AN12849

Linux Host Wake-up Implementation using Bluetooth or Bluetooth Low Energy (LE)

Rev. 3.0 — 16 September 2025 **Application note**

Document information

Information	Content
Keywords	Host wake-up, Bluetooth, Bluetooth Low Energy (LE), GPIO interrupt, firmware
Abstract	Describes Linux host wake-up using Bluetooth or Bluetooth Low Energy (LE) on NXP Wi-Fi and Bluetooth combo solutions (wireless SoCs)

1 Introduction

The document describes the steps for host wake-up using Bluetooth or Bluetooth Low Energy (LE) using:

- NXP proprietary UART driver *hci_uart.ko*
- NXP open source BTNXPUART driver (built-in Bluetooth driver for Linux Kernel versions greater than 6.12.20)

Host wake-up over Bluetooth/Bluetooth LE is also known as chip-to-host (C2H) wake-up.

The implementation described in this document assumes the following:

- The host CPU is powered down, and the host stack is not running.
- The Wi-Fi/Bluetooth module is powered on, and the firmware is running.
- The host has a GPIO handler that monitors the GPIO interrupt by firmware.
- Upon a GPIO interrupt, the host is brought to a running state and the host stack is initialized.

Note:

- *If using the BTNXPUART driver, the GPIO handler is implemented in the btnxpuart.c file. The configuration of the device tree binary (DTB) is detailed in Section 3.*
- *The GPIO handler implementation is not covered in this document for proprietary UART drivers.*

For example, in a typical TV-remote control pair setup, both the TV and the remote control have Bluetooth integrated. If the TV is in Standby mode, and the power button is pressed on the remote control:

- The remote control initiates a Bluetooth connection and sends a magic packet with a key code for "Power on".
- Upon receiving the magic packet, the firmware generates the GPIO interrupt to wake up the host.

The firmware is configured to generate a GPIO interrupt when a certain criterion is met. That is, when receiving the basic rate (BR)/enhanced data rate (EDR), and/or the Bluetooth LE connection request, and/or the advertising packet. Upon receiving the interrupt, the host wakes up.

Section 2.1 describes the different triggering points for the GPIO interrupt by firmware.

1.1 Supported products

Table 1 lists the NXP wireless products that support the feature.

Table 1. Supported products and Bluetooth drivers

Wireless product	NXP *hci_uart.ko* driver	NXP open source *btnxpuart.ko* driver
88W8887 (ref.[4])	Yes	—
88W8887 (Automotive) (ref.[5])	Yes	—
88W8897P (ref.[6])	Yes	—
88W8977 (ref.[7])	Yes	—
IW416 (ref.[8])	Yes	Yes
88W8987 (ref.[9])	Yes	Yes
88W8997 (ref.[10])	Yes	Yes
88Q9098 (ref.[11])	Software versions earlier than r9.x	Yes
88W9098 (ref.[12])	Software versions earlier than r9.x	Yes
AW590 (ref.[13])	Software versions earlier than r9.x	Yes
AW690 (ref.[14])	Software versions earlier than r9.x	Yes
IW610 (ref.[15])	—	Yes
AW611 (ref.[16])	—	Yes
IW611 (ref.[17])	—	Yes
IW612 (ref.[18])	—	Yes
AW692 (ref.[19])	—	Yes
AW693 (ref.[20])	—	Yes
IW693 (ref.[21])	—	Yes
IW623 (ref.[22])	—	Yes

2 GPIO interrupt by firmware

2.1 GPIO interrupt trigger

This section describes trigger points for the GPIO interrupt by firmware.

- Host wake-up over Bluetooth connectivity[1]:
  - Bluetooth Asynchronous Connectionless Link (ACL) connection wake-up:
    The host wake-up is triggered when the ACL connection request from the remote device is received.
  - Unicast Connectionless Data (UCD) based wake-up:
    The host wake-up is triggered when the UCD packet is received. The packet includes the payload with the predefined key code for "Power On".
- Host wake-up over Bluetooth LE connectivity:
  - Host wake-up with Bluetooth LE connection:
    After receiving the connection request from the peer devices (which are included in the allowlist), the firmware generates the interrupt on the configured GPIO pin to wake up the host.
  - Host wake-up with Bluetooth LE scanning:
    After receiving the advertising packet from the peer devices (which are included in the allowlist) or through the defined scan filters, the firmware generates the interrupt on the configured GPIO pin to wake up the host.
  - Host wake-up with RX data:
    The Bluetooth LE connection is active with the peer device and the host is in Sleep mode. After receiving the data packet from the peer devices, the firmware generates the interrupt on the configured GPIO pin to wake up the host.

1 Host wake-up over Bluetooth connectivity works up to kernel version 5.6. In later kernel versions, all Bluetooth activities are disabled when the host is in Suspend mode (ref.[3]).

2.2 GPIO pin polarity

The GPIO pin remains active high in Standby mode. When the firmware receives a trigger packet, the GPIO pin is pulled active low.

Figure 1 illustrates GPIO signal status.

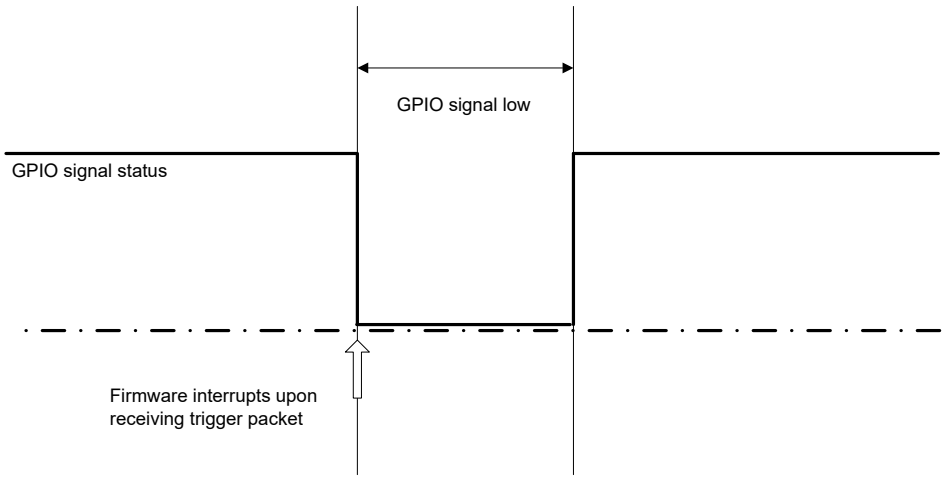

Figure 1. GPIO pin signal status when the firmware generates an interrupt

For wireless products with SDIO-SDIO host configuration, the GPIO pin is pulled active low for the GPIO_GAP time duration. For example: `gpio_gap=n`, bits[16:8] = `GPIO`, and bits[8:0] = `GPIO_GAP`

Where:

- `GPIO` is the GPIO pin number used to wake up the host. The number can be any valid GPIO pin number. If the GPIO pin value is set to 0xFF, the GPIO pin is not used for host wake-up. Instead, the wake-up happens through SDIO in-band signal. If the GPIO pin is not used, the GPIO_GAP value is ignored.
- When the GPIO_GAP value is set to a value comprised between more than 0 and less than 0xFF, for the time equal to the value, the firmware pulls GPIO pin active low. When GPIO_GAP is set to 0xFF, the GPIO pin is pulled active low until the host wakes up and enables the SDIO interface.

2.3 GPIO pin configuration

The GPIO pin configuration differs with the wireless product and interface for Bluetooth. The GPIO pin can also be changed based on availability.

Table 2. GPIO pin configuration of the supported wireless products

Wireless product	Controller-to-host GPIO configuration pin
88W8887	GPIO[13]
88W8887A	GPIO[0][1]
88W8897	GPIO[13]
88W8977	GPIO[13]
IW416	GPIO[12]
88W8987	GPIO[4] or GPIO[20]
88W8997	GPIO[12]
88Q9098	GPIO[16]
88W9098	GPIO[16]
AW590	GPIO[16]
AW690	GPIO[16]
IW610	GPIO[5]
AW611	GPIO[19]
IW611	GPIO[19]
IW612	GPIO[19]
AW692	GPIO[10]
AW693	GPIO[10]
IW693	GPIO[10]
IW623	GPIO[10]

[1] GPIO[13] is not available. Use GPIO[0] for the 88W8887 Automotive QFN package

3 DTB file configuration for BTNXPUART driver

BTNXPUART from Linux kernel version 6.12.20 and greater supports host wake-up over Bluetooth/Bluetooth LE. To configure btnxpuart driver for NXP wireless products, modify the device tree file (dts) and compile the device tree binary (dtb).

Note: *If the default btnxpuart and the default dts and dtb files are used, the host wake-up feature does not work.*

Table 3. Parameters to configure in the dts file

Parameter	Description
`interrupt-parent`	Set the host GPIO bank used for wake-up/interrupt. Syntax: `interrupt-parent = <&gpioX>;` Where X refers to the host parent GPIO bank. Example for i.MX 8M Mini GPIO[3]: `interrupt-parent = <&gpio3>;`
`interrupts`	Host GPIO pin configured for the interrupts triggered with a falling edge. Refer to the figure in Section 2.1. Syntax: `interrupts = <X IRQ_TYPE_EDGE_FALLING>;` Where X refers to the host GPIO pin number. Example for i.MX 8M Mini GPIO3_IO24: `interrupts = <24 IRQ_TYPE_EDGE_FALLING>;`
`interrupt-names`	Enables host wake-up over Bluetooth/Bluetooth LE. Syntax: `interrupt-names = "wakeup";`
`wakeup-source`	Enables host wake-up over Bluetooth/Bluetooth LE. Syntax: `wakeup-source;`
`nxp, wakeout-pin`	Indicates the GPIO pin number of the wireless product. Syntax: `nxp,wakeout-pin = /bits/ 8 <X>;` Where X is the GPIO pin number (Section 2.3). Example for GPIO[19]: `nxp,wakeout-pin = /bits/ 8 <19>;`

Example of *dts* configuration for i.MX 8M Mini and IW612:

```
serial {
    bluetooth {
         compatible = “nxp,88w8987-bt”;
         nxp,wakeout-pin = /bits/ 8 <19>;
         interrupt-parent = <&gpio3>;
         interrupts = <24 IRQ_TYPE_EDGE_FALLING>;
         interrupt-names = "wakeup";
         wakeup-source;
         };
};
```

Compile the *dts* file into a *dtb* file in the build environment.

```
dtc -O dtb -o imx8xx-evk-xxx.dtb imx8xx-evk-xxx.dts
```

Replace the existing *dtb* file on the host platform with the newly generated *dtb* file.

```
cp imx8xx-evk-xxx.dtb /run/media/boot-mmcblk0p1/imx8xxx-evk.dtb
```

4 Host wake-up over Bluetooth

Host wake-up over Bluetooth connectivity is supported up to kernel version 5.6. In later kernel versions, all Bluetooth activities are disabled when the host is in Suspend mode (ref.[3]).

4.1 Bluetooth ACL connection wake-up host

In this implementation, the host wake-up is triggered when there is an ACL connection request over any Bluetooth remote device.

The procedure is as follows:

1. To accept the incoming ACL connection request, enable page scan on the device under test (DUT).

```
hciconfig hci0 pscan
```

2. Configure the GPIO pin for the interrupt.
   The firmware generates an interrupt on the configured GPIO pin. The GPIO pin configuration differs with the wireless products and interface used for Bluetooth. See Section 6.
3. Enable host sleep.
   Different interfaces require a different host sleep command. See Section 7.
4. Initiate the ACL connection from the remote device to the DUT.

```
hcitool -i hci0 cmd cc <BD_Address_DUT>
```

   The host monitors the configured GPIO pin for interrupts. After receiving the ACL connection request, the controller generates the interrupt on the configured GPIO pin. Once the interrupt is detected, the host is in active mode.

4.2 UCD-based host wake-up

In this implementation, the host wake-up is triggered only when the unique Bluetooth UCD packet is received. The packet includes the payload with the predefined key code for the power button. The predefined key code for the power button must be supported on both Bluetooth modules (for example TV and remote control).

Table 4 shows the format for the UCD packet. The protocol service multiplexer (PSM) value to create the logical link control and adaptation protocol (L2CAP) connection is vendor-specific. The PSM value can be changed.

Table 4. L2CAP header format

	Hex Value	Bits								Note
		7	6	5	4	3	2	1	0	
First	0x03	0	0	0	0	0	0	1	1	Length of PSM and payload
Second	0x00	0	0	0	0	0	0	0	0	
Third	0x02	0	0	0	0	0	0	1	0	Channel ID (0x0002 = connectionless traffic)
Fourth	0x00	0	0	0	0	0	0	0	0	
Fifth	0x11	0	0	0	0	0	0	1	1	PSM (0x1011 = vendor-specific value)
Sixth	0x10	0	0	0	0	0	0	1	0	
Seventh	0x82	1	0	0	0	0	0	1	0	Payload (0x82 = key code for power button)

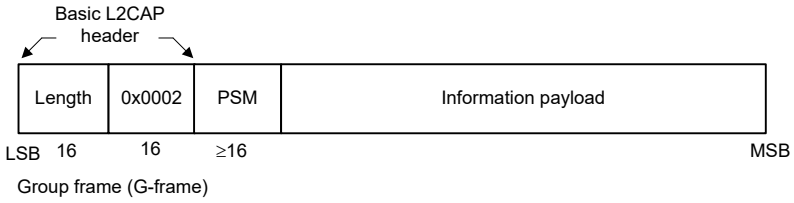

Figure 2. L2CAP PDU format on connectionless channel

4.2.1 Example

Figure 3 shows the flow for UCD-based host wake-up. The TV is in Sleep mode until the remote power key is pressed. The NXP wireless module is integrated in the TV and communicates with the remote control over Bluetooth.

When the power key on the remote control is pressed, the remote control establishes the baseband connection with the controller (wireless product). When the connection is established, the remote control sends a unique UCD packet with a payload that includes the key code for ‘Power On’.

The controller validates the key code. After the key code verification, the controller asserts the GPIO and discards the packet.

After detecting that the GPIO is asserted, the TV wakes up the system. Once the system is active, the controller starts sending packets to the host.

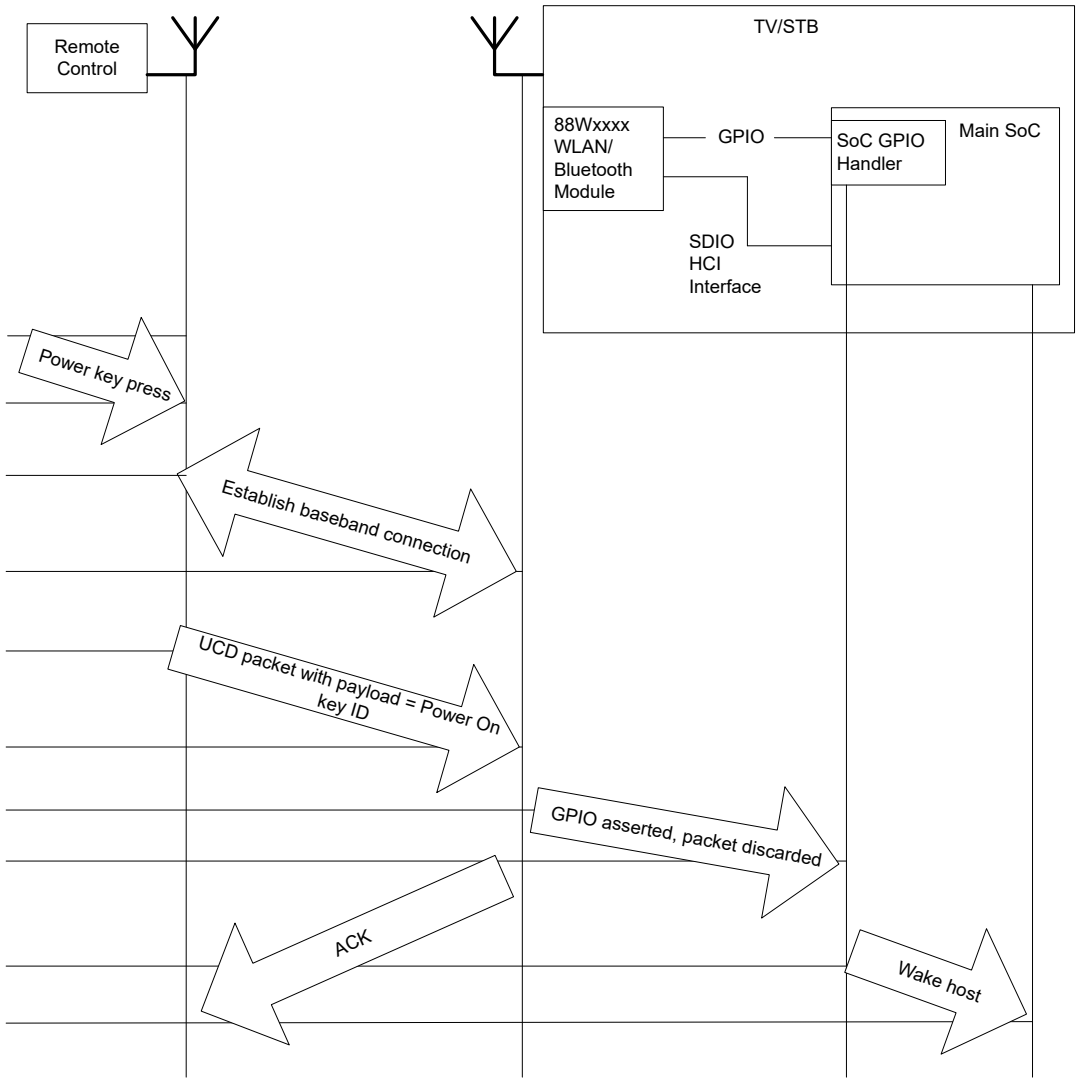

Figure 3. UCD-based host wake-up flow

4.2.2 Possible scenarios

The possible scenarios of the UCD wake-up procedure are:

- Scenario 1—Host standby push the power key
- Scenario 2—Host standby push another key
- Scenario 3—Active to standby status

4.2.2.1 Scenario 1—Host standby, push the power key

The host is in Standby mode. The Bluetooth module (host) waits for the connection from the paired Bluetooth module (remote). After the power key is pressed, the Bluetooth module (remote) initiates the connection. When the baseband connection is created, the authentication procedure begins.

1. For authentication, the host writes the link key of the Bluetooth module (remote) using the `HCI_Write_Stored_Link_Key` command to store the `linkkey` in the controller.
2. To avoid waking up the host to obtain the `linkkey` from the host, the controller uses the stored `linkkey`.
3. After successful authentication, the Bluetooth module (remote) sends the UCD packet with the payload. The payload includes the key code for ‘Power On’.
4. The Bluetooth module (host) parses the key code and asserts the GPIO.
5. The GPIO header module monitors the GPIO and initiates the host wake-up upon the GPIO assertion.

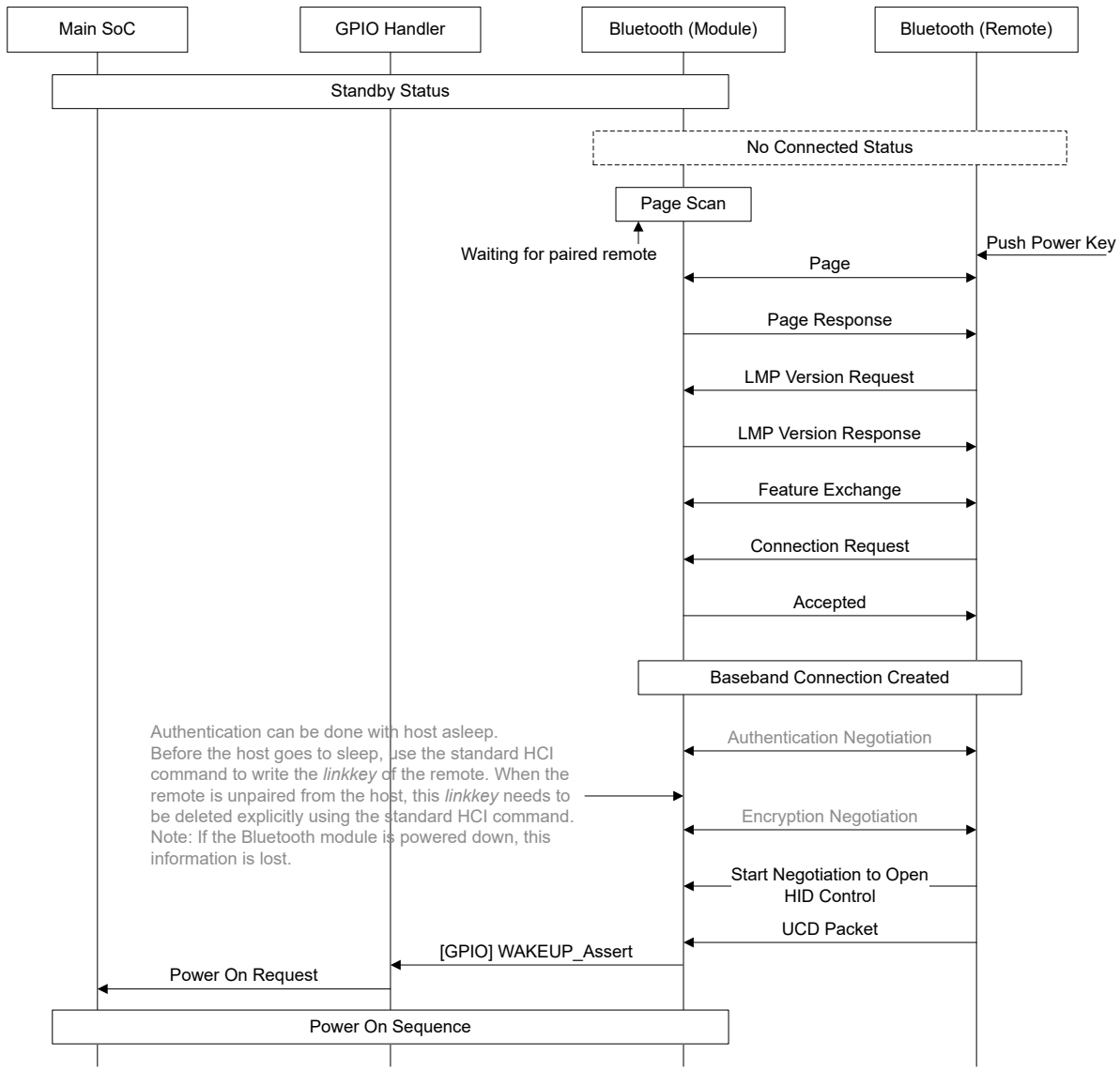

Figure 4. Host Standby Push Power Key Flow

4.2.2.2 Scenario 2—Host Standby, push another key

The host is in Standby mode. After the baseband connection is up, another key than the power key is pressed. The Bluetooth module (host) waits for 'No Traffic Timeout' to receive the UCD packet with the key code for 'Power On'.

If the UCD packet with the key code is not received, the Bluetooth module (host) stops the baseband connection. The Bluetooth module does not wake up the host.

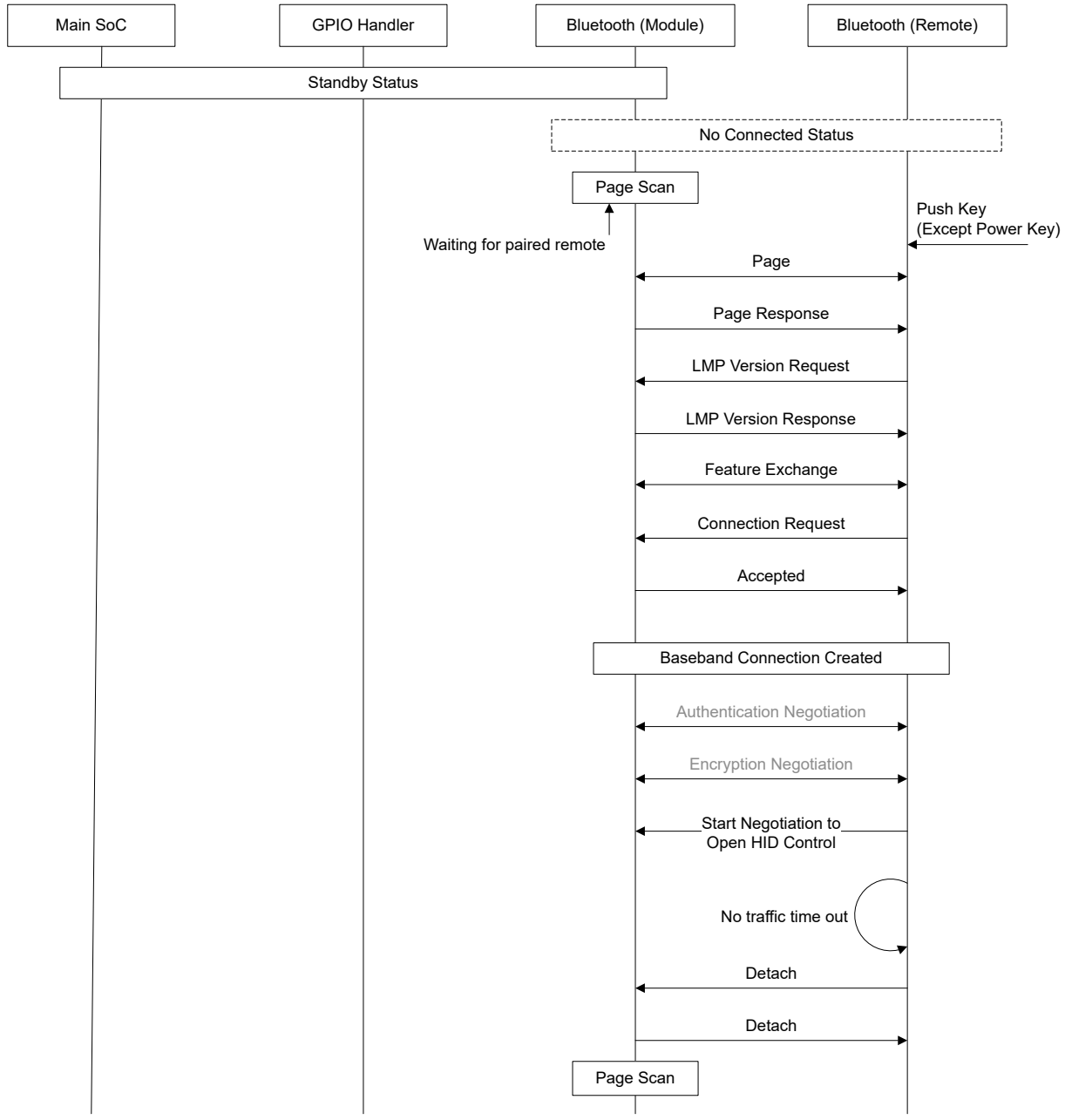

Figure 5. Flow for host in Standby mode – Any key (other than power) is pushed

4.2.2.3 Scenario 3—Active to Standby status

The host is active and Bluetooth is connected. The Bluetooth module (host) parses the pressed keys from the Bluetooth module (remote) and forwards the information to the host. If the power key is pressed, the Bluetooth module (remote) sends the UCD packet.

After receiving the UCD packet, the Bluetooth module (host) asserts the GPIO. The GPIO handler remembers the last power state for the host. If the host was last in active state, the GPIO handler sends the power off request. The standby sequence follows.

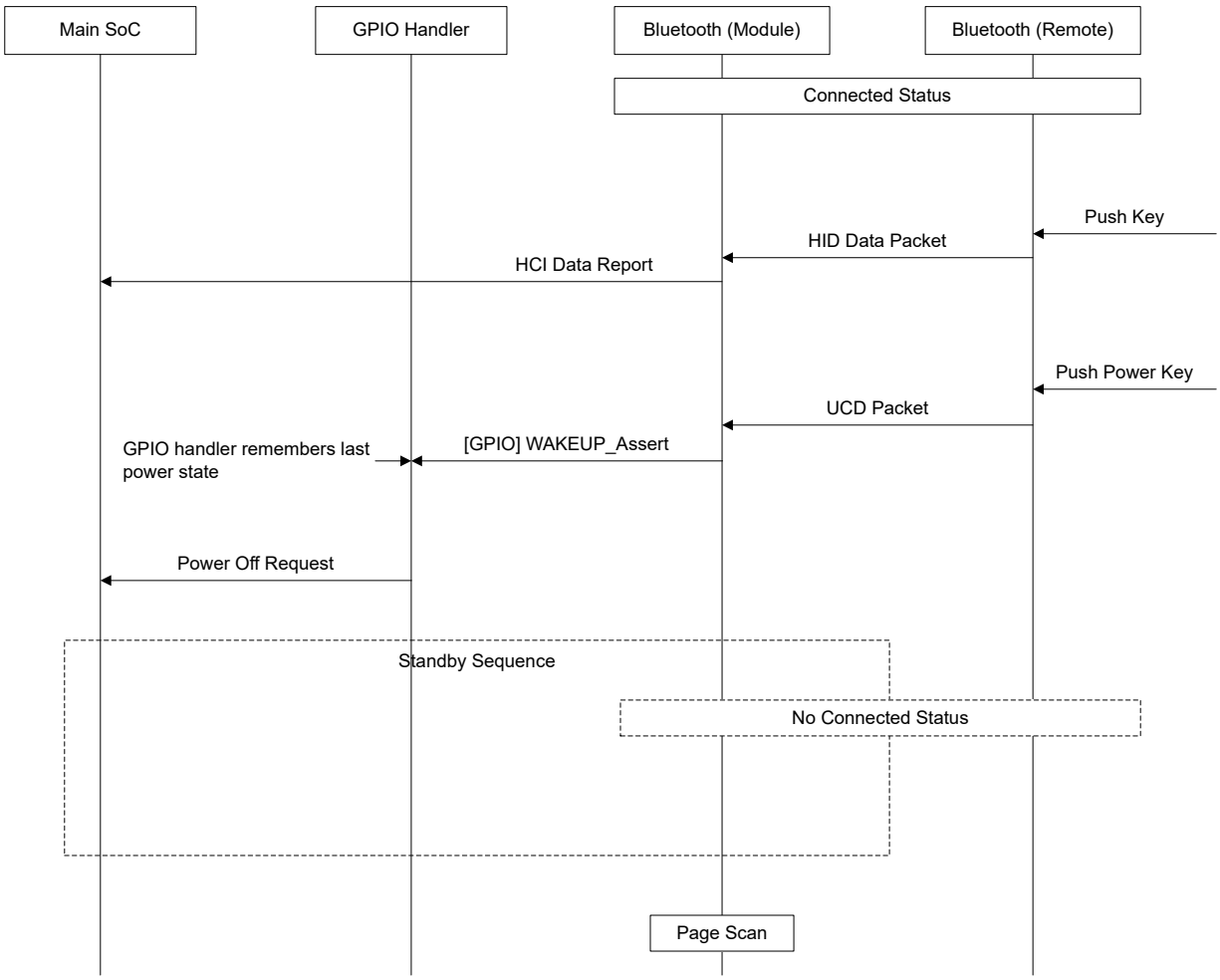

Figure 6. Active to Standby Status Flow

5 Host wake-up over Bluetooth LE

5.1 Host wake-up over Bluetooth LE connection

In this case, the controller is advertising while the host is in Sleep mode. When there is a Bluetooth LE connection request from the remote device (which is in the allowlist), the host wakes up. The sequence is as follows:

Step 1 – For BTNXPUART driver, configure the host and controller GPIO pins in the *.dtb* file (Section 3).

The firmware generates an interrupt on the configured controller GPIO pin. The GPIO pin differs with the wireless product and the interface used for Bluetooth. See Section 6 for the commands.

Step 2 – Add the peer device to the DUT allowlist using `HCI_LE_Add_Device_To_Filter_Accept_List` command.

```
hcitool -i hci0 cmd 0x08 0x0011 <Address_Type> <BD_Address>
```

Table 5. Command parameters

Parameter	Description
`Address_Type`	The device address type 0x00 = public device address 0x01 = random device address
`BD_Address`	The public or random Bluetooth device address to be added to the allowlist

Step 3 – Start advertising on the DUT.

1. Set the advertising parameter.
   - Advertising type: connectable low-duty cycle directed advertising (0x04)
   - Peer device address in Little Endian format and advertising policy to process scan
   - Connection requests only from devices in the allowlist (0x03)

   ```
   hcitool -i hci0 cmd 08 06 00 02 00 02 04 00 00 <Peer_Device_Address> 07 03
   ```
2. Set the advertising data.

   ```
   hcitool -i hci0 cmd 08 08 1F 00 99 88 77 66 55 44 33 22 11 00 99 88 77 66 55 44 33 22
    11 00 99 88 77 66 55 44 33 22 11 00
   ```
3. Enable advertising.

   ```
   hcitool -i hci0 cmd 08 0A 01
   ```

Step 4 – Enable Set event mask for Bluetooth LE.

```
hcitool -i hci0 cmd 0x03 0001 ff ff ff ff ff ff ff ff
```

Step 5 – Enable host Sleep mode.

The host sleep command differs with the interface. See Section 7 for the commands.

Step 6 – Initiate the Bluetooth LE connection from the remote device with `Initiator_Filter_Policy` set to the allowlist.

Expected results

After receiving the Bluetooth LE connection request from the peer device (which is in the allowlist), the controller accepts the Bluetooth LE connection request and generates the interrupt on the configured GPIO pin.

5.2 Host wake-up over Bluetooth LE scanning

The procedure to wake up the host over Bluetooth LE scanning is:

Step 1 – For BTNXPUART driver, configure the host and controller GPIO pins in the *.dtb* file (Section 3).

The firmware generates an interrupt on the configured controller GPIO pin. The GPIO pin differs with the wireless product and the interface used for Bluetooth. See Section 6 for the commands.

Step 2 – Enable `Set event mask` for Bluetooth LE.

```
hcitool -i hci0 cmd 0x03 0001 ff ff ff ff ff ff ff ff
```

Step 3 – Add the peer device to DUT allowlist using `HCI_LE_Add_Device_To_Filter_Accept_List` command.

```
hcitool -i hci0 cmd 0x08 0x0011 <Address_Type> <BD_Address>
```

Table 6. Command parameters

Parameter	Description
`Address_Type`	The device address type 0x00 = public device address 0x01 = random device address
`BD_Address`	The public or random Bluetooth device address to be added to the allowlist

Step 4 – Set the scan parameters on the DUT.

```
hcitool -i hci0 cmd 0x08 0x000B <Parameters>
```

Table 7. Command parameters

Parameter	Description
`Parameters`	LE_Scan_Type LE_Scan_Interval LE_Scan_Window Own_AddresS_Type Scanning_Filter_Policy (0x01 = set to allowlist) Other parameters are detailed in the Bluetooth specification.

Step 5 – Enable Bluetooth LE scan on the DUT.

```
hcitool -i hci0 cmd 0x08 0x000C 0x01 0x01
```

Step 6 – Enable host Sleep mode.

The host sleep command differs with the interface. See Section 7 for the commands.

Step 7 – Start advertising on the peer device.

1. Set the advertising parameter.
   - Advertising type: connectable low-duty cycle directed advertising (0x04)
   - Peer device address and advertising policy to process scan
   - Connection requests only from devices in the allowlist (0x03)

```
hcitool -i hci0 cmd 08 06 00 02 00 02 04 00 00 <Peer_Device_Address> 07 03
```

2. Set the advertising data.

```
hcitool -i hci0 cmd 08 08 1F 00 99 88 77 66 55 44 33 22 11 00 99 88 77 66 55 44 33 22
 11 00 99 88 77 66 55 44 33 22 11 00
```

3. Enable advertising.

```
hcitool -i hci0 cmd 08 0A 01
```

Expected results

After receiving the Bluetooth LE advertising packet from the peer device (which is in the allowlist), the controller generates the interrupt on the configured GPIO pin.

5.3 Active Bluetooth LE connection – Host wake-up with RX data packet

The controller has an active Bluetooth LE connection and the host is in Sleep mode. When the remote device sends a data packet to the controller, the interrupt is sent by the controller GPIO pin to wake up the host.

Step 1 – For BTNXPUART driver, configure the host and controller GPIO pins in the *.dtb* file (Section 3).

The firmware generates an interrupt on the configured controller GPIO pin. The GPIO pin differs with the wireless product and the interface used for Bluetooth. See Section 6 for the commands.

Step 2 – Enable Set event mask for Bluetooth LE.

```
hcitool -i hci0 cmd 0x03 0001 ff ff ff ff ff ff ff ff
```

Step 3 – Add the peer device to DUT allowlist using `HCI_LE_Add_Device_To_Filter_Accept_List` command.

```
hcitool -i hci0 cmd 0x08 0x0011 <Address_Type> <BD_Address>
```

Table 8. Command parameters

Parameter	Description
`Address_Type`	The device address type 0x00 = public device address 0x01 = random device address
`BD_Address`	The public or random Bluetooth device address to be added to the allowlist

Step 4 – Start advertising on the DUT.

1. Set the advertising parameter.
   - Advertising type: connectable low-duty cycle directed advertising (0x04)
   - Peer device address and advertising policy to process scan
   - Connection requests only from devices in the allowlist (0x03)

   ```
   hcitool -i hci0 cmd 08 06 00 02 00 02 04 00 00 <Peer_Device_Address> 07 03
   ```
2. Set the advertising data.

   ```
   hcitool -i hci0 cmd 08 08 1F 00 99 88 77 66 55 44 33 22 11 00 99 88 77 66 55 44 33 22
    11 00 99 88 77 66 55 44 33 22 11 00
   ```
3. Enable advertising.

   ```
   hcitool -i hci0 cmd 08 0A 01
   ```

Step 5 – Initiate a Bluetooth LE connection from the remote device.

Step 6 – Enable host Sleep mode.

The host sleep command differs with the interface. See Section 7 for the commands.

Step 7 – Send any Bluetooth LE data packet from the remote device.

Example of command on Bluez stack to send a packet from the remote device:

```
hcitool -i hci1 acldat -p dcd -d 0x5454 -c 1 -s 10 -P i <BD_Addr_Dut> -H 128
```

Expected results

The controller receives the data packet from the remote device and generates the interrupt on the configured GPIO pin.

6 Commands to configure the GPIO pins

The command includes parameters for the wireless product and for the following host interface configurations:

- SDIO-SDIO: the Wi-Fi interface is SDIO and the Bluetooth interface is SDIO.
- PCIe-UART: the Wi-Fi interface is PCIe and the Bluetooth interface is UART.
- SDIO-UART: the Wi-Fi interface is SDIO and the Bluetooth interface is UART.
- USB-USB: the Wi-Fi interface is USB and the Bluetooth interface is USB.

For the wireless products that support BTNXPUART driver:

- BTNXPUART driver for host wake-up applies to Bluetooth UART interface only.
- The controller GPIO pin is configured in the DTS file (Section 3). By default, the BTNXPUART driver sends the GPIO pin configuration command during the *init* stage.

Table 9. Commands to configure GPIO pin with UART interface for Bluetooth

Wireless product	Interface	Command	GPIO
88W8887	SDIO-UART (Automotive)	`hcitool -i hci0 cmd 0x3F 0x53 0x04 0x00 0x01 0xFF`	GPIO[0]
	SDIO-UART (IoT)	`hcitool -i hci0 cmd 0x3F 0x53 0x03 0x0D 0x01 0xFF`	GPIO[13]
88W8897	PCIe-UART	`hcitool -i hci0 cmd 0x3F 0x53 0x04 0x0C 0x01 0xFF`	GPIO[12]
88W8977	SDIO-UART	`hcitool -i hci0 cmd 0x3F 0x53 0x03 0x0D 0x01 0xFF`	GPIO[13]
IW416	SDIO-UART	`hcitool -i hci0 cmd 0x3F 0x53 0x04 0x0C 0x01 0xFF`	GPIO[12]
88W8987	SDIO-UART	`hcitool -i hci0 cmd 0x3F 0x53 0x03 0x04 0x01 0xFF`	GPIO[4]
88W8997	PCIe-UART	`hcitool -i hci0 cmd 0x3F 0x53 0x03 0x0C 0x01 0xFF`	GPIO[12]
88Q9098/88W9098	SDIO-UART/ PCIe-UART (88Q9098)	`hcitool -i hci0 cmd 0x3F 0x53 0x03 0x10 0x01 0xFF`	GPIO[16]
	SDIO-UART/ PCIe-UART (88W9098)		
AW590	PCIe-UART	`hcitool -i hci0 cmd 0x3F 0x53 0x03 0x10 0x01 0xFF`	GPIO[16]
AW690	PCIe-UART	`hcitool -i hci0 cmd 0x3F 0x53 0x03 0x10 0x01 0xFF`	GPIO[16]
IW610	SDIO-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x05 0x01 0xFF`	GPIO[5]
AW611	SDIO-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x13 0x01 0xFF`	GPIO[19]
IW611	SDIO-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x13 0x01 0xFF`	GPIO[19]
IW612	SDIO-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x13 0x01 0xFF`	GPIO[19]
AW692	PCIe-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x0A 0x01 0xFF`	GPIO[10]

Table 9. Commands to configure GPIO pin with UART interface for Bluetooth*...continued*

Wireless product	Interface	Command	GPIO
AW693	PCIe-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x0A 0x01 0xFF`	GPIO[10]
IW693	PCIe-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x0A 0x01 0xFF`	GPIO[10]
IW623	PCIe-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x0A 0x01 0xFF`	GPIO[10]
	SDIO-UART	`hcitool -ihci0 cmd 0x3F 0x53 0x03 0x0A 0x01 0xFF`	GPIO[10]

Table 10. Commands to configure GPIO pin with SDIO interface for Bluetooth

Wireless product	Interface	Command	GPIO

88W8887	SDIO-SDIO (Automotive)	`echo "gpio_gap=0x0064" /proc/mbt/hci0/config`
		or
		`hcitool -i hci0 cmd 0x3F 0x59 0x00 0x64`
			GPIO[0]

	SDIO-SDIO (Non-automotive)	`echo "gpio_gap=0x0D64" /proc/mbt/hci0/config`
		or
		`hcitool -i hci0 cmd 0x3F 0x59 0x0D 0x64`
			GPIO[13]

88W8897	SDIO-SDIO	`echo "gpio_gap=0x0D64" /proc/mbt/hci0/config`
		or
		`hcitool -i hci0 cmd 0x3F 0x59 0x0D 0x64`
			GPIO[13]

88W8977	SDIO-SDIO	`echo "gpio_gap=0x0D64" /proc/mbt/hci0/config`
		or
		`hcitool -i hci0 cmd 0x3F 0x59 0x0D 0x64`
			GPIO[13]

88W8987	SDIO-SDIO	`echo "gpio_gap=0x0432" /proc/mbt/hci0/config`
		or
		`hcitool -i hci0 cmd 0x3F 0x59 0x04 0x64 0x01`
			GPIO[4]

Table 11. Commands to configure GPIO pin with USB interface for Bluetooth

Wireless product	Interface	Command	GPIO
88W8897	USB-USB	`hcitool -i hci0 cmd 0x3F 0x59 0x0D 0x64`	GPIO[13]
88W8997	USB-USB	`hcitool -i hci0 cmd 0x3F 0x59 0x0C 0x64`	GPIO[12]
IW610	USB-USB	`hcitool -i hci0 cmd 0x3F 0x59 0x05 0x64`	GPIO[5]

7 Command to configure host Sleep mode

The command includes parameters for the wireless product and for the following host interface configurations:

- PCIe-UART: the Wi-Fi interface is PCIe and the Bluetooth interface is UART.
- SDIO-UART: the Wi-Fi interface is SDIO and the Bluetooth interface is UART.
- USB-USB: the Wi-Fi interface is USB and the Bluetooth interface is USB.

For the wireless products that support BTNXPUART driver:

- BTNXPUART driver for host wake-up applies to Bluetooth UART interface only.
- The GPIO is configured as part of DTS file (Section 3). By default, the BTNXPUART driver sends the GPIO pin configuration command during the *init* stage.

Table 12. Commands to configure host Sleep mode using proprietary hci_uart.ko driver – UART interface for Bluetooth

Wireless product	Interface	Command

88W8887	SDIO-UART (Automotive)	For host Sleep mode:
		`echo "psmode=1" > /proc/mbt_uart/hci0/config`
		For host Suspend mode:
		`systemctl suspend`

	SDIO-UART (Non-Automotive)	
88W8897	PCIe-UART	
88W8977	SDIO-UART	
IW416	SDIO-UART	
88W8987	SDIO-UART	
88W8997	PCIe-UART	
88Q9098/ 88W9098	SDIO-UART and/or PCIe-UART (88Q9098)	
	SDIO-UART and/or PCIe-UART (88W9098)	
AW590/AW690	PCIe-UART	
IW610	SDIO-UART	
AW611/IW611/IW612	SDIO-UART	
AW692/AW693/IW693/IW623	PCIe-UART	
IW623	SDIO-UART	

Table 13. Commands to configure host Sleep mode – SDIO interface for Bluetooth

Wireless product	Interface	Command

88W8887	SDIO-SDIO (Automotive)	`echo "hsmode=1" /proc/mbt/hci0/config`
		`echo "hscmd=1" /proc/mbt/hci0/config`
		or
		`hcitool cmd -ihci0 0x3F 0x5A 0x00`

	SDIO-SDIO (Non-Automotive)	
88W8897	SDIO-SDIO	
88W8977	SDIO-SDIO	
88W8987	SDIO-SDIO	
88W8997	SDIO-SDIO	

Table 14. Commands to configure host Sleep mode with USB interface

Wireless product	Interface	Command

88W8897	USB-USB	For host Sleep mode:
		`./mlanutl mlan0 usbsuspend`
		For host Suspend mode:
		`systemctl suspend`

88W8997	USB-USB	
IW610	USB-USB	

8 Examples

8.1 Host wake-up over Bluetooth LE connection on AW692 / AW693 PCIE-UART using BTNXPUART

The Bluetooth controller continues advertising while the host remains in a sleep state. When the Bluetooth LE connection request is received from a remote device (listed in the allowlist), the host wakes up.

Step 1 – Configure the host and controller GPIO pins in *.dtb* file (Section 3).

Step 2 – Load the Bluetooth firmware using BTNXPUART.

```
modprobe btnxpuart
```

Step 3 – Add the peer device to the DUT allowlist.

```
hcitool -i hci0 cmd 0x08 0x0011 <Address_Type> <BD_Address>
```

Table 15. Command parameters

Parameter	Description
Address_Type	The device address type 0x00 = public device address 0x01 = random device address
BD_Address	The public or random Bluetooth device address to be added to the allowlist

Step 4 – Start advertising on the DUT.

- Set the advertising parameter

The advertising type is set to connectable low duty cycle directed advertising (0x04), with the peer device address and advertising policy to process scan, and connection requests only from devices in the allowlist (0x03).

```
hcitool -i hci0 cmd 08 06 00 02 00 02 04 00 00 <Peer_Device_Address> 07 03
```

- Set the advertising data

```
hcitool -i hci0 cmd 08 08 1F 00 99 88 77 66 55 44 33 22 11 00
99 88 77 66 55 44 33 22 11 00 99 88 77 66 55 44 33 22 11 00
```

- Enable advertising

```
hcitool -i hci0 cmd 08 0A 01
```

Step 5 – Enable `Set event mask` for Bluetooth LE.

```
hcitool -i hci0 cmd 0x03 0001 ff ff ff ff ff ff ff ff
```

Step 6 – Enable host suspend.

```
systemctl suspend
```

Step 7 – Set up the Bluetooth LE connection from the remote device

```
hcitool –i hci1 lecc 00:50:43:21:30:CF
```

Where `00:50:43:21:30:CF` is the DUT address

Expected result

After receiving the Bluetooth LE connection request from the peer device (which is in the allowlist), the controller accepts the Bluetooth LE connection request and generates the interrupt on the configured GPIO pin.

8.2 Host wake up over Bluetooth LE scanning IW611/IW612 SD-UART using BTNXPUART

The controller performs a Bluetooth LE scan while the host remains in sleep mode. When an advertising (ADV) packet is received from a remote device listed in the allowlist, the host wakes up.

Step 1 – Configure the host and controller GPIO pins in *.dtb* file (Section 3).

Step 2 – Load the Bluetooth firmware using BTNXPUART.

```
modprobe btnxpuart
```

Step 3 – Add the peer device to the DUT allowlist.

```
hcitool -i hci0 cmd 0x08 0x0011 <Address_Type> <BD_Address>
```

Table 16. Command parameters

Parameter	Description
`Address_Type`	The device address type 0x00 = public device address 0x01 = random device address
`BD_Address`	The public or random Bluetooth device address to be added to the allowlist

Step 4 – Enable `Set event mask` for Bluetooth LE.

```
hcitool -i hci0 cmd 0x03 0001 ff ff ff ff ff ff ff ff
```

Step 5 – Set the scan parameters on the DUT.

```
hcitool -i hci0 cmd 08 0B 01 04 00 04 00 00 01
```

Step 6 – Enable Bluetooth LE scanning on the DUT

```
hcitool -i hci0 cmd 0x08 0x000C 0x01 0x01
```

Step 7 – Enable Host sleep suspend.

```
systemctl suspend
```

Step 8 – Start advertising on the peer device newly added to the DUT allowlist.

- Set the advertising parameter.

```
hcitool -i hci0 cmd 08 06 00 02 00 02 04 00 00 <Peer_Device_Address> 07 03
```

- Set the advertising data.

```
hcitool -i hci0 cmd 08 08 1F 00 99 88 77 66 55 44 33 22 11 00
99 88 77 66 55 44 33 22 11 00 99 88 77 66 55 44 33 22 11 00
```

- Enable advertising.

```
hcitool -i hci0 cmd 08 0A 01
```

Expected result

After receiving the Bluetooth LE advertising packet from the peer device (added to the DUT allowlist), the DUT generates the interrupt on the configured GPIO pin.

8.3 Host wake-up over Bluetooth LE connection on 88Q9098/88W9098 SD-UART using *hci_uart.ko* driver

The controller is advertising while the host is in Sleep mode. When the remote Bluetooth device (in the allowlist) requests a Bluetooth LE connection, the Host wakes up.

Step 1 – Add the peer device to the DUT allowlist.

```
hcitool -i hci0 cmd 0x08 0x0011 <Address_Type> <BD_Address>
```

Table 17. Command parameters

Command	Description
`Address_Type`	The device address type 0x00 = public device address 0x01 = random device address
`BD_Address`	The public or random Bluetooth device address to be added to the allowlist

Step 2 – Configure the GPIO pin.

```
hcitool -i hci0 cmd 0x3F 0x53 0x03 0x10 0x01 0xFF
```

Step 3 – Start advertising on the DUT.

1. Set the advertising parameters.
   - Advertising type: connectable low-duty cycle directed advertising (0x04)
   - Peer device address and advertising policy to process scan
   - Connection requests only from devices in the allowlist (0x03)

   ```
   hcitool -i hci0 cmd 08 06 00 02 00 02 04 00 00 <Peer_Device_Address> 07 03
   ```
2. Set the advertising data.

   ```
   hcitool -i hci0 cmd 08 08 1F 00 99 88 77 66 55 44 33 22 11 00 99 88 77 66 55 44 33 22
    11 00 99 88 77 66 55 44 33 22 11 00
   ```
3. Enable advertising.

   ```
   hcitool -i hci0 cmd 08 0A 01
   ```

Step 4 – Enable host Sleep mode.

```
echo "psmode=1" > /proc/mbt_uart/hci0/config
```

Step 5 – Initiate the Bluetooth LE connection from the remote device.

```
hcitool -i hci1 lecc 00:50:43:21:30:CF
```

Where `00:50:43:21:30:CF` is the DUT address.

Expected result

The controller receives and accepts the Bluetooth LE connection request from the peer device (in the allowlist), and generates the interrupt on the configured GPIO pin.

8.4 Host wake-up on 88W8887A SD-UART using *hci_uart.ko* driver

The controller has an active Bluetooth LE connection and the host is in Sleep mode. The remote device sends a data packet to the controller. The controller generates an interrupt on the GPIO pin to wake up the Host.

Step 1 – Add the peer device to the DUT allowlist.

```
hcitool -i hci0 cmd 0x08 0x0011 00  F7 EE 6B 83 15 00
```

In the above command, `00` is the address type of the public device, and `F7 EE 6B 83 15 00` is the Bluetooth device address.

Step 2 – Configure the GPIO pin.

The firmware generates the interrupt on the configured GPIO pin.

```
hcitool  -i hci0 cmd 0x3F 0x53 0x04 0x00 0x01 0xFF
```

Step 3 – Start advertising on the DUT.

1. Set the advertising parameters.

   ```
   hcitool -i hci0 cmd 08 06 00 02 00 02 04 00  00 <Peer_Device_Address> 07 03
   ```

2. Set the advertising data.

   ```
   hcitool -i hci0 cmd 08 08 1F 00 99 88 77 66 55 44 33 22 11 00 99 88 77 66 55 44 33 22
    11 00 99 88 77 66 55 44 33 22 11 00
   ```

3. Enable advertising.

   ```
   hcitool -i hci0 cmd 08 0A 01
   ```

Step 3 – Initiate the Bluetooth LE connection from the remote device.

```
hcitool -i hci1 lecc 00:50:43:21:30:CF
```

In the above command, `00:50:43:21:30:CF` is the DUT address.

Step 4 – Enable the host Sleep mode.

```
echo "psmode=1" >/proc/mbt_uart/hci0/config
```

Step 5 – Send the Bluetooth LE data packet from the remote device to the DUT.

```
hcitool -i hci1 acldat -p dcd -d 0x5454 -c 1 -s 10 -P i 00:50:43:21:30:CF -H 128
```

In the above command, `00:50:43:21:30:CF` is the DUT address.

Expected results

The controller receives the data packet from the remote device and generates the interrupt on the configured GPIO pin.

9 Abbreviations

Table 18. Abbreviations

Abbreviation	Definition
ACK	acknowledgment
ACL	asynchronous connectionless link
BR	basic rate
CPU	central processor unit
DTB	device tree binary
DUT	device under test
EDR	enhanced date rate
GPIO	general purpose input/output
L2CAP	logical link control and adaptation protocol
LE	low energy
PCIe	peripheral component interconnect express
PSM	protocol service multiplexer
SDIO	secure digital input/output
SoC	system on chip
UART	universal asynchronous receiver/transmitter
UCD	unicast connectionless data
USB	universal serial bus
WLAN	wireless local area network

10 References

[1] DTB file (link)
[2] BTNXPUART driver code (*btnxpuart.c*) (link)
[3] bluez/bluetooth-next on GitHub – Kernel 5.7.0 commits disabling Bluetooth activities during Host suspend mode (link)
[4] Webpage – 88W8887: 1x1 Wi-Fi® 5 (802.11ac) + Bluetooth® Solution (link)
[5] Webpage – 88W8887 (Automotive): 2.4/5 GHz Dual-band 1x1 Wi-Fi® 5 (802.11ac) + Bluetooth® Solution (link)
[6] Webpage – 88W8897P (Automotive): 2.4/5 GHz Dual-band 2x2 Wi-Fi® 5 (802.11ac) + Bluetooth® Solution (link)
[7] Webpage – 88W8977: 2.4/5 GHz Dual-band 1x1 Wi-Fi® 4 (802.11n) + Bluetooth® Solution (link)
[8] Webpage – IW416: 2.4/5 GHz Dual-Band 1x1 Wi-Fi® 4 (802.11n) + Bluetooth® Solution (link)
[9] Webpage – 88W8987: 2.4/5 GHz Dual-Band 1x1 Wi-Fi® 5 (802.11ac) + Bluetooth® Solution (link)
[10] Webpage – 88W8997: 2.4/5 GHz Dual-Band 2x2 Wi-Fi® 5 (802.11ac) + Bluetooth® Solution (link)
[11] Webpage – 88Q9098: 2.4/5 GHz Dual-band 2x2 Wi-Fi® 6 (802.11ax) + Bluetooth® Automotive Solution (link)
[12] Webpage – 88W9098: 2.4/5 GHz Dual-band 2x2 Wi-Fi® 6 (802.11ax) + Bluetooth® (link)
[13] Webpage – AW590: Wi-Fi® 5 1x1 Concurrent Dual Wi-Fi (CDW) and Bluetooth® Combo SoC (link)
[14] Webpage – AW690: Wi-Fi® 6 1x1 Concurrent Dual Wi-Fi (CDW) and Bluetooth® Combo SoC (link)
[15] Webpage – IW610: 2.4/5GHz Dual-band 1x1 Wi-Fi® 6 + Bluetooth Low Energy + 802.15.4 Tri-Radio Solution link
[16] Webpage – AW611: 2.4/5 GHz Dual-band 1x1 Wi-Fi® 6 (802.11ax) + Bluetooth® Automotive Solution (link)
[17] Webpage – IW611: 2.4/5 GHz Dual-band 1x1 Wi-Fi® 6 (802.11ax) + Bluetooth® Solution (link)
[18] Webpage – IW612: 2.4/5 GHz Dual-band 1x1 Wi-Fi® 6 (802.11ax) + Bluetooth® + 802.15.4 Tri-radio Solution (link)
[19] Webpage – AW692: 2x2 Single-band (5 GHz) Concurrent Dual Wi-Fi® 6, 1x1 (2.4 GHz) Wi-Fi 6, and Bluetooth® Combo Solution (link)
[20] Webpage – AW693: 2x2 Dual-band (5-7 GHz), 1x1 (2.4 GHz) Concurrent Dual Wi-Fi 6/6E and Bluetooth Combo Solution (link)
[21] Webpage – IW693: 2x2 Dual-band (5-7 GHz), 1x1 (2.4 GHz) Concurrent Dual Wi-Fi 6/6E and Bluetooth Combo Solution (link)
[22] Webpage – IW623: 2x2 Tri-band (2.4G/5/6 GHz) Wi-Fi® 6E and Bluetooth Combo Solution (link)

11 Note about the source code in the document

The example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2018, 2020, 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

12 Revision history

Table 19. Revision history

Document ID	Release date	Description
AN12849 v.3.0	16 September 2025	• Section 1 "Introduction": updated. • Section 1.1 "Supported products": added. • Section *Related documentation*: removed. • Section *Overview*: split the content into Section 1 and Section 2. • Section 2.2 "GPIO pin polarity ": added. Content taken from Section 2.1. Removed the content related to SD-SD configuration. • Section 3 "DTB file configuration for BTNXPUART driver": added. • Section 4 "Host wake-up over Bluetooth": added a paragraph about the kernel version. • Section 5 "Host wake-up over Bluetooth LE": added a paragraph about the kernel version. • Section 5.1 "Host wake-up over Bluetooth LE connection": updated. • Section 5.2 "Host wake-up over Bluetooth LE scanning": updated. • Section 6 "Commands to configure the GPIO pins": updated. • Section 7 "Command to configure host Sleep mode": updated. • Section 8.1 "Host wake-up over Bluetooth LE connection on AW692 / AW693 PCIE-UART using BTNXPUART": added. • Section 8.2 "Host wake up over Bluetooth LE scanning IW611/IW612 SD-UART using BTNXPUART": added. • Removed the section *Host wake-up over Bluetooth LE scanning on 88 Q9098/88W9098 SD-SD*. • Section 10 "References ": added. • Section 11 "Note about the source code in the document": added.
AN12489 v.2.0	12 June 2020	• Applied NXP branding and revision numbering scheme. • Extended the document scope to 88Q9098/88W9098 Wi-Fi and Bluetooth Combo SoC in section *Related documentation*, Section 2.3, Section 6, and Section 7 • Added Section 8.3 and section *Host wake-up over Bluetooth on 88 W9088/88Q9098 SD-SD*.
AN12489 v.1.0	28 November 2018	Initial version

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

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.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. 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 B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Bluetooth — the Bluetooth wordmark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by NXP Semiconductors is under license.

Tables

Figures

Contents