AN4820
Application note

BlueNRG-1 low power modes

Introduction

The BlueNRG-1 is a very low power Bluetooth low energy (BLE) single-mode system-on-chip, compliant with Bluetooth® specification. The architecture core is a Cortex-M0 32-bit.This application note explains the low power modes for the BlueNRG-1 device.

Contents

1 BlueNRG-1 HW low power modes

Three low power modes are provided from the BlueNRG-1 HW to achieve the best compromise between low power consumption, short startup time and available wakeup sources:

- CPU-Halt mode
  - In this mode only the CPU is stopped. All device peripherals continue to operate and they can wake up the CPU when an interrupt/event occurs. This is the lowest power save mode.
- Sleep mode
  - In this mode the CPU is stopped and all the peripherals are disabled. Only the low speed oscillator block and the external wakeup source block are running.
  - The wakeup source are the *wakeup timer* or the *IO9*, *IO10*, *IO11*, *IO12* and *IO13*.
  - When the wakeup is triggered by a previous listed sources, the system reverts to the run mode with all the peripherals on. Exiting from the sleep mode, the application needs to wait until the high speed oscillator is stable.
- Standby mode
  - In this mode the CPU is stopped and all the peripherals are disabled. The only wakeup source are *IO9*, *IO10*, *IO11*, *IO12* and *IO13*.
  - Exiting from the standby mode, the application needs to wait until both the high speed and low speed oscillator are stable.
  - This mode is the highest power save mode

The current consumptions for all the low power modes are listed on the BlueNRG-1 datasheet.

2 BlueNRG-1 low power modes support

The BlueNRG-1 DK software package provides a software that supports all the BlueNRG-1 HW low power modes.

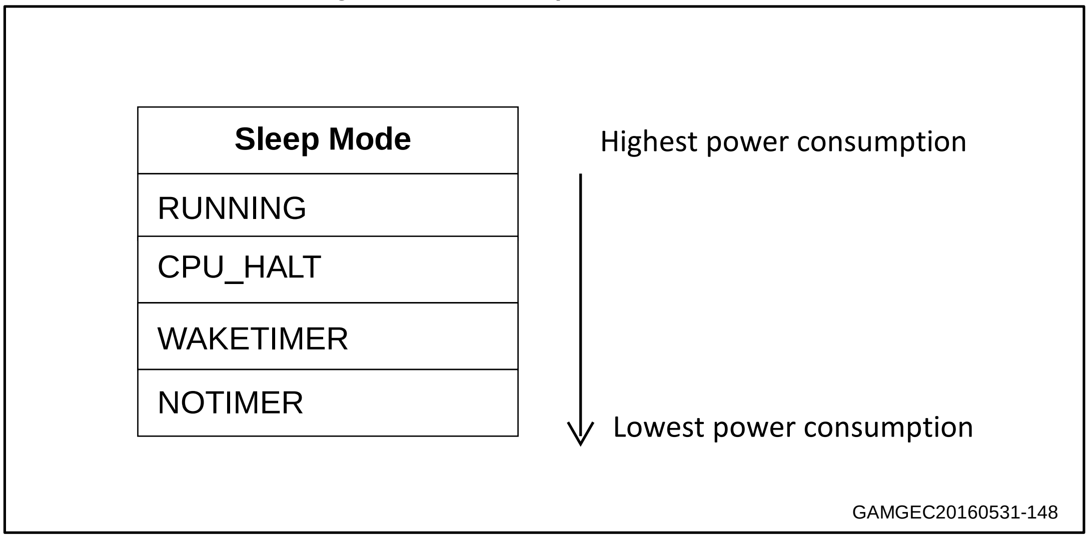
The low power software combines the low power requests coming from the application with the radio operating mode, choosing the best low power mode applicable in the current scenario. This negotiation between the radio module and the application requests is done to avoid losing of data exchanged over the air. This negotiation is in charge of the low power software.

When the BlueNRG-1 exits from any low power mode a reset occurs. This reset causes that all the peripherals configuration and the application context are lost. The low power software implements a mechanism to save and restore all the peripherals configuration and the application context when a power save procedure is called. So, from application point of view, the exit from a low power procedure is fully transparent, that means when a wakeup from low power occurs, the CPU returns to execute the next instruction after the low power function call.

The low power software implements the following power save modes:

- **SLEEPMODE_RUNNING**
  - In this low power mode, everything is active and running. In practice this mode is not used, but it is defined for completeness of information: it’s not a real power save mode.
- **SLEEPMODE_CPU_HALT**
  - This power save mode implements the *HW CPU-Halt* low power mode.
- **SLEEPMODE_WAKETIMER**
  - This power save mode implements the *HW Sleep* low power mode.
- **SLEEPMODE_NOTIMER**
  - This power save mode implements the *HW Standby* low power mode.

Figure 1: BlueNRG-1 power save modes



To enable any low power mode the application shall call the function “BlueNRG_Sleep()”.

```
uint8_t BlueNRG_Sleep(SleepModes sleepMode,
                      uint8_t gpioWakeBitMask,
                      uint8_t gpioWakeLevelMask,
                      uint32_t sleep_time);
```

Where:

- **sleepMode** is the low power mode to enable:
  - SLEEPMODE_RUNNING
  - SLEEPMODE_CPU_HALT
  - SLEEPMODE_WAKETIMER
  - SLEEPMODE_NOTIMER
- **gpioWakeBitMask** is a bit mask of the IOs that can wake the chip from low power mode:
  - WAKEUP_IO9
  - WAKEUP_IO10
  - WAKEUP_IO11
  - WAKEUP_IO12
  - WAKEUP_IO13

    If this field is zero the **gpioWakeLevelMask** parameter is ignored.
- **gpioWakeLevelMask** is a bit mask used to setup the active wakeup level for each wakeup source:
  - WAKEUP_IO_LOW, the system wakes up when IO is low
  - WAKEUP_IO_HIGH, the system wakes up when IO is high
- **sleep_time** is the deep sleep timeout, expressed in milliseconds. The wakeup sources can be also the IOs. If this timeout is equal to zero, the power save software sets automatically a timeout according the current radio operating mode.

The function returns the status:

- **BLUENRG_SLEEP_CONFIGURATION_ERROR**
- **SUCCESS**

The low power software exports other functions useful for the application:

- **BlueNRG_WakeupSource.**
  - This function returns the last wakeup source from the low power mode.
- **BlueNRG_IdleSleep.**
  - This function executes a wait for interrupt instruction “WFI” and suspends the core execution until one interrupt occurs. This is another way to put the BlueNRG-1 in *HW CPU-Halt* low power mode.
- **App_SleepMode_Check.**
  - This function allows the application to setup a different low power mode in special cases. This means that the application can modify the low power mode set in the main loop, using the function **App_SleepMode_Check**(). When the **BlueNRG_Sleep**() function is called, the Low Power software negotiates the sleep mode to be applied according to the radio state machines, the low power mode requested from the application in the main loop, and the low power mode requested from the **App_SleepMode_Check**().
  - For example, let’s assume there is an application that sets the sleep mode SLEEPMODE_WAKETIMER in the main loop. An interrupt signals to the application that a sensor is ready to provide some measurements. To avoid to enter in sleep mode, the application can set the **App_SleepMode_Check**() to SLEEPMODE_RUNNING, so, when the main loop executes the

BlueNRG_Sleep(), this function negotiates the low power mode for applying the SLEEPMODE_RUNNING one.

3 BlueNRG-1 low power modes examples

Some low power modes application use cases are provided in the following sections.

3.1 Low power SLEEPMODE_RUNNING

In this low power mode, everything is active and running: it is not a real low power mode. A typical source code is:

```
 while(1) /*main loop*/
{
 /* BLE Stack Tick */
 BTLE_StackTick();
 /* Application Tick */
 APP_Tick();
 /* Power Save management */
 BlueNRG_Sleep(SLEEPMODE_RUNNING, 0, 0, 0);
}
```

This low power mode does not need wakeup sources.

3.2 Low power SLEEPMODE_CPU_HALT

In this mode only the CPU is stopped. All peripherals continue to operate and can wake up the CPU when an interrupt occurs. A typical source code is:

```
while(1) /*main loop*/
{
 /* BLE Stack Tick */
 BTLE_StackTick();
 /* Application Tick */
 APP_Tick();
 /* Power Save management */
 BlueNRG_Sleep(SLEEPMODE_CPU_HALT,
               WAKEUP_IO13| WAKEUP_IO11,
               (WAKEUP_IO_LOW<< WAKEUP_IO11)|(WAKEUP_IO_LOW<< WAKEUP_IO13),
               0);
 }
```

The wakeup sources are IO13 and IO11 they are sensitive to the low level. All the peripheral interrupt can also wake-up the BlueNRG-1.

If the radio operating mode doesn’t allow this low power request, because it was executing a non-stoppable operation, this low power mode is converted automatically inside the low power software in SLEEPMODE_RUNNING.

3.3 Low power SLEEPMODE_WAKETIMER

In this mode the CPU is stopped and all the peripherals are disabled. Only the low speed oscillator block and the external wakeup source block are running.

A typical source code is:

```
while(1) /*main loop*/
 {
   /* BLE Stack Tick */
   BTLE_StackTick();
   /* Application Tick */
   APP_Tick();
   /* Power Save management */
   BlueNRG_Sleep(SLEEPMODE_WAKETIMER,
                 WAKEUP_IO13,
                 WAKEUP_IO_LOW<< WAKEUP_IO13,
                 2000);
 }
```

The wakeup sources are both the IO13 with low level sensitive and a timeout of 2 seconds.

In this scenario, the application wants to enable the low power mode *WAKETIMER*. If the radio operating mode is in connection state or advertising state, the stack accepts the low power mode proposed by the application, but, if necessary, the timeout is changed, inside the low power software, to follow the connection interval time profile or the advertising interval time profile.

3.4 Low power SLEEPMODE_NOTIMER

In this low power mode the CPU is stopped and all the peripherals are disabled. Only the external wakeup source block is running. A typical source code is:

```
while(1) /*main loop*/
{
 /* BLE Stack Tick */
 BTLE_StackTick();
 /* Application Tick */
 APP_Tick();
 /* Power Save management */
 BlueNRG_Sleep(SLEEPMODE_NOTIMER,
               WAKEUP_IO13,
               WAKEUP_IO_HIGH << WAKEUP_IO13,
               0);
 }
```

The wakeup source is the IO13 with high level sensitive.

In this scenario the application wants to enable the low power mode *NOTIMER*. If the radio module is in connection state, after the negotiation with the radio stack, the low power software changes the low power mode in *WAKETIMER* to follow the connection time profile. Instead if the radio module is in an idle state, the radio stack accepts the low power mode *NOTIMER* requested from the application, and the low power software doesn't change it.

4 List of acronyms

Table 1: List of acronyms used in the document

Term	Meaning
BLE	Bluetooth low energy
HW	Hardware

5 Revision history

Table 2: Document revision history

Date	Version	Changes
29-Jun-2016	1	Initial release.