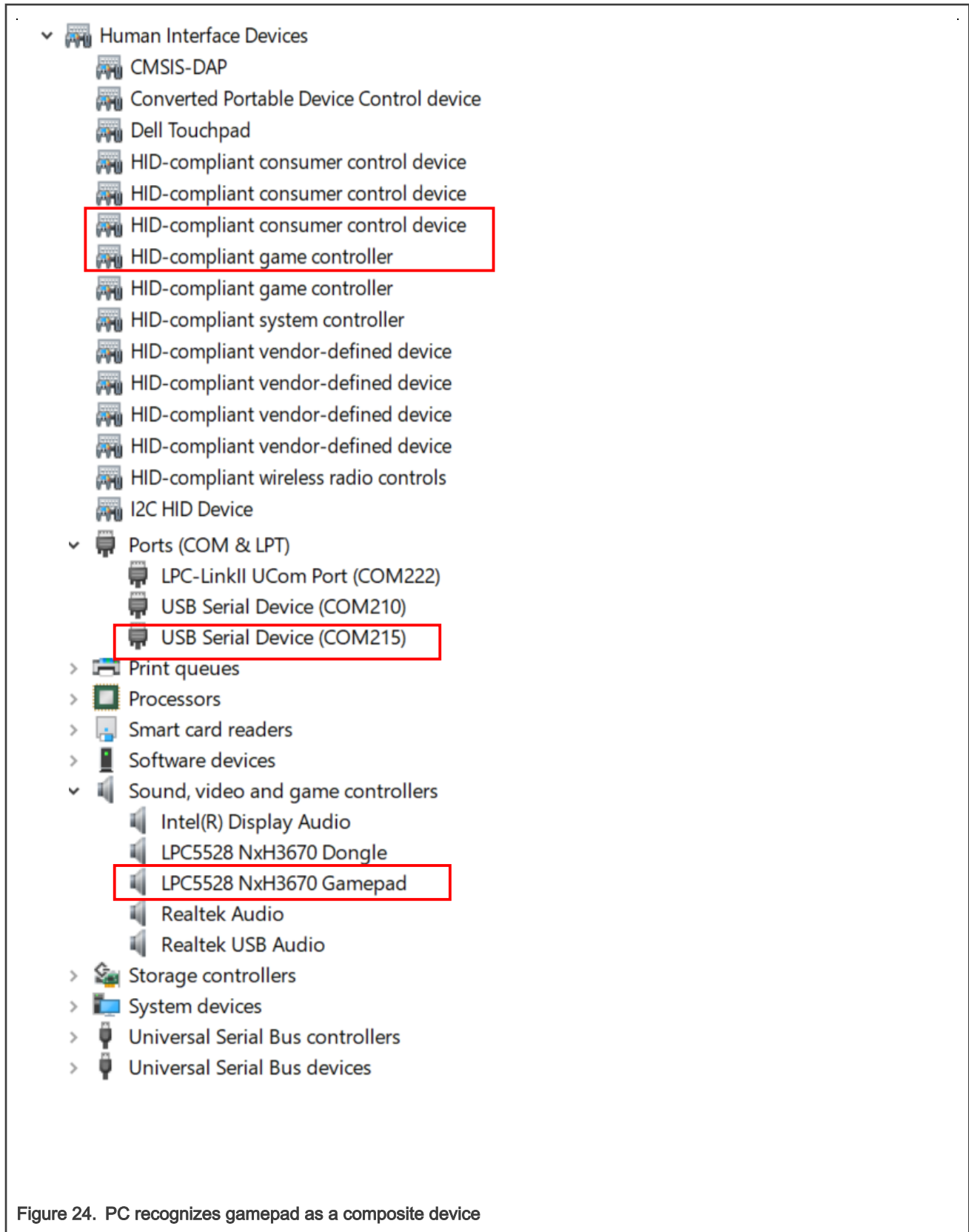


Figure 24. PC recognizes gamepad as a composite device

- Press the **XE + SATRT** buttons to switch the working mode between wired and wireless mode.
- Press the **XE + UP/DOWN** buttons to adjust the volume of the gamepad.

For more details about gamepad operation, see [LED state to Low-power mode](#).

4.2 Wired mode

Connecting the PC and the gamepad with a micro USB cable and no need to use LPC5528 USB dongle board. After the gamepad is started normally, the PC will recognize a composite USB device, as shown in [Figure 24](#). The gamepad will still initiate the pairing process with the dongle first, but the pairing process will fail because there is no dongle exist, and the gamepad automatically switches to wired mode.

4.3 LED state

[Table 1](#) describes the LED state of gamepad main board.

Table 1. LED state of gamepad main board

Work mode		RED	GREEN
Run mode	Wired	ON	ON
	Wireless	OFF	ON
Low power mode		OFF	OFF

[Table 2](#) describes the LED state of dongle board.

Table 2. LED state of dongle board

Work mode		RED	GREEN
Run mode	Wired	ON	ON
	Wireless	OFF	ON

4.4 Volume control

- Volume UP: **XE + UP**
- Volume Down: **XE + DOWN**

4.5 Mode switch

Press the **XE** button and then the **START** button to switch between wired mode and wireless mode.

4.6 Software reset

Press the **XE** button and hold it for ten seconds.

4.7 Low-power mode

If no button event is detected for five minutes, gamepad will enter the deep-sleep mode. To wake up the gamepad, press the **START** button.

5 Useful document

When developing this solution, documents provided in NxH3670 SDK Rev 5.2, as listed in [Table 3](#), were referenced.

Table 3. Documentation in NxH3670 SDK Rev5.2

Key words	Document name
OTA	<i>NxH3670_AN12361_OTA_Firmware_Updrade.pdf</i>
	<i>NxH3670_UM11203_HAPI_OTA.pdf</i>
VCOM tool	<i>NxH3670-0032_UM_HAPI_VCOM_Rev0.6.pdf</i>
Software Architecture	<i>NxH3670_AN12280_Reference_Application_Software_Architecture.pdf</i>
Performance	<i>NxH3670_0029_AN_Performance.pdf</i>
Host API	<i>NxH3670_AN11953_HAPI_Bootloader.pdf</i>
	<i>NxH3670_HAPI_overview.pdf</i>
	<i>NxH3670_UM11148_HAPI_Gaming.pdf</i>
	<i>NxH3670_UM11149_Host_Interface.pdf</i>
Flash Tool	<i>NxH3670_UM11198_FlashTool.pdf</i>

These documents cannot be downloaded separately on the NXP website and they are stored in the NxH3670 SDK Rev 5.2 package. For more development information, please refer to the complete document in the NxH3670 SDK.

6 Reference

- *Getting started with NxH3670 gaming use case* (document [AN12360](#))
- *Over-the-air firmware update* (document [AN12361](#))
- *NxH3670 Reference Application Software Architecture* (document AN12280)
- *NxH3670 Performance*
- *NxH3670 HAPI OTA* (document UM11203)
- *NxH3670 HAPI VCOM*
- *NxH3670 FlashTool* (document UM11198)

7 Revision history

Rev.	Date	Substantive changes
0	16 December, 2020	Initial release
1	21 May, 2021	Updated Figure 1 , Figure 2 , Figure 22 , Figure 23 and Figure 24

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.

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.

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, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, CodeWarrior, ColdFire, ColdFire+, the Energy Efficient Solutions logo, Kinetis, 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, μ Vision, 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. 2020-2021.

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: 21 May, 2021

Document identifier: AN13082

