# AN14852

## i.MX RT700 Deep Sleep and Wake-up Time

Rev. 1.0 — 10 November 2025 Application note

**Document information**

| Information | Content |
|---|---|
| Keywords | AN14852, i.MX RT700, Deep Sleep Mode, Async Deep Sleep, Low Power MCU, Wake-up Latency, Power Management, NXP RT700 SDK, SRAM Retention, PMC Configuration, PCA9422 PMIC |
| Abstract | This application note provides a comprehensive overview of the Deep Sleep mode on i.MX RT700 devices. |

# 1 Introduction

The i.MX RT700 features a segmented power architecture with nine independent power domains and multiple voltage rails. This design enables fine-grained control over power distribution, allowing domains to be selectively placed in low-power states based on application demands. The device includes separate Compute and Sense domains, each capable of independently entering Sleep or Deep Sleep modes. Additionally, modules within the Media and Common domains operate on a higher voltage rail (VDDN), while other domains support lower voltage operation, enabling further power optimization.

This application note provides a comprehensive overview of the Deep Sleep mode on i.MX RT700 devices. It explains the underlying hardware mechanisms, configuration steps, wake-up sources, and best practices for achieving optimal power savings.

# 2 Deep Sleep mode and Async Deep Sleep

The i.MX RT700 MCU supports two advanced Deep Sleep modes that significantly reduce power consumption while maintaining selective wake-up capabilities:

- Deep Sleep Mode
- Async Deep Sleep mode

## 2.1 Deep Sleep mode

Deep Sleep mode on NXP i.MX RT700 microcontrollers is a low-power operational state designed to minimize energy consumption while retaining critical system context. In this mode, the Compute Domain Core and/or Sense Domain Core and most high-speed peripherals are powered down, while select domains remain active or retained. The system clock can optionally remain active depending on the configuration, allowing for faster recovery and continued operation of select peripherals. In this mode, the device supports multiple wake-up sources, which can be used to resume normal execution based on application requirements. Transitioning into Deep Sleep involves configuring the domains to safely shut down unused domains and define wake-up conditions.

## 2.2 Deep Sleep Async mode

Asynchronous Deep Sleep mode is a specialized low-power state that achieves maximum energy savings by placing both the compute and sense domains into Deep Sleep simultaneously. In this mode, all system clocks, including core, bus, and peripheral clocks, are completely disabled, with the sole exception of the RTC oscillator located in the always-on (VDD1V8_AON) domain. This configuration ensures that no synchronous logic remains active, allowing the device to reach its lowest possible power consumption while maintaining minimal timekeeping functionality.

## 2.3 Comparing Deep Sleep vs Deep Sleep Async

Table 1 highlights the primary distinctions between Deep Sleep and Asynchronous Deep Sleep modes. For a comprehensive list of wake-up sources applicable to each mode, users can refer to the “Wake-up Sources for Different Power Modes” table located in the Power Overview section of the i.MX i.MX RT700 Reference Manual to obtain a complete list of supported wake-up sources. This includes both asynchronous sources such as RTC alarms and GPIO edge detection and non-asynchronous sources that require active clock domains.

**Table 1. Deep Sleep and Async Deep Sleep comparison**

| Feature | Deep Sleep | Async Deep Sleep |
|---|---|---|
| Domains Powered Down | Compute and Sense domain can remain active or shutdown independently | Both Compute and Sense Domains required to shut down |
| Clock sources active | A selectable clock can remain active | Only RTC oscillator in VDD1V8_AON remains active |
| Wake-up Sources | Synchronous and Asynchronous interrupts are available (RTC, GPIO, timers, serial interface, and other peripherals) | Only Asynchronous interrupts |
| Peripheral Clock Availability | A selectable clock can remain active | All peripheral clocks are disabled |
| SRAM Retention | Configurable per memory bank | Configurable per memory bank |
| Wake-up Latency | Low | Slightly higher due to full clock reinitialization |

## 2.4 Key features and benefits

Deep Sleep mode allows selective clock gating and power domain control, enabling low-power operation while retaining access to key peripherals such as ADCs, communication interfaces, and timers. This mode is ideal for applications that require periodic activity or low-latency wake-up, as it supports a broad range of wake-up sources and retains SRAM and register states for fast recovery.

In contrast, Asynchronous Deep Sleep mode achieves the lowest possible power consumption by disabling all clocks except the RTC oscillator in the Always-On domain. This mode is suited for scenarios where the system remains idle for extended periods and only requires minimal wake-up functionality, such as from an RTC alarm or GPIO edge detection. While it offers greater power savings, it also introduces longer wake-up latency and limits peripheral availability.

## 2.5 SLEEPCON and PMC aggregation

The ability to transition the Compute and Sense domains into Deep Sleep mode independently requires precise configuration of the SLEEPCON and PMC peripherals. Understanding the interaction between these components is critical to ensure correct entry and exit sequences for Deep Sleep.

The SLEEPCFG and RUNCFG register banks within SLEEPCON0 and SLEEPCON1 control power settings based on the power state of the CPU. SLEEPCFG applies when the CPU enters Deep Sleep or deeper low-power modes, while RUNCFG is used during Active or standard Sleep modes. Software must configure SLEEPCON before entering Deep Sleep. Once SLEEPCON completes its setup, control passes to the specific PMC instance, PMC0 for CPU0 and PMC1 for CPU1, which also has two register banks: PDRUNCFG0 to PDRUNCFG5 when the CPU is in active or sleep. And PDSLEEPCFG0 to PDSLEEPCFG5 for lower power states. If there are conflicting settings between domains, PMC ensures that the higher power mode takes precedence via the aggregation mechanism.

This staged aggregation ensures a controlled and efficient power-up sequence, minimizing inrush current and allowing selective domain activation based on application needs.

## 2.6 Voltage setting considerations

The power architecture supports lower voltage operation in Deep Sleep and Deep Sleep Async modes, allowing for reduced power consumption. However, users must ensure that all voltage levels are configured appropriately to maintain system stability and meet the minimum requirements for each mode.

In Deep Sleep mode, the minimum voltage for VDD1 and VDD2 is 0.63 V, while in Deep Sleep Async mode, it is 0.5 V. VDDN requires a minimum of 1.0 V in Active and standard sleep modes, and 0.6 V in Dual Deep Sleep Async or FDSR modes, with the condition that VDDN must always be equal to or greater than VDD1 and VDD2. For more reference and detailed specifications, see the i.MX RT700 Datasheet.

**Table 2. Minimum voltage requirements for Deep Sleep modes**

| Symbol | Description | Min. | Type | Max. | Unit | Condition |
|---|---|---|---|---|---|---|
| VDD1 | Supply voltage for core logic in the Sense domain | 0.63 | — | 1.155 | V | Deep Sleep mode |
| VDD1 | Supply voltage for core logic in the Sense domain | 0.5 | — | 1.155 | V | Deep Sleep Async mode |
| VDD2 | Supply voltage for core logic in the Main compute domain | 0.63 | — | 1.155 | V | Deep Sleep mode |
| VDD2 | Supply voltage for core logic in the Sense domain | 0.5 | — | 1.155 | V | Deep Sleep Async mode |
| VDDN | Supply voltage for media and common peripherals[1] | 1.0 | — | 1.155 | V | Active mode or Sleep mode or Deep Sleep or Compute Deep Sleep<br>Async: where VDDN = 1.0 V min, VDD1 and VDD2 can be 1.0 V, 0.9 V, 0.8 V, 0.7 V for external supply. |
| VDDN | Supply voltage for media and common peripherals[1] | 0.6 | — | 1.155 | V | Dual Deep Sleep Async or FDSR |

[1] VDDN voltage must be equal to or greater than VDD1 and VDD2.

## 2.7 Software demo and measurements

The following section uses SDK examples (SDK version 25.06.00) to validate i.MX RT700 behavior and measure power consumption across critical power rails during Deep Sleep and Deep Sleep Async modes. Two demonstration applications are used:

1. **Power Modes Switch Compute Only Demo**: This example demonstrates the configuration required to enter Deep Sleep Async mode, enabling minimal power consumption by isolating the Compute domain while Sense domain enters deep sleep by default.
2. **Power Mode Switch Demo**: This example showcases independent power mode control for both the Compute and Sense domains, including transitions into Deep Sleep and other low-power states. It is used to evaluate Deep Sleep mode behavior and domain-level power management.

Each SDK demo example for the i.MX RT700 is structured as a dual-core application composed of two projects:

- **Primary Project:** Contains the application code for CPU0 (Compute Domain).
- **Secondary Project:** Contains the application code for CPU1 (Sense Domain).

When building the primary project, the build system automatically compiles the secondary project to ensure that both binaries are synchronized. The secondary image is embedded into the primary image and loaded together onto the device. Within the primary application, dedicated boot routines are responsible for initializing and launching the secondary core. This architecture enables concurrent execution and debugging of both domains, allowing developers to validate inter-core interactions and system-level behavior during runtime.

Similarly, when using the MIMXRT700-EVK on-board debugger, two virtual COM ports enumerate, each corresponding to one of the MCU's cores. One port is associated with CPU0 (Compute Domain), and the other with CPU1 (Sense Domain). During execution, each core outputs diagnostic messages to its respective COM

port. These messages typically include the CPU identifier, allowing developers to distinguish between the two ports and monitor activity independently.

By default, SDK examples operate in Power Supply Mixed mode, using both internal LDOs and PMICs to supply specific power rails. For this evaluation, the system is configured to operate in Power Supply PMIC mode, where all power rails are supplied exclusively by the external PMIC. This configuration enhances flexibility in managing power delivery across the i.MX RT700 platform, allowing developers to tailor voltage levels and enablement states according to application-specific requirements.

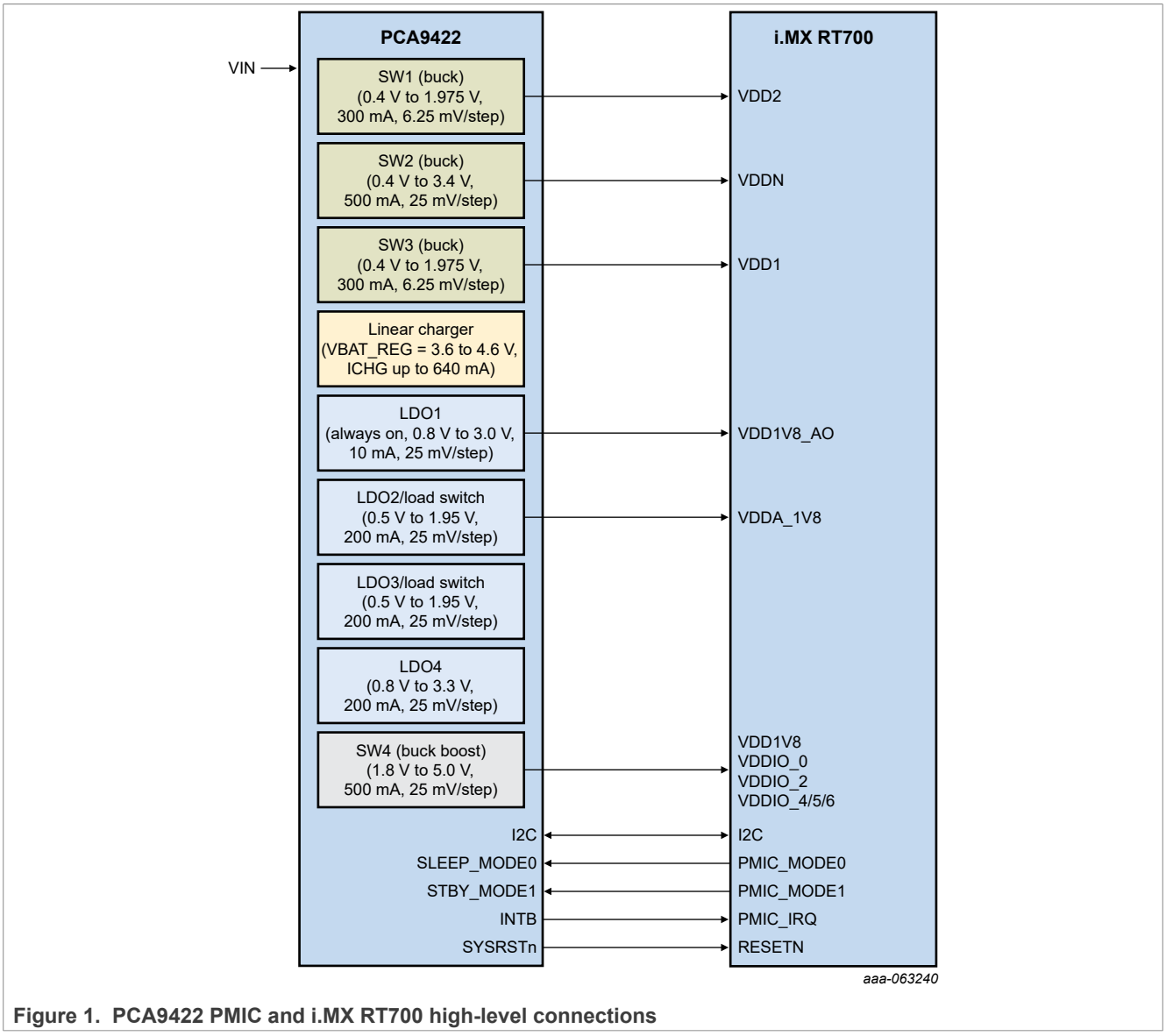



**Figure 1. PCA9422 PMIC and i.MX RT700 high-level connections**

In both demos, the Power Supply option is controlled by a macro in *power_demo_config.h*. Change the DEMO_POWER_SUPPLY_OPTION for DEMO_POWER_SUPPLY_PMIC in both Compute (primary) and Sense (secondary) projects.

```
/*******************************************************************************
 * Definitions
 ******************************************************************************/
/* Define the power supply mode on the board.
 *   - DEMO_POWER_SUPPLY_MIXED, the VDDN is supplied by external PMIC. VDD1 and VDD2 are supplied by internal LDOs.
 *   - DEMO_POWER_SUPPLY_PMIC, the VDDN, VDD1 and VDD2 are supplied by PMIC.
 */
#define DEMO_POWER_SUPPLY_OPTION DEMO_POWER_SUPPLY_PMIC

#define DEMO_POWER_SUPPLY_PMIC  2U
#define DEMO_POWER_SUPPLY_MIXED 3U
```

**Figure 2. Power supply options in SDK code**

For these tests, the power consumption is measured across three key rails on the MIMXRT700-EVK: VDD2 (Compute Domain), VDD1 (Sense Domain), and VDDN (Media and Common Peripherals). Current measurement devices are placed on the designated jumper locations, as shown in Figure 3.

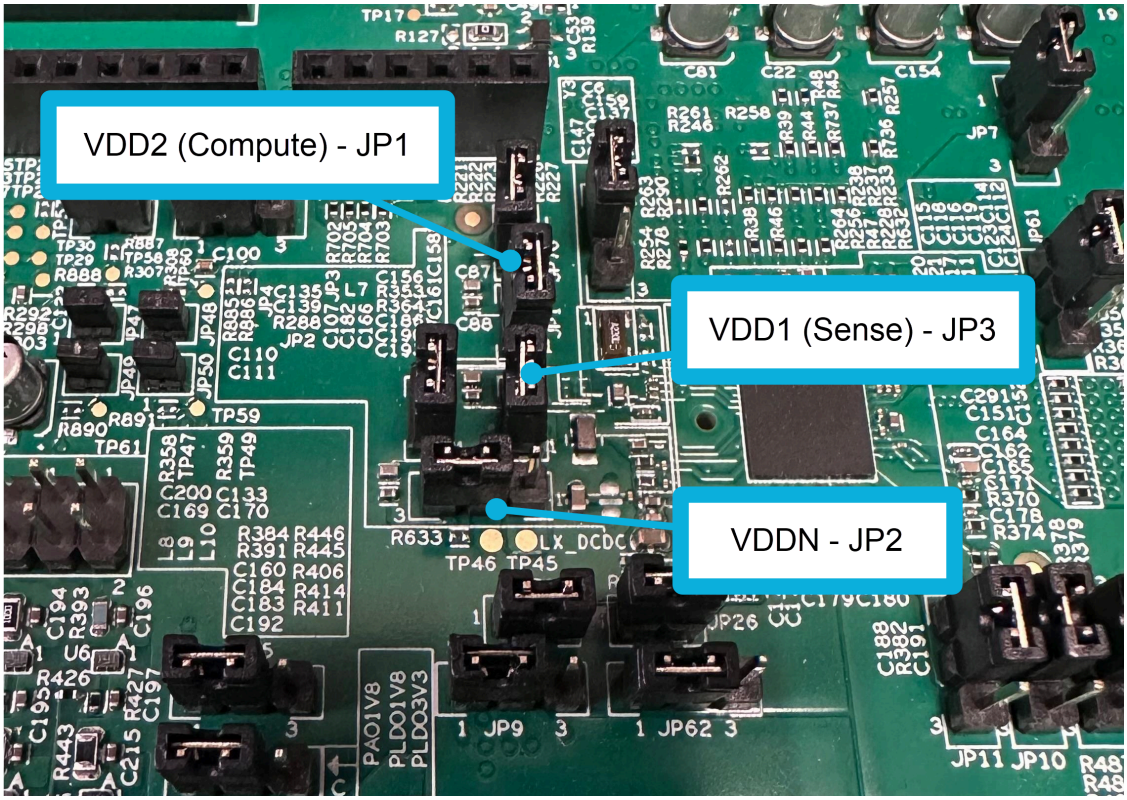



**Figure 3. Jumper location of power supply rails**

Table 3 describes the settings of each of the jumpers indicated above:

**Table 3. Jumper settings**

| Part identifier | Jumper type | Description | Jumper settings |
|---|---|---|---|
| JP2 | 1 × 3 pin header | Current test point jumper to measure VDDN current | • 1-2 shorted: In this setting, the on-chip DCDC is used for VDDN power supply.<br>• 2-3 shorted (default setting): In this setting, the external PMIC is used for VDDN power supply. |

**Table 3. Jumper settings*...continued***

| Part identifier | Jumper type | Description | Jumper settings |
|---|---|---|---|
| JP3 | 1 × 2 pin header | Current test point jumper to measure VDD1 current | • Open: In this setting, the internal LDO is used for VDD1 power supply.<br>• Shorted (default setting): In this setting, the external PMIC PCA9422 s used for VDD1 power supply. |
| JP1 | 1 × 2 pin header | Current test point jumper to measure VDD2 current | • Open: In this setting, the internal LDO is used for VDD2 power supply.<br>• Shorted (default setting): In this setting, the external PMIC PCA9422 s used for VDD2 power supply. |

## 2.8 Deep Sleep Async mode testing

Run the `power_mode_switch_comp_only` SDK example, and this message shows as below:

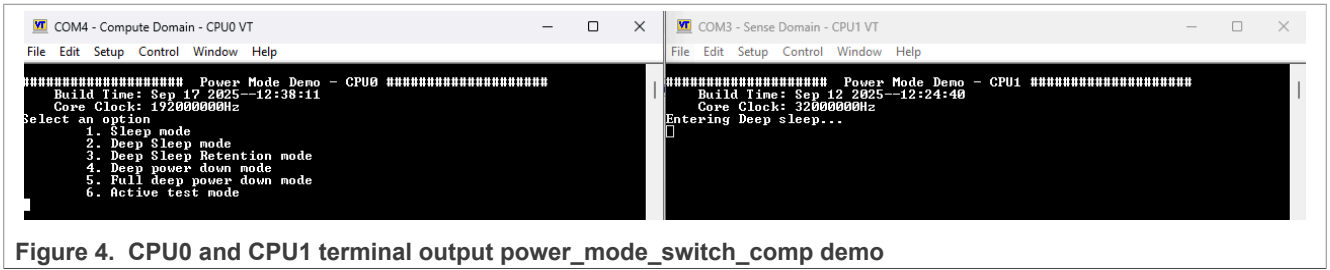



**Figure 4. CPU0 and CPU1 terminal output power_mode_switch_comp demo**

When running the `power_mode_switch_comp_only` SDK example, the system outputs a message indicating that CPU1 (Sense Domain) boots and immediately enters Deep Sleep mode. Similarly, CPU0 (Compute Domain) remains active, allowing the user to select between various low power modes or continue in active mode. In this specific case, the measured power consumption corresponds to CPU0 entering Deep Sleep mode, which aligns with the operational characteristics of the Deep Sleep Async mode.

It is important to note the voltage configuration for each power rail during this operation, as it directly impacts the overall power profile. Proper voltage scaling on VDD1 and VDD2 is essential to achieve the expected power savings in Deep Sleep Async mode.

**Table 4. Measurement results power_mode_switch_comp demo out-of-the-box**

| Rail | Current (µA) | Voltage (mV) | Power (µW) |
|---|---|---|---|
| VDD2 - Compute | 29.42 | 503.5 | 14.81 |
| VDD1 - Sense | 9.3 | 505.2 | 4.7 |
| VDDN - Media and common | 1.45 | 599.4 | 0.87 |
| TOTAL | | | **20.38** |

To demonstrate the impact of peripheral shutdown on power consumption, we perform a simple exercise using the `PDSLEEPCFG` register configuration. This example focuses on adjusting the number of SRAM partitions that remain powered during Deep Sleep mode. By selectively disabling unused SRAM blocks, we can observe measurable reductions in current draw, providing insight into how memory retention settings influence overall power efficiency.

In this demo, the configuration of SRAM retention during Deep Sleep mode is controlled by the macro `APP_DEEPSLEEP_RAM_APD`. According to the i.MX RT700 Reference Manual, this macro maps to the `PDSLEEPCFG2` register, which determines whether the RAM Partitions Array remains powered or is shut down during Deep Sleep.

### PDSLEEPCFG2 register

| Register | Offset |
|---|---|
| PDSLEEPCFG2 | C0h |

| Bits | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 1 | | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM |
| W | | | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bits | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM | SRAM |
| W | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

| Field | Function |
|---|---|
| 29-0<br>SRAMn | RAM Partition n Array Power Down<br>This field has two copies separately accessed by the Compute and Sense Slots.<br>0b: Powers on RAM Partition 0 array<br>1b: Powers down RAM Partition 0 array and periphery |

For this exercise, we modify the `PDSLEEPCFG2` value to reduce the number of retained SRAM partitions. Specifically, the configuration is changed from `0x3FFFFU`, which retains partitions P0 to P17, to `0x3FFU`, retaining only partitions P0 to P9. This adjustment powers down both the periphery and array of partitions P10 to P17, leaving only 1 MB of SRAM retained (P0 to P9, with array ON and periphery OFF). This change allows us to observe the impact of reduced memory retention on overall power consumption during Deep Sleep mode.

**Table 5. Measurement results power_mode_switch_comp after PDSLEEPCFG2 change**

| Rail | Current (µA) | Voltage (mV) | Power (µW) |
|---|---|---|---|
| VDD2 - Compute | 17.97 | 503.6 | 9.05 |
| VDD1 - Sense | 9.30 | 505.2 | 4.70 |
| VDDN - Media and common | 1.45 | 599.4 | 0.87 |
| TOTAL | | | 14.62 |

As demonstrated in this example, using the `PDSLEEPCFG` register to control peripheral behavior during Deep Sleep mode enables measurable reductions in power consumption. This configuration illustrates how targeted adjustments to power domain settings can optimize energy efficiency without compromising essential functionality.

## 2.9 Deep Sleep mode testing

Run the `power_mode_switch` SDK example, and this message shows as below:

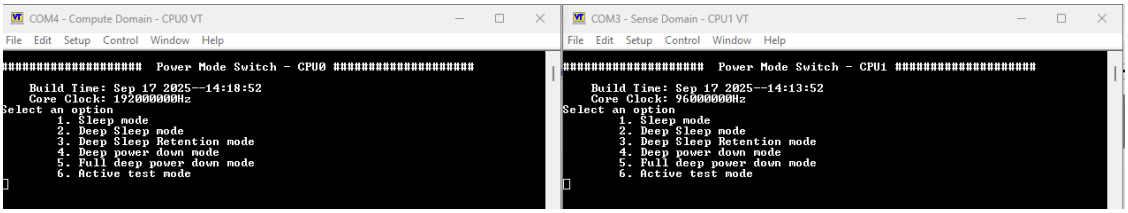



**Figure 5. CPU0 and CPU1 terminal output power_mode_switch demo**

For this test scenario, we focus on a common use case where CPU0 enters Deep Sleep mode while CPU1 remains active. Similarly to other dual-core examples, users can select this configuration using the interactive menu provided by the demo.

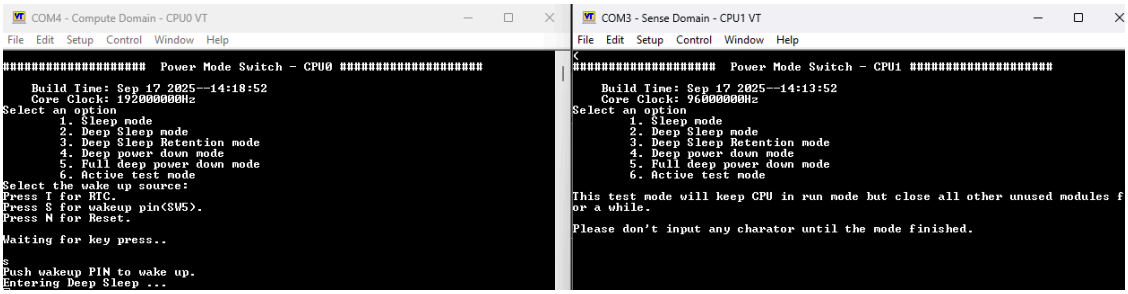



**Figure 6. CPU0 and CPU1 terminal output with use case selection**

To achieve the lowest possible power consumption during Deep Sleep mode, we adjust the voltage levels of the power rails to the minimum values permitted by the mode. These adjustments are guided by the specifications outlined in Section 2.6.

The following sections of the demo code are modified to reflect the updated voltage configuration.

- **VDD2 voltage:** This is controlled by macro `DEMO_LOW_POWER_RUN_VOLT` in *hardware_init.c* file.

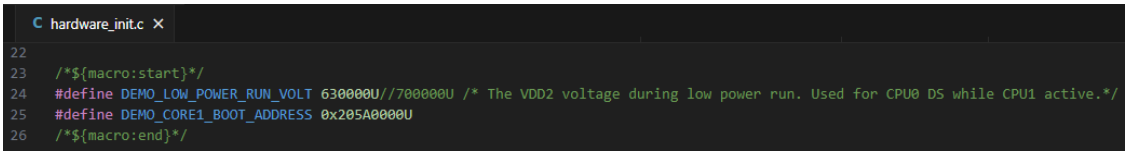
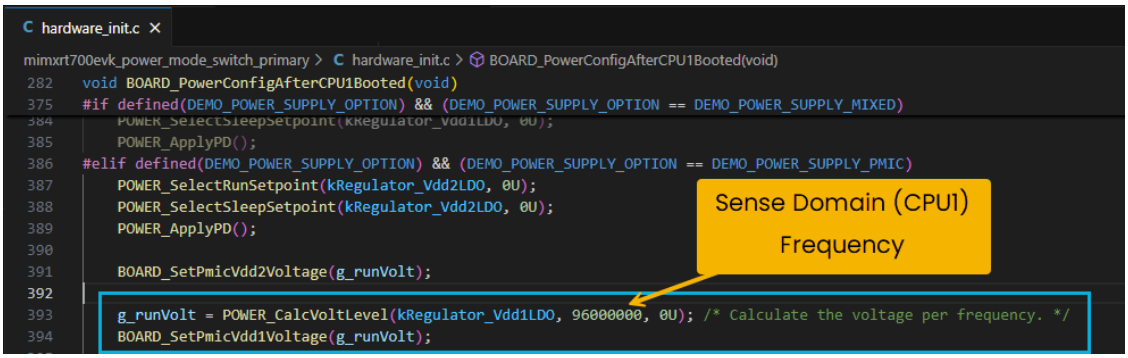



**Figure 7. Code change VDD2**

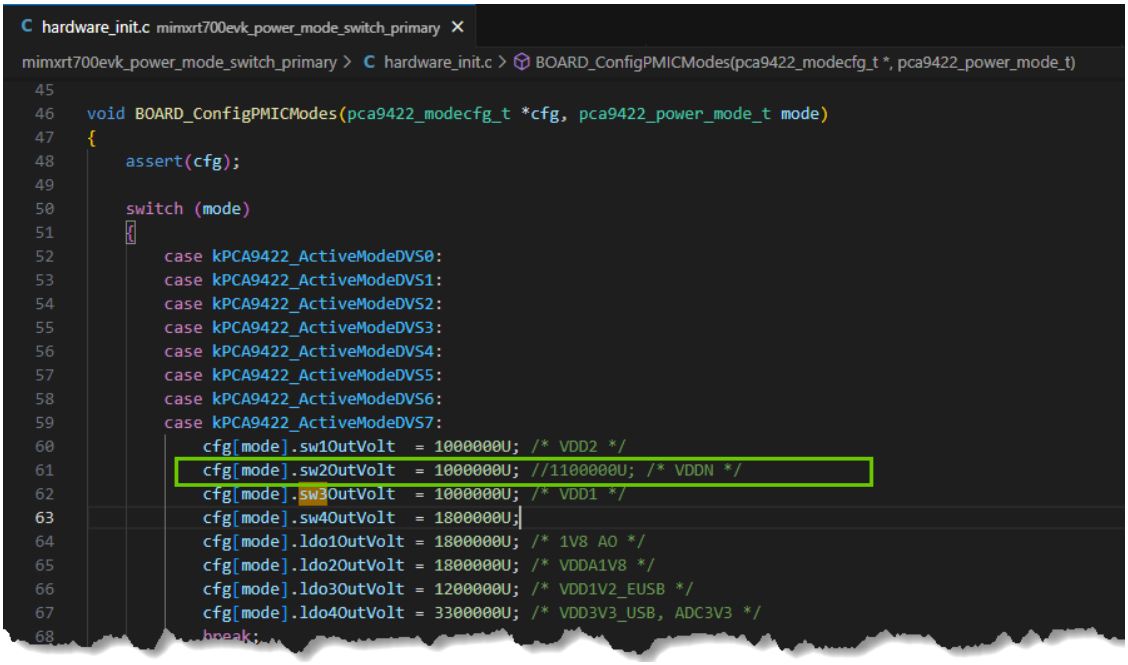
- **VDD1 voltage:** This can be changed using the `POWER_CalcVoltLevel` function, which adjusts PMIC output voltage to the appropriate level based on the core frequency.




```
282  void BOARD_PowerConfigAfterCPU1Booted(void)
375  #if defined(DEMO_POWER_SUPPLY_OPTION) && (DEMO_POWER_SUPPLY_OPTION == DEMO_POWER_SUPPLY_MIXED)
385      POWER_ApplyPD();
386  #elif defined(DEMO_POWER_SUPPLY_OPTION) && (DEMO_POWER_SUPPLY_OPTION == DEMO_POWER_SUPPLY_PMIC)
387      POWER_SelectRunSetpoint(kRegulator_Vdd2LDO, 0U);
388      POWER_SelectSleepSetpoint(kRegulator_Vdd2LDO, 0U);
389      POWER_ApplyPD();
390
391      BOARD_SetPmicVdd2Voltage(g_runVolt);
392
393      g_runVolt = POWER_CalcVoltLevel(kRegulator_Vdd1LDO, 96000000, 0U); /* Calculate the voltage per frequency. */
394      BOARD_SetPmicVdd1Voltage(g_runVolt);
```

**Figure 8. Code change VDD1**

- **VDDN voltage:** This can be changed by changing the initial PMIC configuration in `BOARD_ConfigPMICModes` function.




```
45
46   void BOARD_ConfigPMICModes(pca9422_modecfg_t *cfg, pca9422_power_mode_t mode)
47   {
48       assert(cfg);
49
50       switch (mode)
51       {
52           case kPCA9422_ActiveModeDVS0:
53           case kPCA9422_ActiveModeDVS1:
54           case kPCA9422_ActiveModeDVS2:
55           case kPCA9422_ActiveModeDVS3:
56           case kPCA9422_ActiveModeDVS4:
57           case kPCA9422_ActiveModeDVS5:
58           case kPCA9422_ActiveModeDVS6:
59           case kPCA9422_ActiveModeDVS7:
60               cfg[mode].sw1OutVolt  = 1000000U; /* VDD2 */
61               cfg[mode].sw2OutVolt  = 1000000U; //1100000U; /* VDDN */
62               cfg[mode].sw3OutVolt  = 1000000U; /* VDD1 */
63               cfg[mode].sw4OutVolt  = 1800000U;
64               cfg[mode].ldo1OutVolt = 1800000U; /* 1V8 AO */
65               cfg[mode].ldo2OutVolt = 1800000U; /* VDDA1V8 */
66               cfg[mode].ldo3OutVolt = 1200000U; /* VDD1V2_EUSB */
67               cfg[mode].ldo4OutVolt = 3300000U; /* VDD3V3_USB, ADC3V3 */
68               break;
```

**Figure 9. Code change VDDN**

Run this use case, and these power numbers are delivered:

**Table 6. Measurement results power_mode_switch after voltages change**

| Rail | Current (µA) | Voltage (mV) | Power (µW) |
|---|---|---|---|
| VDD2 - Compute | 60.28 | 631.4 | 38.07 |
| VDD1 - Sense | 3510 | 800.0 | 2810 |
| VDDN - Media and common | 13.43 | 999.8 | 13.43 |
| TOTAL | | | 2861 (28.61 mW) |

This example demonstrates how to achieve minimal power consumption in a scenario where CPU0 enters Deep Sleep mode while CPU1 remains active. By reducing the voltage rails to the lowest levels permitted by Deep Sleep mode, the system operates in a highly energy-efficient state. In practical applications, developers must consider system-level constraints and performance requirements to determine the appropriate configuration.

This includes adjusting voltage levels, selecting which SRAM partitions to retain, and managing peripheral availability during low power operation to balance power savings with functional needs.

## 3 Wake-up time considerations

In IoT applications with strict power constraints, one critical factor that developers must consider is the wake-up latency from low power modes to active operation. Since the device typically wakes periodically to perform tasks before returning to a low power state, the time required to resume full functionality directly affects the balance between performance and energy efficiency. Minimizing wake-up time ensures that the system can complete operations quickly and return to a low-power state, helping maintain optimal power budgets without compromising responsiveness.

The i.MX RT700 supports a feature known as FRO Fast Wake-Up, which enables rapid oscillator startup during transitions from low power modes. This feature is controlled via the FROx_CNFG1[FSTUPEN] register bit. When enabled, the FRO outputs a divide-by-2 clock during the initial startup phase, allowing the system to resume operation quickly. Once the oscillator frequency stabilizes within 5 % of the target, the FRO automatically switches to the undivided clock output. Throughout this ramp-up period, the amplitude and duty cycle remain stable, ensuring reliable clock behavior. All three FROs in the i.MX RT700 support this fast wake-up mechanism.

In addition to enabling FRO Fast Wake-Up, developers must configure key system registers to minimize wake-up latency. The following list outlines important register options that influence wake-up time behavior:

1. **MODEDLY field in PMC register:** It configures a delay timer that defines how long the system should wait for a higher-power PMIC configuration to take effect after the `PMIC_MODE` signal changes state. This delay ensures that the power rails stabilize before the system resumes normal operation. When setting this value, developers must carefully consider the timing requirements of their power management circuitry to ensure reliable transitions and avoid premature wake-up or unstable behavior.
2. **PMIC ramp up:** The MIMXRT700-EVK integrates NXP's PCA9422 PMIC, which supports configurable ramp-up speeds for voltage rails. By adjusting this setting, developers can achieve faster voltage ramp-up during transitions, reducing overall system wake-up time. This configuration should be evaluated alongside system constraints to ensure reliable operation.
3. **PMC register access clock:** The PMC instances are attached to the `VDD1_SENSE` bus. For either CPU to be able to access the PMC registers, the main clock for Sense Domain (`SENSE_MAIN_CLK_1`) must be enabled. As a result, the timing characteristics of register access are directly influenced by the clock source selected for the Sense domain.

Wake-up time from Deep Sleep mode is characterized by measuring the latency between the assertion of the MODESEL0 signal, indicating the transition to active mode, and a GPIO toggle executed immediately after the `__WFI()` instruction in firmware

The following configuration is used to evaluate wake-up latency on the MIMXRT700-EVK. In this case, CPU1 remains in Deep Sleep mode while CPU0 transitions from Deep Sleep to Active.

These settings are selected to simulate a realistic low power scenario while achieving the shortest possible transition back to active mode.

- **Test #1:**
  - FRO0 as Source Clock for Compute domain (CPU0) @ 325 MHz
  - 5.5 MB SRAM retained
  - PMC[MODEDLY] = 0, equal to 250 ns delay
  - PMIC Buck1 (VDD2) set to 25 mV/1 µS ramp
  - FRO Fast Startup **Disabled**
  - Wake up time = **126.51 µs**

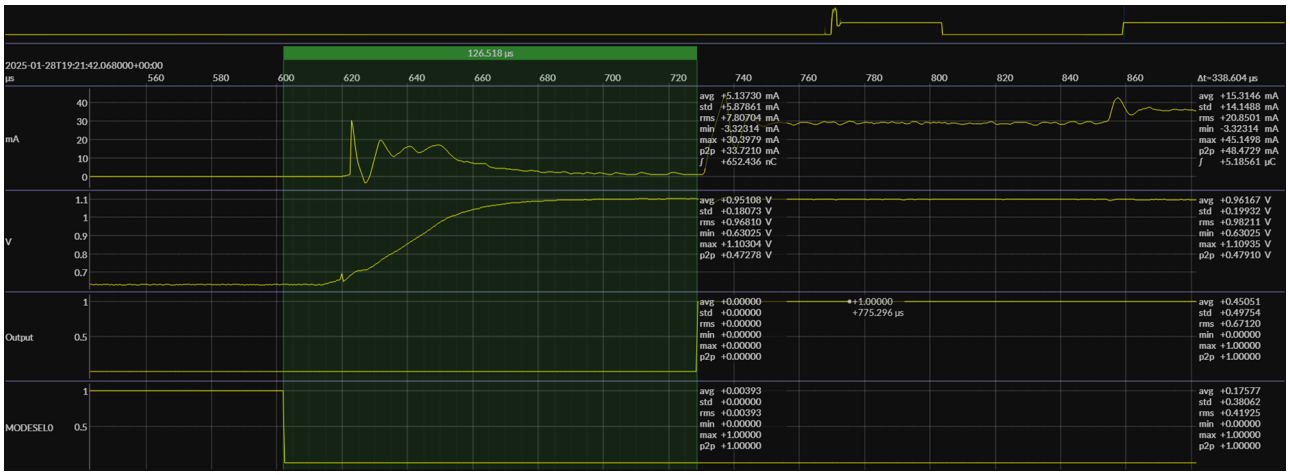



**Figure 10. Wake-up time measurement Test #1**

- Test #2:
  - FRO0 as Source Clock for Compute domain (CPU0) @ 325 MHz
  - 5.5 MB SRAM retained
  - PMC[MODEDLY] = 0, Equal to 250 ns delay
  - PMIC Buck1 (VDD2) set to 25 mV/1 µS ramp
  - FRO Fast Startup **Enabled**
  - Wake up time = **95.08 µs**


**Figure 11. Wake-up time measurement Test #2**

As demonstrated in the oscilloscope captures, enabling the FRO Fast Wake-Up feature results in approximately 30 % reduction in wake-up latency, assuming all other operating conditions remain constant.

# 4 Conclusion

The i.MX RT700 microcontroller provides a robust and configurable power management framework that supports both Deep Sleep and Asynchronous Deep Sleep modes. By using features such as segmented power domains, SRAM retention control, and fast oscillator startup, developers can significantly reduce power consumption while maintaining system responsiveness. Careful tuning of voltage levels, wake-up sources, and peripheral retention allows for optimized energy efficiency tailored to application-specific requirements. These capabilities make the i.MX RT700 a compelling solution for embedded systems with demanding low-power and fast wake-up needs.

# 5 Glossary

**Table 7. Glossary**

| Acronym | Definition |
|---|---|
| EVK | Evaluation Kit |
| PMIC | Power Management Integrated Circuit |
| FRO | Free Running Oscillator |
| WFI | Wait For Interrupt (ARM instruction) |
| SRAM | Static Random Access Memory |
| SDK | Software Development Kit |
| CPU0/CPU1 | Compute Domain Core/Sense Domain Core |
| VDD1/VDD2/VDDN | Voltage rails for Sense, Compute, and Media/Common domains respectively |
| SLEEPCON | Sleep Configuration Controller |
| PMC | Power Management Controller |
| RUNCFG/SLEEPCFG | Configuration registers for active and sleep modes |
| PDSLEEPCFG/PDRUNCFG | PMC registers for sleep/run state configuration |
| MODEDLY | Delay field in PMC register for power mode transition timing |
| RTC | Real-Time Clock |
| GPIO | General-Purpose Input/Output |
| LDO | Low-Dropout (linear) regulator |
| DCDC | Switching DC-to-DC converter |

## 6 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 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.

## 7 Revision history

Table 8 summarizes the revisions to this document.

**Table 8. Revision history**

| Document ID | Release date | Description |
|---|---|---|
| AN14852 v.1.0 | 10 November 2025 | Initial public release |

# 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.

**Suitability for use** — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

**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.

**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.

**Suitability for use in non-automotive qualified products** — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

**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.

# Contents

**1 Introduction ..... 2**
**2 Deep Sleep mode and Async Deep Sleep ..... 2**
2.1 Deep Sleep mode ..... 2
2.2 Deep Sleep Async mode ..... 2
2.3 Comparing Deep Sleep vs Deep Sleep Async ..... 2
2.4 Key features and benefits ..... 3
2.5 SLEEPCON and PMC aggregation ..... 3
2.6 Voltage setting considerations ..... 3
2.7 Software demo and measurements ..... 4
2.8 Deep Sleep Async mode testing ..... 7
2.9 Deep Sleep mode testing ..... 9
**3 Wake-up time considerations ..... 11**
**4 Conclusion ..... 13**
**5 Glossary ..... 13**
**6 Note about the source code in the document ..... 14**
**7 Revision history ..... 14**
**Legal information ..... 15**