AN219528

PSoC® 6 MCU Low-Power Modes and Power Reduction Techniques

Author: Brian Lee

Associated Part Family: All PSoC 6 MCU parts

Associated Code Examples: For a complete list, click here.

Related Application Notes: For a complete list, click here.

More code examples? We heard you.

To access an ever-growing list of hundreds of PSoC code examples, please visit our code examples web page. You can also explore the PSoC video library here.

AN219528 describes how to use the PSoC 6 MCU power modes to optimize power consumption. Major topics include the low-power modes in PSoC 6 MCU devices, and power management techniques using PSoC 6 MCU features. Three associated code examples demonstrate different low-power techniques.

Contents

1 Introduction

PSoC 6 MCU gives the best power saving benefit when low-power modes are implemented with other power-saving features and techniques, without sacrificing significant performance. This application note describes not only general power-saving methods but also the unique low-power modes in PSoC 6 MCU. It also discusses other low-power considerations. This application note requires a basic knowledge of the PSoC architecture, and ability to develop a PSoC 6 MCU application using PSoC Creator™. If you are new to PSoC 6 MCU, see AN210781 – Getting Started with PSoC 6 MCU with Bluetooth Low Energy (BLE) Connectivity. If you are new to PSoC Creator, see PSoC Creator™ Integrated Design Environment (IDE).

2 Power Modes

2.1 Power Mode Transition

PSoC 6 MCU features six power modes of operation. These are Active, Low Power Active (LPACTIVE), Sleep, Low Power Sleep (LPSLEEP), Deep Sleep, and Hibernate. Table 1 lists the power modes in which the devices operate.

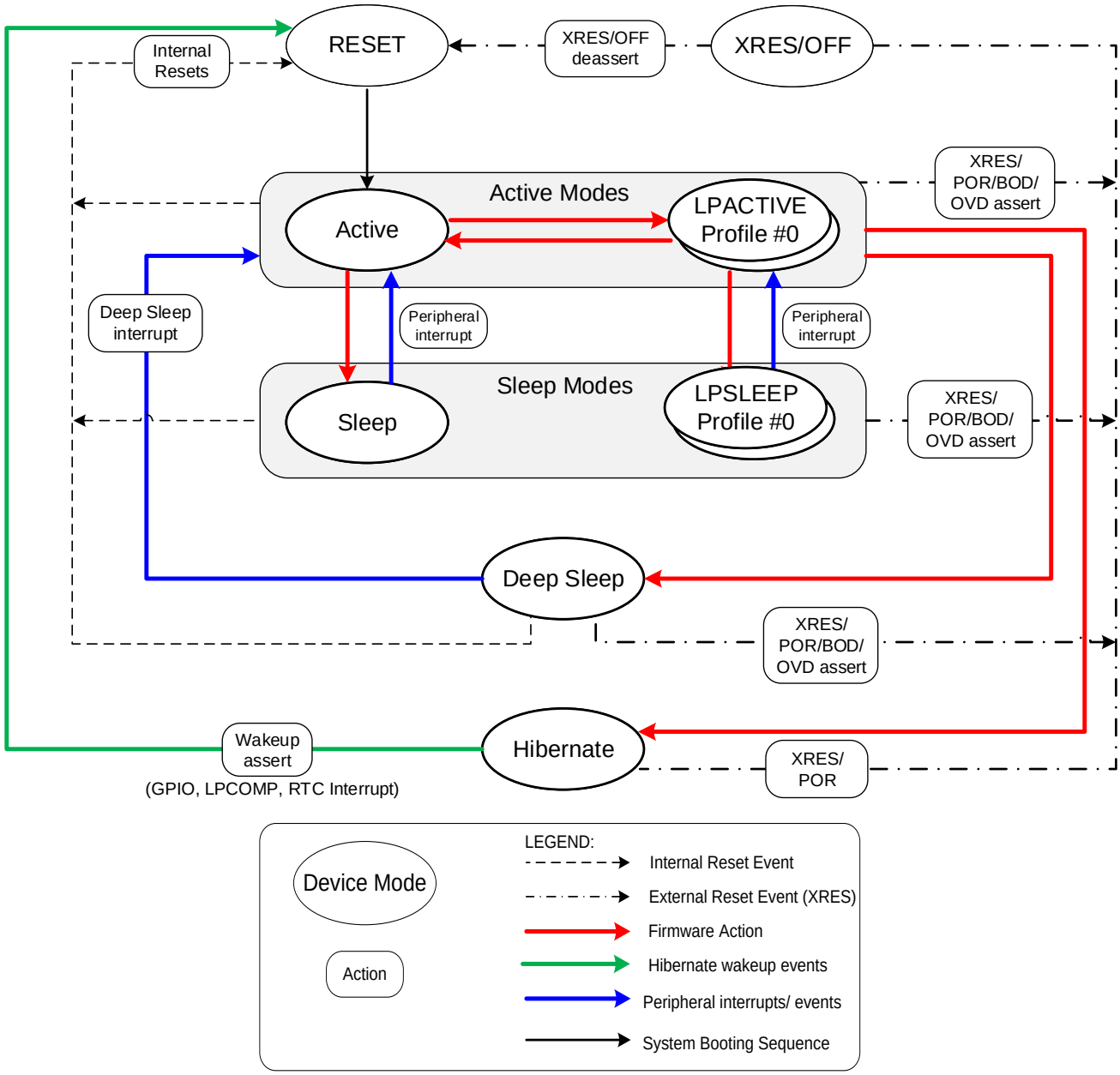
Table 1. Power Mode Description

Power Mode	Description
Active	• Code execution by either or both CPU cores • All blocks are available with maximum power and speed
Low Power Active	• Code execution by either or both CPU cores • All blocks are available, but limited high-frequency clock sources to achieve lower system current • Regulator and clock configuration
Sleep	• No code execution • All peripherals are available
Low Power Sleep	• No code execution • All peripherals are available with limited current and clock frequency. • Multiple options for regulator and clock configuration
Deep Sleep	• Cores, most peripherals and high-frequency clocks are off • Low frequency clock is on • Low-power analog and some digital peripherals are available for operation and as wakeup sources • SRAM is retained
Hibernate	• Cores and clocks are OFF • GPIO states are frozen • Low-power comparator, RTC alarm, and dedicated WAKEUP pins are available to wake up the system • Backup domain is available. For more information on backup domain, see External PMIC Control and PSoC 6 MCU Architecture Technical Reference Manual.

Figure 1 shows power mode transitions are based on different events and actions, including interrupts, firmware actions, and reset events. In some cases, mode transitions are done through multiple modes, for example Sleep to LPSLEEP.

For more detailed information, see PSoC 6 MCU Architecture Technical Reference Manual and Appendix A.

Figure 1. PSoC 6 MCU Device Power Mode Transition Diagram

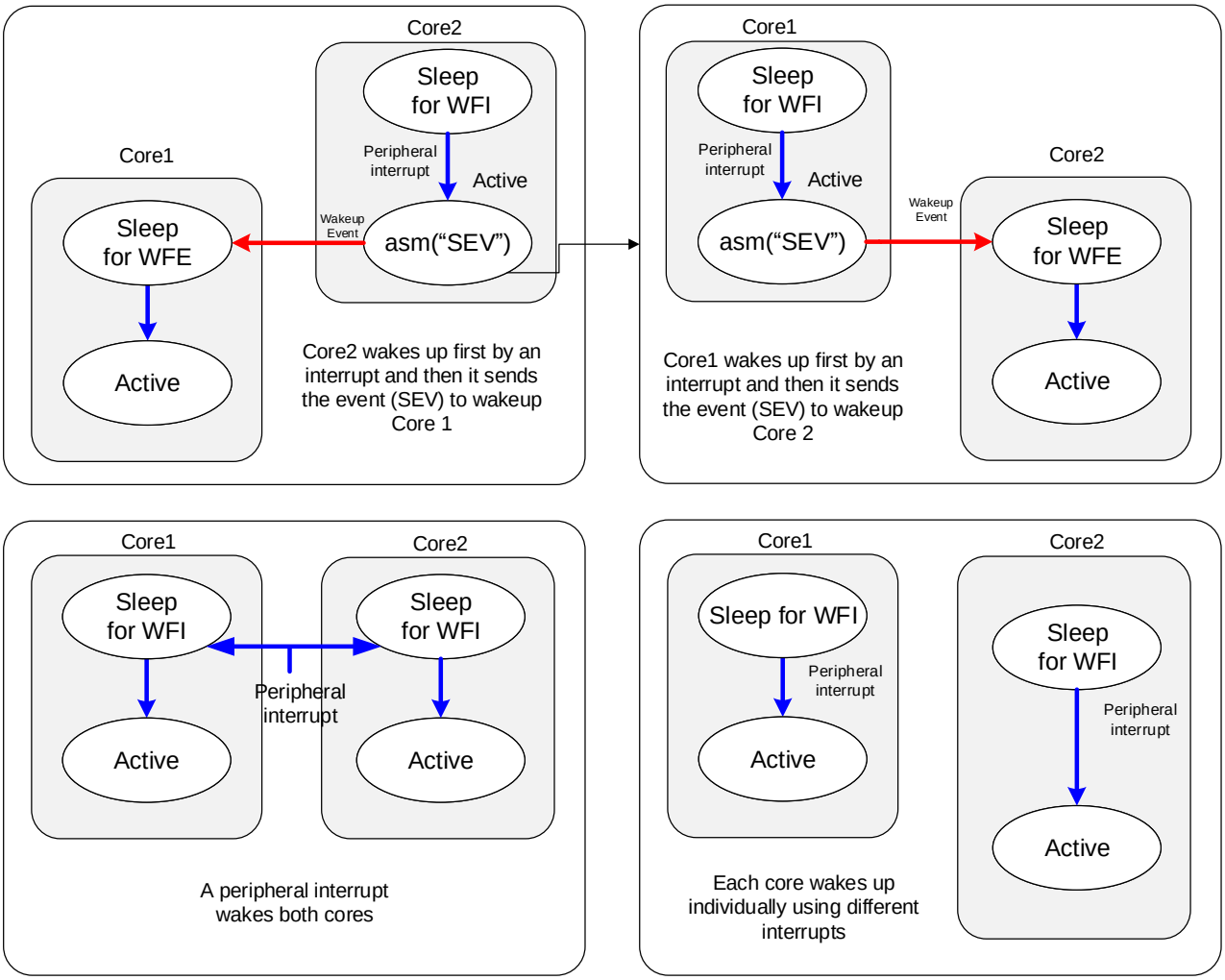


2.2 Core Sleep and Wakeup Instructions

ARM® cores transition between sleep and wakeup independently. Figure 2 shows several scenarios of wakeup from sleep.

Wait-for-Interrupt (__WFI) is the core sleep instruction. After a core executes __WFI, the core goes to sleep and stays in sleep until any interrupt is asserted. Wait-for-Event (__WFE) is a similar to __WFI, but it wakes up when the wakeup event is received instead of interrupt. Set Event (__SEV) is used for waking up other cores in sleep mode because of a __WFE. Deep Sleep uses the same instructions for sleep and wakeup, but the SLEEPDEEP bit[2] of ARM system control register (SCR) is set before a sleep instruction. For more information on SCR, see ARM System Control Register User Guide.

Figure 2. Multi Core Sleep and Wakeup Cases



The CPU power states are different from device power modes. Figure 2 shows that each CPU core supports its own sleep modes, independent of the state of the other core. The device is in Sleep or Deep Sleep mode when both cores are in sleep or deep-sleep, respectively. For more detailed information, see AN215656 – PSoC 6 MCU Dual-Core CPU System Design.

2.3 Subsystem Availability and Power Consumption

2.3.1 Subsystem Availability

Each system resource works differently in power modes. For example, the CPU can be in ON, OFF, and Retention modes. It is important to select proper peripherals for the power mode to work correctly. Table 5 lists the resources available in different power modes.

2.3.2 Approximating Power Consumption

The device datasheet provides power consumption data for a given condition. As there are different combinations to achieve best power consumption, the actual power consumption of the application can be different from the datasheet.

2.4 Example Case Scenarios

Proper power mode selection saves power consumption without performance degradation. Table 2 lists sample case scenarios of power modes. In some examples, only few power modes are used effectively.

Table 2. Sample Case Scenarios of Power Modes

	Wearable device	Air conditioner	Remote Controller	Thermometer
Active	GUI interaction by user	Motor run	-	Communicates over BLE
Low Power Active	Processes heartbeat	-	Sends command	Reads temperature Updates result on LCD
Sleep	-	-	-	-
Low Power Sleep	Analog block detects heartbeat	-	-	-
Deep Sleep	Goes to Deep Sleep when the device does not detect heartbeat for 30 seconds (device is not in use)	Waits for command No motor run Wakeup by Infra-Red (IR) triggering	-	Wakes up every 1 second using watchdog timer (WDT)
Hibernate	Low battery – Does nothing Resets device when charge is plugged	-	Waits for button press	-

2.5 System Power Management (SysPM) Library

2.5.1 Overview

The Cypress peripheral driver library (PDL) is a complete software tool that includes APIs for configuring peripherals and system register to implement the desired functionality. It reduces the need to understand register information.

Within the PDL, the SysPm API provides functions to change power modes as shown in Figure 1. The API can also register callback functions to execute a peripheral function before or after a power mode transition as shown in Figure 4.

2.5.2 Mode Transition Functions

Figure 1 shows the firmware transitions for power modes. SysPm provides the default five transition functions for Sleep, Deep Sleep, Hibernate, Enter LP, and Exit LP.

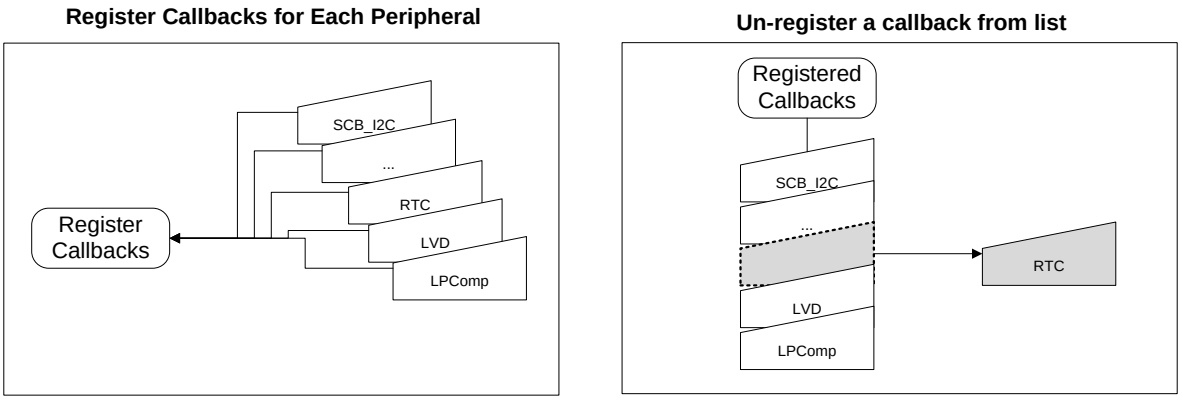
The power mode changing functions provide four different callback operations to execute a necessary action for each peripheral:

- `CY_SYS_PM_CHECK_READY`: Checks the ready state to transition to other mode. Exits without transition, if it returns `CY_SYSPM_FAIL`.
- `CY_SYSPM_BEFORE_TRANSITION`: Callbacks execute and configure required actions before mode transition.
- `CY_SYSPM_AFTER_TRANSITION`: Callbacks execute after mode transition or configuration.
- `CY_SYS_CHECK_FAIL`: Callbacks execute only when `CY_SYSPM_CHECK_READY` fails. It executes roll back action.

SysPm provides three functions for callback: registration, un-registration, and execution. These functions not only help in power optimization, but also in preventing an abnormal peripheral state after mode transition. The PDL expects the user to register callbacks for each power mode, as shown in Figure 3. Most peripheral drivers have predefined callbacks associated with each power mode. User can choose to register the defined peripheral callback or can make a custom callback. The SysPm transition function executes the registered callbacks one-by-one. The first registered function is executed first.

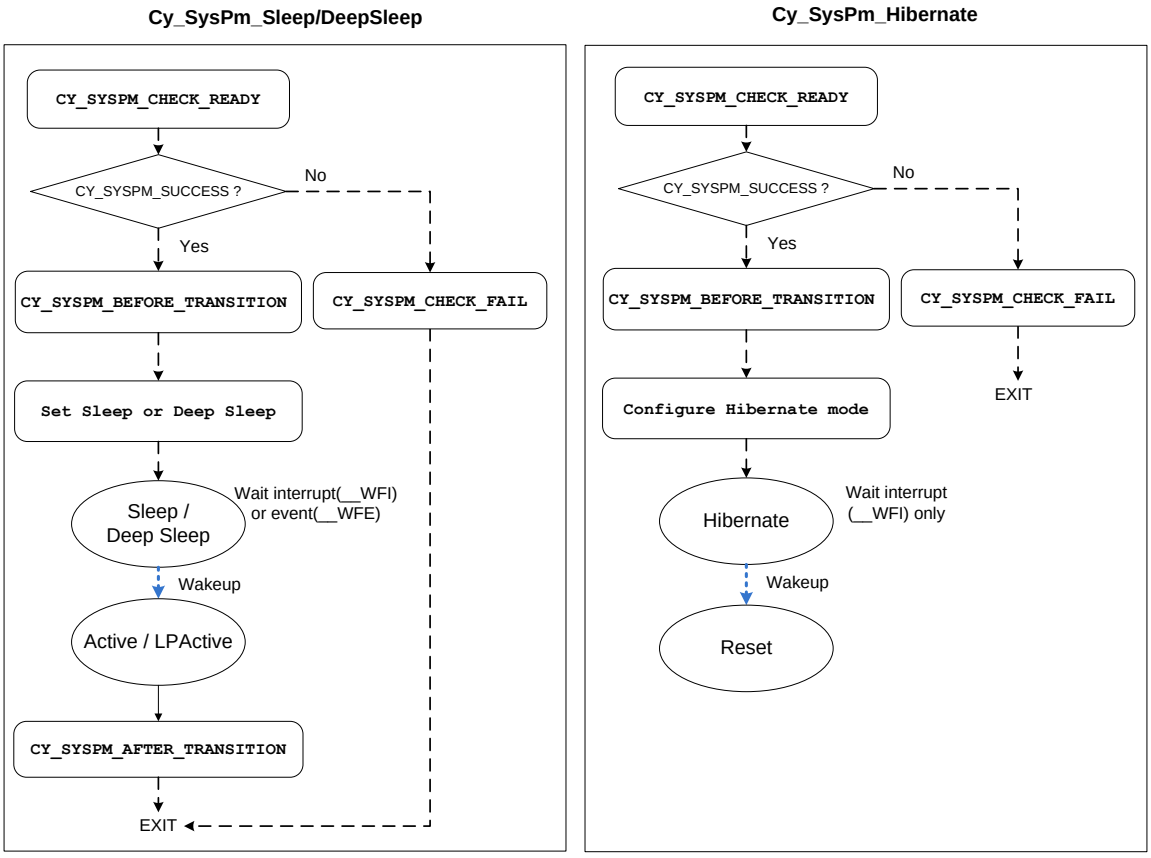
For more information on callback registration and implementation, see Appendix D and E.1 CE219881 - PSoC 6 MCU Switching between Power Modes, which is the mode transition example for Active, LPACTIVE, Sleep, LPSLEEP, and Deep Sleep.

Figure 3. Power Mode Callback Registration and Un-registration



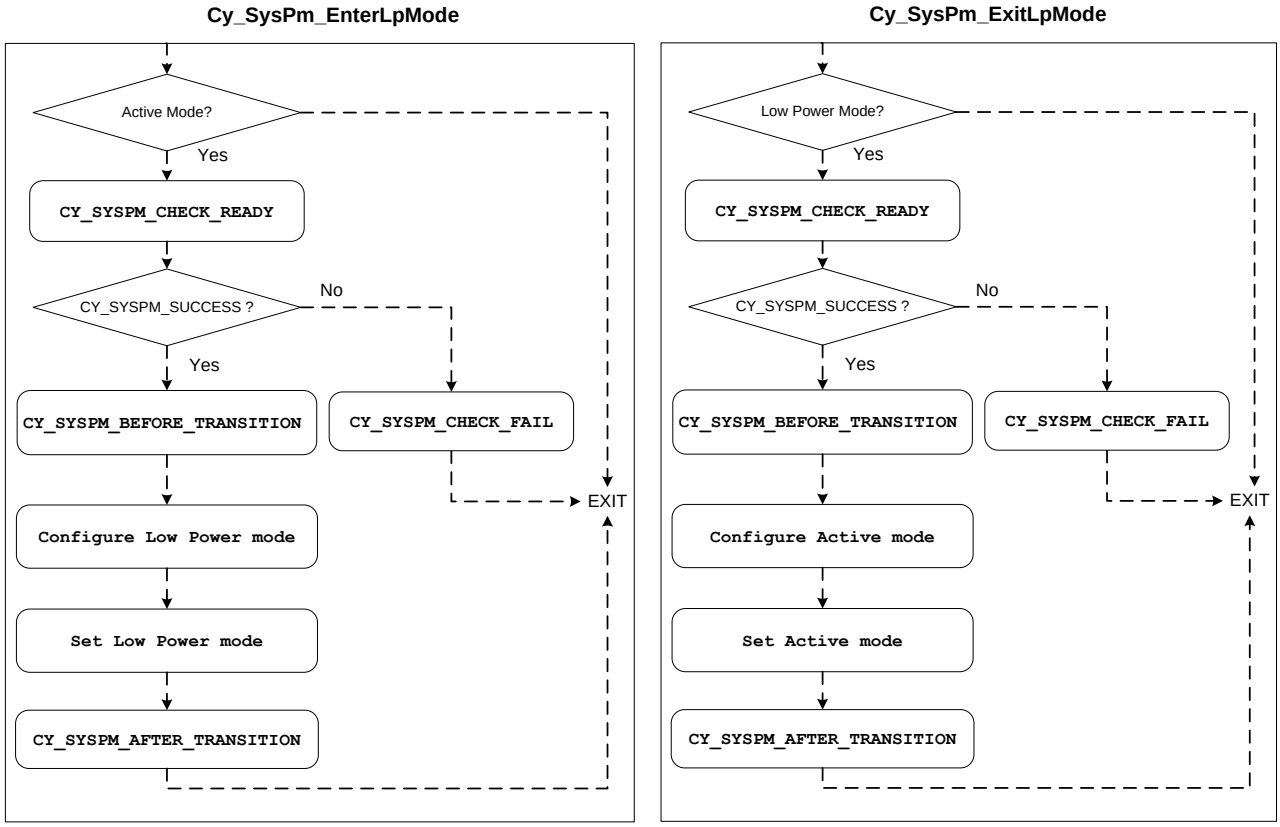
By calling the mode transition function, the device starts to transition with four callback operations. The Sleep, Deep Sleep, and Hibernate modes use ARM sleep instructions. The code execution stops and waits for an interrupt during the power mode. Figure 4 shows the CPU waiting for a wakeup source by calling sleep instruction: WFI() or __WFE(). So, the actual mode transition to Sleep, Deep Sleep, and Hibernate is done after the sleep instruction is executed. After wakeup, the device automatically transitions to Active, LPACTIVE, or Reset.

Figure 4.Sleep/ Deep Sleep/ Hibernate Mode Transition Flowchart



In the Active and LPACTIVE modes, all system resources keep running. So, entering and exiting LP mode are done by configuring the power mode control register, and there is no sleep and no wait any interrupt. SysPm PDL provides the associated driver functions, as shown in Figure 5 . For the best power efficiency, it is necessary to configure the core voltage regulator and the system clock. For more detailed information, see 3.1 Core Voltage Selection, 3.2 Low Power Mode Clock and Peripheral Driver Library Document (**PSoC Creator** > **Help** > **Documentation** > **Peripheral Driver Library**).

Figure 5. Low Power Mode Transition Flowchart



3 PSoC 6 MCU Power Management Techniques

3.1 Core Voltage Selection

3.1.1 Linear Regulator and Buck Regulator

PSoC 6 MCU supports multiple on-chip regulators [low drop out (LDO) and single input multiple output (SIMO) buck] for core power, as listed in Table 3. The LDO can provide up to 300 mA when the output voltage is 1.1 V and 25 mA when the output voltage is 0.9 V. The SIMO buck regulator can provide up to 20 mA for one output and 30 mA for combined output. The SIMO buck regulator gives a better efficiency under normal load conditions. Once switched to SIMO, it is not recommended to switch back to LDO without resetting.

Table 3. Different Options of Core Voltage Regulators for Low-Power Profile

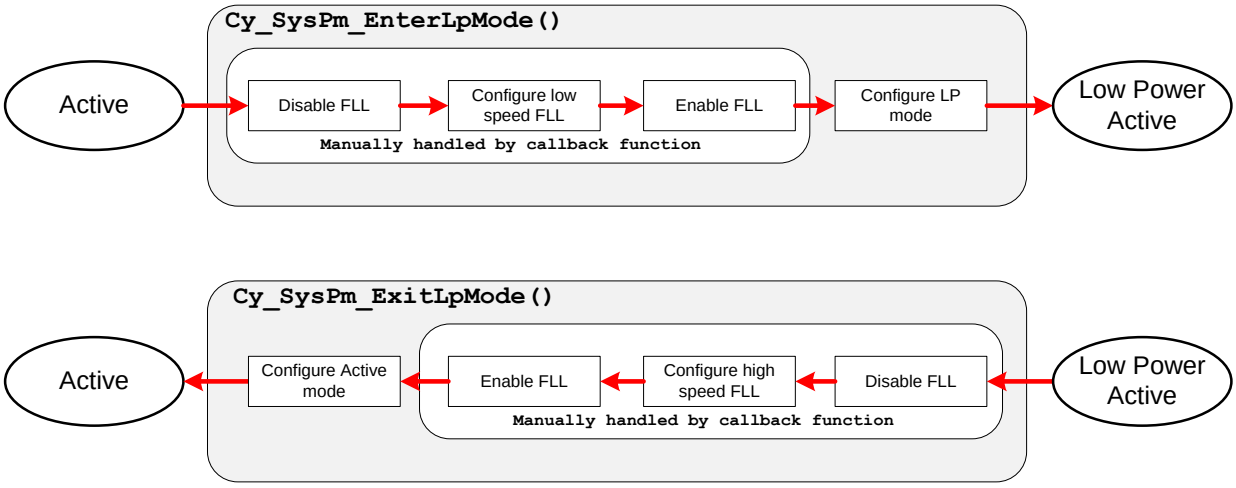
	Output	Max Load	Max Clock Frequency
LDO	0.9 V	25 mA	50 MHz for Cortex®-M4 25 MHz for Cortex-M0+
	1.1 V	300 mA	Allow maximum supportable clock frequency
SIMO Buck	0.9 V	20 mA	50 MHz for CM4 25 MHz for CM0+
	1.1 V	20 mA	50 MHz for CM4 25 MHz for CM0+

3.2 Low Power Mode Clock

Transition to a low-power mode can be done by configuring the power mode control register, instead of the ARM core configurations (_WFI, _WFE, and _SEV). There is a maximum clock speed limitation in Low Power mode as described earlier, so the clock configuration should be adjusted based on the regulator output when entering or exiting LP mode. PDL provides associated functions to configure the PWR_CTL register. For more information, see PSoC 6 MCU Registers Technical Reference Manual.

Figure 6 shows how to transition between Active and LPACTIVE using PDL functions with the registered callback function. Because of the clock limitation of LP mode, either the FLL clock speed or HFClk should be adjusted before the mode transition, if HFClk is faster than the limit. Changing FLL impacts the blocks that use the FLL clock, so all active peripherals should register own callbacks to handle the changing frequency. CE219881 – PSoC 6 MCU Switching between Power Modes provides an example of clock adjustment callback.

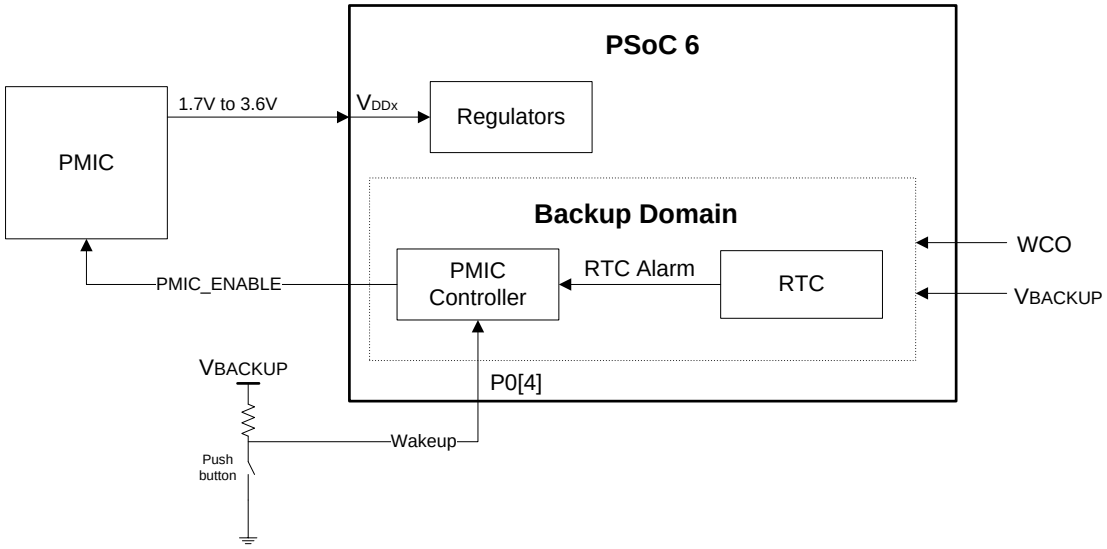
Figure 6. LP Mode Enter/Exit Transition



3.3 External PMIC Control

PSoC 6 MCU backup block provides power management IC (PMIC) control functions. Figure 7 shows the external PMIC supplying system power (V_{DDD}). PSoC 6 MCU can be shut down completely by PMIC, but a backup power to control PMIC keeps the backup domain alive.

Figure 7. An External PMIC Control Block Diagram



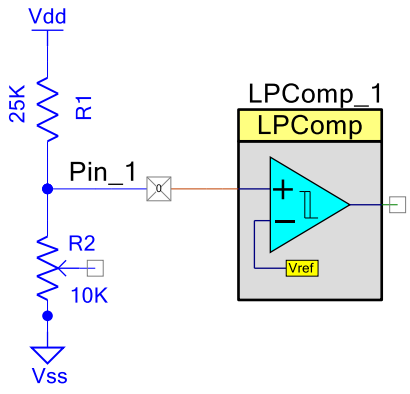
4 Other Power Saving Techniques

4.1 Use PSoC to Gate Current Paths

Your PCB may contain other components that draw power; PSoC 6 MCU can be used to control the current through them. For example, GPIO can draw the current. Note that the maximum pin source and sink capabilities listed in the datasheet must not be exceeded.

A good example of this scenario is a Low-power comparator (LPComp) application, as shown in Figure 8. In this case, the PSoC device compares the voltage on an analog pin, which changes as the potentiometer resistance changes.

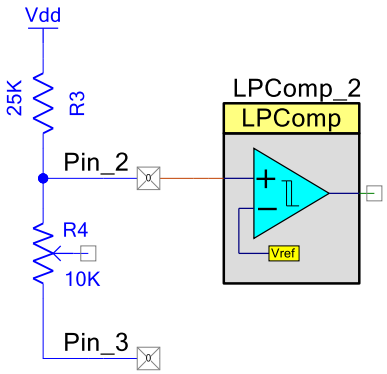
Figure 8. Typical LPComp Application



The LPComp can be turned OFF when not in use, but external components will still consume power because the current path through the resistor and potentiometer remains. A simple solution with PSoC is to use a second pin as a switch to ground, as shown in Figure 9.

In this configuration, the current flow can be stopped by writing a ‘1’ to Pin_3 and allowing the pin to float. This removes the current consumption by reducing the voltage differential across the two resistors to 0 V. Writing a ‘0’ resumes the current flow. The resource use of this power-saving feature is only one pin and a few lines of code.

Figure 9. Using a GPIO as a Ground Switch



4.2 Disable Unused Blocks

You can save unnecessary current consumption by disabling unused blocks.

4.3 Use DMA to Move Data

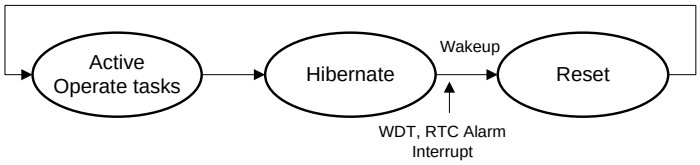
You can save power any time you offload a task from the CPU and either halt the CPU or let it do something else in parallel. The DMA engine that can be used in Active or Sleep modes to transfer data with no CPU use.

4.4 Periodic Wakeup Timers

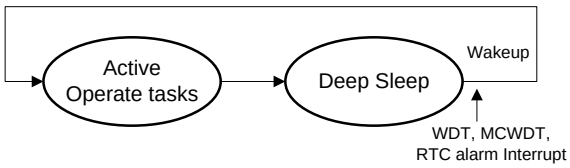
The periodic wakeup from sleep method is a traditional way to save power consumption. The average power consumption is determined by Active period power consumption and Sleep period power consumption. To achieve best result, the Sleep period should be as long as possible and the Active period should be as short as possible. WDT and multi-counter WDT (MCWDT) can be good periodic wakeup sources in Deep Sleep mode; WDT runs in Hibernate Mode. If your application needs a longer wakeup period, RTC alarm can be a good periodic wakeup source. See E.3 CE218542 - PSoC 6 MCU Custom Tick Timer Using RTC Alarm Interrupt for a code example on RTC periodic timer.

The following scenarios show how to change the mode states:

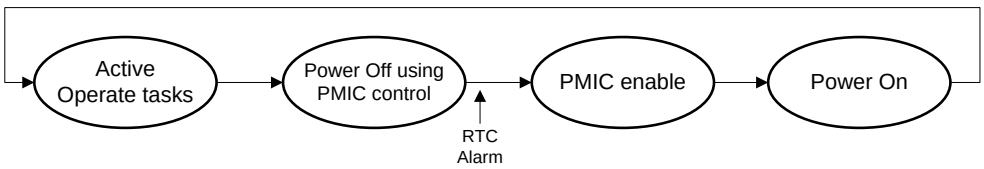
- Active and Hibernate



- Active and Deep Sleep



- Active and Power OFF with an external PMIC

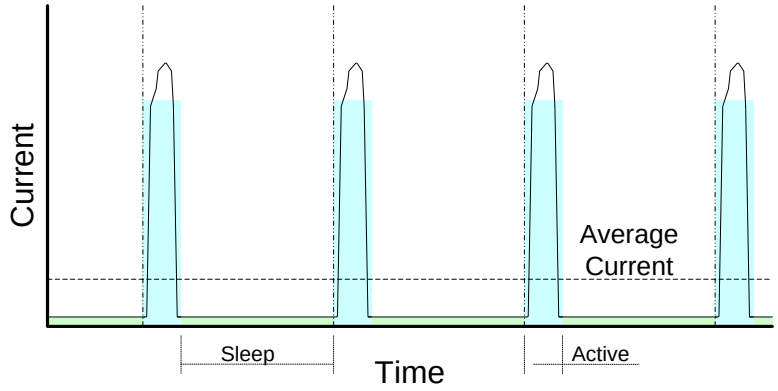


4.5 Clocks

In some cases, running the clocks faster can result in a lower average current consumption. For example, consider a PSoC design that takes a reading from a sensor once every second, performs several calculations, and then transmits the results to another device.

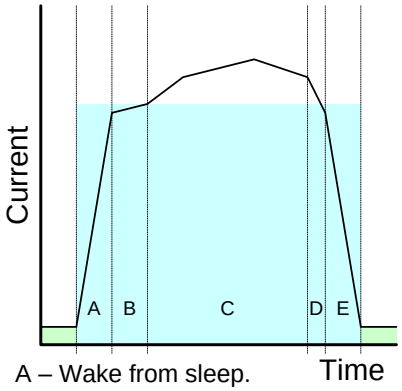
You can use Sleep or Deep Sleep mode to reduce the power when the PSoC device is idle, but the average current consumption is higher because of the time spent in Active mode. Figure 10 is a representation of the current consumption of this example with the system clocks set at 3 MHz.

Figure 10. Example Current Profile with 3-MHz Clocks



Depending on the tasks or calculations that are being performed when the PSoC device is awake, it may be possible to complete them sooner by running the system clocks faster. This can reduce the average current consumption because the PSoC device is in Active mode for less time. Figure 11 is a representation of Active mode timing, broken up into tasks.

Figure 11. Analysis of Tasks in Active Mode at 3 MHz

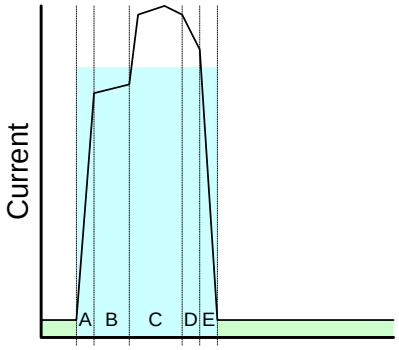


A – Wake from sleep.
B – Read sensor data.
C – Manipulate data.
D – Transmit result.
E – Go back to sleep.

The time required for some tasks does not change even if the system clock frequency increases. Sensor reading and data transmitting fall into this category. Other tasks, however, require less time if the CPU operates at a faster frequency.

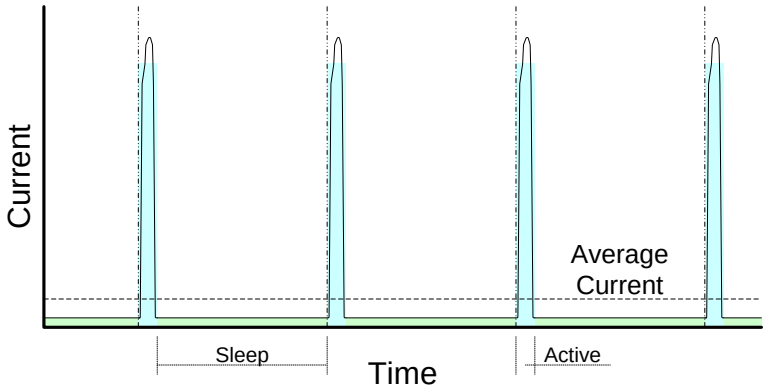
At some point, the benefit of a shorter active time is overcome by the energy required to drive the clocks at a higher rate. Assume that the optimal speed is 12 MHz, as Figure 12 shows. With a 12-MHz clock, the time spent in Active mode is about half the time spent with a 3-MHz clock. Figure 13 shows that the peak current consumption is greater when the clocks are faster, but the overall average is lower.

Figure 12. Analysis of Tasks in Active Mode at 12 MHz



A – Wake from sleep.
B – Read sensor data.
C – Manipulate data.
D – Transmit result.
E – Go back to sleep.

Figure 13. Example Current Profile with 12-MHz Clocks



4.6 GPIOs

GPIOs can continue to drive the external circuitry when the PSoC device is in a low-power mode. This is helpful when you need to hold external logic at a fixed level, but it can lead to wasted power if the pins needlessly source or sink current.

You should analyze your design and determine the best state for your GPIOs during low-power operation. If holding a digital output pin at logic 1 or 0 is best, match the same digital level using `Cy_GPIO_Write()`.

```
/* Set MyPin to '0' for low power. */
Cy_GPIO_Write(MYPIN_0_PORT, MYPIN_0_NUM, 0u);
```

Configure all unused GPIOs to Analog HI-Z unless there is a specific reason to use a different drive mode. A Pins Component's port-wide drive mode may be set using the `Cy_GPIO_SetDrivemode()` function.

```
/* Set MyPin to Alg HI-Z for low power. */
Cy_GPIO_SetDrivemode(MYPIN_0_PORT, MYPIN_0_NUM, CY_GPIO_DM_HIGHZ);
```

The flexibility of PSoC makes it easy to manage GPIO drive modes to prevent unwanted current leakage. In Hibernate mode, the GPIO drive modes and data registers may be “frozen.” The pin states are frozen by calling `Cy_SysPm_IoFreeze()`. This should be “unfrozen” after a wakeup reset to allow their states to change by calling `Cy_SysPm_IoUnfreeze()`. See Section E.2 CE218129 - PSoC 6 MCU Wakeup from Hibernate Using a Low-Power Comparator for a code example on Hibernate and IO control.

5 Power Supply Protection System

5.1 Hardware Control Power Supply Protections

5.1.1 Brownout Detect (BOD)

Brownout detect (BOD) can reset system before the logic crashes against the loss of V_{DDD} and V_{CCD} power. The brownout system guarantees a reset before V_{DDD} reaches the minimum system operating voltage, which works for all logic, SRAM, flash, and so on. It is controlled by hardware, and there is no configurable register.

5.1.2 Low-Voltage Detect (LVD)

Low-voltage detect (LVD) is similar to BOD but it is configurable. LVD generates an interrupt when V_{DDD} falls under a configured trip voltage. This interrupt helps to handle important data before triggering BOD reset. LVD provides 15 selectable trip voltages. LVD is not available in Deep Sleep and Hibernate modes.

5.1.3 Overvoltage Detect (OVD)

Overvoltage detect (OVD) is the reverse of BOD. These circuits generate a reset when unsafe over-voltage supply conditions on V_{DDD} and V_{CCD} are detected. No firmware control is required. OVD helps to protect the system from high voltage damage.

6 Summary

Many power managing options can be used in PSoC 6 MCU. By following proper methods, you can optimize your design and ensure that new power modes and features of PSoC 6 MCU give the best options for the lowest power consumption without degrading the performance of battery powered devices.

7 Related Documents

AN215656 – PSoC 6 MCU Dual-Core CPU System Design

CE219881 – PSoC 6 MCU Switching between Power Modes

CE218542 – PSoC 6 MCU Custom Tick Timer Using RTC Alarm Interrupt

CE218129 – PSoC 6 MCU Wakeup from Hibernate Using a Low-Power Comparator

About the Author

Name:	Brian Lee
Title:	Application Engineer Principal
Background:	Brian Lee has a BSEE from Yeungnam University, South Korea. He has many years’ experience developing cellular phone and MCU based embedded system.

Appendix A. Power Mode Details

A.1 Active

Active mode is the primary mode of operation. One or both cores are executing codes, and all logic and memories are powered. So, this mode has the best performance with the highest power. There is no limit to operate peripherals with maximum powers and speed.

A.1.1 Core Clocks and Linear Core Regulator

The linear core regulator generates V_{CCD}, which is 1.1 V for Active mode. It can drive a maximum 100 mA load current. Cores can run with maximum clock speed.

A.1.2 Mode Transition

The mode transition uses an interrupt or a firmware action for Active mode. As Figure 1 shows, it uses the firmware action when there is a transition from Active to another mode. But opposite transitions use only an interrupt. The mode automatically transitions to Active without firmware action. The device enters Active upon any device reset.

A.2 Low Power Active (LPACTIVE)

LPACTIVE has the same functionality of Active, but the mode has some tradeoffs to reduce power consumption. The major differences of this mode are a current limit and a clock frequency limit.

A.2.1 Core Clocks and Regulator

Low Power Active mode uses V_{CCD} 0.9 V. The load current is limited to 25 mA. The CM4 core clock can be up to 50 MHz and the CM0+ core clock can be up to 25 MHz.

A.2.2 Mode Transition

The device transitions from Active to LPACTIVE when the firmware action sets the low-power configurations in Active. It returns to LPACTIVE when wakeup from Sleep or Deep Sleep, if the low-power configurations are enabled. But the device enters Active upon any device reset, even when previous low-power configurations were enabled.

A.3 Sleep

In Sleep mode, ARM cores (CM0+ and CM4) do not execute any codes and stay in Sleep by calling the ARM sleep instruction (__WFE or __WFI), but all other peripherals are available. Each core can be dismissed from the Sleep state independently.

A.3.1 Peripherals

The device can use all system resources like Active and Low Power Active as listed in Table 5.

A.3.2 Cores States

The CPU clock is turned off and the CPU enters sleep.

A.3.3 Wakeup Sources

Any registered interrupt (masked to CPU) and the __SEV event from a core can be wakeup sources when the CPU is in Sleep mode. Figure 2 shows the wakeup flow using __WFI, __WFE, and __SEV.

A.3.4 Mode Transition

Firmware actions are required to transition to Sleep mode for both cores. By calling the firmware function that enters Sleep mode, Active transitions to Sleep. As Figure 1 shows, the other modes are not allowed to transition to Sleep except Active.

A.4 Low Power Sleep (LPSLEEP)

LPSLEEP has same functionality as Sleep mode, but the mode has some tradeoffs to reduce power consumption. The major difference of this mode is a current limit.

A.4.1 Peripherals

As listed in Table 5, a device can use all system resources like Sleep, but universal digital block (UDB) will slow down. The maximum device current should not exceed 25 mA, so the clock frequency must be reduced.

A.4.2 Cores States

The CPU clock is turned OFF and the CPU enters sleep.

A.4.3 Wakeup Sources

Any registered interrupt (masked to CPU) and the __SEV event from a core can be wakeup sources when the CPU is in Low Power sleep.

A.4.4 Mode Transition

If the firmware calls the Sleep mode transition function in LPACTIVE, the power mode changes to LPSLEEP. Both cores should be in Sleep. Otherwise, the system will stay in LPACTIVE. As Figure 1 shows, the other modes are not allowed the transition to LPSLEEP except LPACTIVE.

A.5 Deep Sleep

Deep Sleep turns OFF the system resources that use a high-frequency clock. However, low-frequency clocks (internal low-speed oscillator (ILO), precision ILO, and watch crystal oscillator (WCO)) and peripherals which use these low-frequency clocks continue to operate. The SRAM and CPU will be retained during Deep Sleep for fast transition to Active or LPACTIVE.

A.5.1 Peripherals

As Table 5 shows, the high-frequency clocks, such as the internal main oscillator (IMO), external crystal oscillator (ECO), phase-locked loop (PLL), and frequency lock loop (FLL) do not work in Deep Sleep.

Deep Sleep allows the internal low-speed oscillator (ILO) and WCO, so LCD, GPIO, watchdog timer (MCWDT, EDT), CTBm, and RTC can work properly even in an extreme low power consumption state.

The SRAM, SMIF, CSD, SCB, and backup register do not work, but stay in retention state for a quick wakeup.

A.5.2 Cores States

Each core stays in deep-sleep until an interrupt occurs and the CPU, SRAM, and all registers are retained.

A.5.3 Regulators

In Deep Sleep, the deep-sleep regulator (V_{CCD}SLP) and the retention regulator (V_{CC}RET) supply the power for the resources in retention and the various peripherals that are operating. The regulators are sourced by V_{DDD} or SIMO buck regulator. The SIMO buck regulator is efficient for low power consumption.

A.5.4 Wakeup Sources

Any interrupt of GPIO edge detect logic, the SEV event from a core, (LPComp), SCB I2C/SPI slave, CTBm, and RTC alarm can be a wakeup source.

A.5.5 Mode Transition

In Deep Sleep, both cores must be in deep-sleep by a firmware action. Even if a core wakes up, the other core stays in Deep Sleep until an interrupt occurs. As Figure 1 shows, the device returns to its previous state after wakeup.

A.6 Hibernate

Hibernate turns OFF most resources to achieve the lowest power. It is intended to put applications into a dormant state but it is not in OFF state, so V_{DDD} is present and some peripherals still work. The system and firmware should restart on wakeup, if the system is in Hibernate.

A.6.1 Peripherals

As Table 5 shows, most peripherals are not working even if ILO is available. The WDT and backup domain (RTC and alarm) remain functional. Some Hibernate registers in the system and backup registers stay in retention. The GPIO states are frozen. The LVD and BOD are disabled automatically to prevent accidental wakeup.

A.6.2 Cores States

The CPU and SRAM do not operate but two 4-byte Hibernate registers are retained.

A.6.3 Regulators

All internal regulators are OFF. V_{DDD} and V_{BACKUP} supply the power.

A.6.4 Wakeup Sources

Two wakeup pins, watchdog timer, LPComp, and RTC alarm can be the wakeup sources.

A.6.5 Transition

The firmware actions are required to transition to Hibernate mode. One of cores calls the function for entering Hibernate, and the entire system goes into Hibernate mode. As Figure 1 shows, it always returns to reset after wakeup to restart the system. Otherwise, the system will not run correctly because of unretained system resources. The firmware must check the reset cause to unfreeze the frozen GPIO when it resets from Hibernate.

A.7 RESET/ XRES/ OFF

External reset (XRES) and OFF modes are functionally similar. All resources are not powered and I/Os are tristated to prevent leakage current. After deasserting external reset or when POR/BOD is asserted, RESET starts but it is not a clear power mode.

Appendix B. Power Modes Summary

B.1 Power Modes and Wakeup Source

Table 4. Power Modes and Wakeup Source

Power Mode	Description	Entry Condition	Wakeup Sources	Wakeup Action
Active	• Primary mode of operation • Code execution • Supports maximum powers and speed for peripherals	• A boot sequence done after POR, XRES, and BOD • Firmware action from LPACTIVE: `Cy_SysPm_ExitLpMode()` • Peripheral interrupts from Sleep mode • Interrupts from Deep Sleep mode .	Not applicable	Not applicable
Low Power Active	• Same functionality of Active • Code execution • Limited current and clock frequency	• Firmware action from Active: `Cy_SysPm_EnterLpMode()` • Peripheral interrupts from LPSLEEP mode • Interrupts from Deep Sleep mode	Not applicable	Not applicable
Sleep	• Cores keep in sleep • No code execution • All peripherals are available	Firmware action from Active: `Cy_SysPm_Sleep()`	Any interrupt to CPU	Interrupt
Low Power Sleep	• Same functionality of LPACTIVE • No code execution • Limited current and clock frequency	Firmware action from LPACTIVE: `Cy_SysPm_Sleep()`	Any interrupt to CPU	Interrupt
Deep Sleep	• Cores, most peripherals, and high-frequency clocks are off • Low-frequency clock is on • Low-power analog and some digital peripherals are available for operation and as wakeup sources • SRAM is retained	Firmware action from Active or LPACTIVE: `Cy_SysPm_Deep Sleep()`	• GPIO interrupt • Low-power comparator • SCB • CTBm • Watchdog timer • RTC alarms	Interrupt
Hibernate	• Cores and clocks are off • GPIO states are frozen • Low-power comparator, RTC alarm, and dedicated WAKUP pins are available to wake up system • Backup domain is available	Firmware action from Active or LPACTIVE: `Cy_SysPm_Hibernate()`	• WAKEUP pin • Low-power comparator • RTC alarms	Reset

Appendix C. Subsystem Availability

C.1 Resources Available in Different Power Modes

Table 5 shows information for the resource availability in different power modes.

Table 5. Resources Available in Different Power Modes

Component	Power Modes							
	Active	Low Power Active[1]	Sleep	Low Power Sleep[1]	Deep Sleep	Hibernate	XRES Asserted	Power off with Backup[2]
Core functions								
CPU	ON	ON	Sleep	Sleep	Retention[3]	OFF	OFF	OFF
SRAM	ON	ON	ON	ON	Retention[3]	OFF	OFF	OFF
Flash	ON	ON	ON	ON	OFF	OFF	OFF	OFF
High Speed Clock (IMO, ECO, PLL, FLL)	ON	ON	ON	ON	OFF	OFF	OFF	OFF
LVD	ON	ON	ON	ON	OFF	OFF	OFF	OFF
ILO	ON	ON	ON	ON	ON	ON	OFF	OFF
System registers (ARM registers, Sub system registers)	ON	ON	Retention[3]	Retention[3]	Retention[3]	Retained Hibernate registers	OFF	OFF
Peripherals								
SMIF	ON	ON	ON	ON	Retention[3]	OFF	OFF	OFF
UDB	ON	ON (slow)[1]	ON	ON (slow)[1]	OFF	OFF	OFF	OFF
SAR ADC	ON	ON	ON	ON	OFF	OFF	OFF	OFF
CTBm	ON	ON	ON	ON	ON (lower GBW)	OFF	OFF	OFF
TCPWM	ON	ON	ON	ON	OFF	OFF	OFF	OFF
CSD	ON	ON	ON	ON	Retention[3]	OFF	OFF	OFF
BLE	ON	ON	ON	ON	Retention[3]	OFF	OFF	OFF
LCD	ON	ON	ON	ON	ON	OFF	OFF	OFF
SCB	ON	ON	ON	ON	Retention (I^2C/SPI wakeup available)[4]	OFF	OFF	OFF
GPIO	ON	ON	ON	ON	ON	Frozen[5]	OFF	OFF
LPComp	ON	ON	ON	ON	ON (Wakeup Interrupt)[6]	ON (Wakeup Interrupt)[6]	OFF	OFF
WDT	ON	ON	ON	ON	ON	ON	OFF	OFF
MCWDT	ON	ON	ON	ON	ON	OFF	OFF	OFF
Resets								

[1] Low Power Active/Sleep: In the Low Power Active/Sleep modes, the maximum device current should not exceed 25 mA (refer to the datasheet for min/max numbers). Firmware should enable/disable blocks and reduce the clock frequencies appropriately to ensure power consumption does not exceed the limit.

[2] The backup domain power is on but PMIC doesn't supply power to the device.

[3] Retention: The register values are retained through the Low Power mode and wakeup.

[4] Only the SCB with Deep Sleep support is available in the Deep Sleep power mode; other SCBs are not available in the Deep Sleep power mode.

[5] Frozen: The configuration, mode, and state of all GPIOs in the system are locked. Changing GPIO attribution is not allowed until the firmware unfreezes after system reset from Hibernate.

[6] It allows an external GPIO input against the local voltage reference (0.4 V ~ 0.6 V)

Component	Power Modes							
	Active	Low Power Active[1]	Sleep	Low Power Sleep[1]	Deep Sleep	Hibernate	XRES Asserted	Power off with Backup[2]
XRES	ON	ON	ON	ON	ON	ON	ON	OFF
POR	ON	ON	ON	ON	ON	ON	OFF	OFF
BOD	ON	ON	ON	ON	ON	OFF	OFF	OFF
Watchdog reset	ON	ON	ON	ON	ON	OFF	OFF	OFF
Backup domain								
WCO, RTC, Alarms	ON	ON	ON	ON	ON	ON	ON	ON
Backup register	ON	ON	ON	ON	Retention[3]	Retention[3]	Retention[3]	Retention[3]

Appendix D. Callback Function Example

D.1 Register Callback Functions

```
cy_stc_syspm_callback_params_t myParams;
cy_stc_syspm_callback_t myAppSleep =

{
       &Application_Callback,              /* Callback function */
       CY_SYSPM_SLEEP,                     /* Select Power Mode */
       (CY_SYSPM_SKIP_CHECK_READY |
        CY_SYSPM_SKIP_CHECK_FAIL),         /* Skip CHECK_READY and CHECK FAIL */
       &myParams,                          /* Operation, contexts */
       NULL,                               /* Previous list callback */
       NULL                                /* Next list callback */

};

cy_stc_syspm_callback_t myAppHibernate =
{
       &Application_Callback,              /* Callback function */
       CY_SYSPM_HIBERNATE,                 /* Select Power Mode */
       0U,                                 /* Skip mode, no skip */
       &myParams,                         /* Operation, contexts */
       NULL,                               /* Previous list callback */
       NULL                                /* Next list callback */

};

/* Register Callback functions for each power mode */
Cy_SysPm_RegisterCallback(&myAppSleep);
Cy_SysPm_RegisterCallback(&myAppHibernate);
```

D.2 Implement Custom Callback Functions

```
cy_en_syspm_status_t Application_Callback(
cy_stc_syspm_callback_params_t *callbackParams)
{
    cy_en_syspm_status_t retVal = CY_SYSPM_SUCCESS;

     switch(callbackParams->mode)
        {
          case CY_SYSPM_CHECK_READY:
           {
             if(Check_HW())
              {
                     /* Hardware is ready */
              }
            else
              {
                     retVal = CY_SYSPM_FAIL;
              }
           }
         break;

          case CY_SYSPM_CHECK_FAIL:
           {
           /* Rollback any configuration during CHECK_READY */
           Rollback_HW();
           retVal = CY_SYSPM_SUCCESS;
           }
         break;
```

```
                case CY_SYSPM_BEFORE_TRANSITION:
                {
                     /* configure HW for new mode before transition */
                     ConfigureHW_BeforeMode();
                     retVal = CY_SYSPM_SUCCESS;
                }
                break;

                case CY_SYSPM_AFTER_TRANSITION:
                 {
                     /* configure HW after mode transition */
                     ConfigureHW_AfterMode();
                     retVal = CY_SYSPM_SUCCESS;
                }
                break;

                default:

                    break;
            }

    return (retVal);
}
```

Appendix E. Code Examples

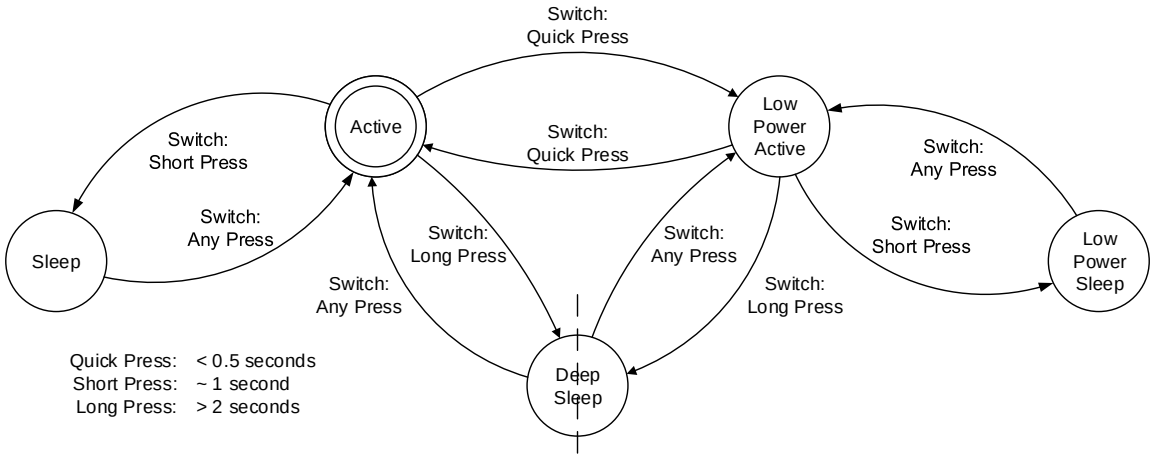
Following code examples are included with this application note to demonstrate PSoC 6 MCU power modes and power reduction techniques.

E.1 CE219881 - PSoC 6 MCU Switching between Power Modes

This code example shows how to enter and exit low-power modes, and transition from Active to Deep Sleep or Sleep. Once in either mode, the example also shows how to wake up and return to one of the Active modes.

The project uses a button switch to transition among the power modes and shows different LED colors to indicate the current power mode. Figure 14 shows the state machine implemented in the firmware to execute the transitions. For more information, see CE219881 - PSoC 6 MCU Switching between Power Modes.

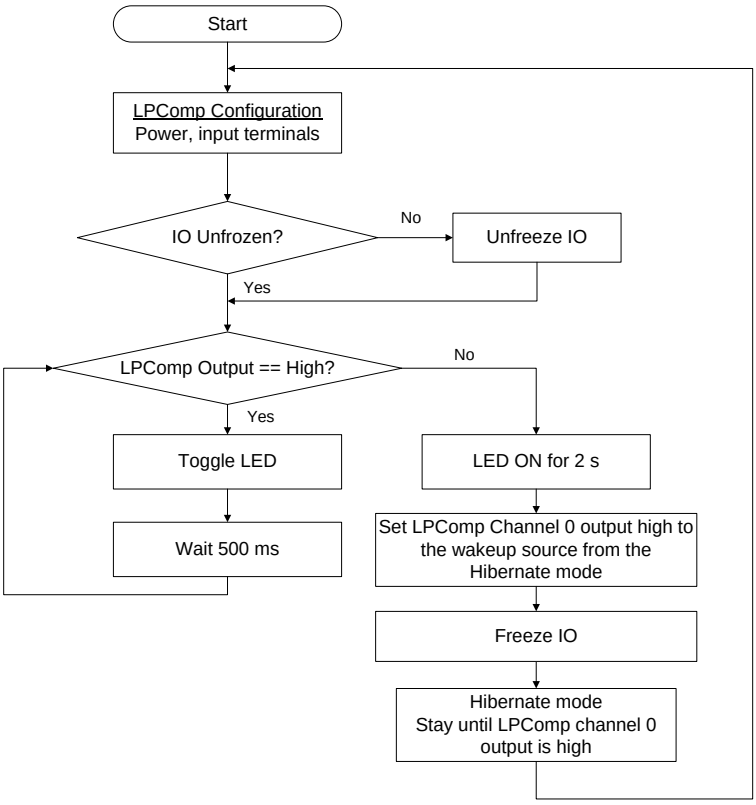
Figure 14. Power Mode State Machine



E.2 CE218129 - PSoC 6 MCU Wakeup from Hibernate Using a Low-Power Comparator

This code example demonstrates how to set the Component options for the LPComp internal reference voltage and how to set the external input from a GPIO using the LPComp driver. The project is a good example of Hibernate power mode transition. It teaches you how to handle the GPIOs before and after Hibernate, and it shows how to register the wakeup source for Hibernate. Figure 15 shows the basic flow of the project. For more information, see CE218129 - PSoC 6 MCU Wakeup from Hibernate Using a Low-Power Comparator.

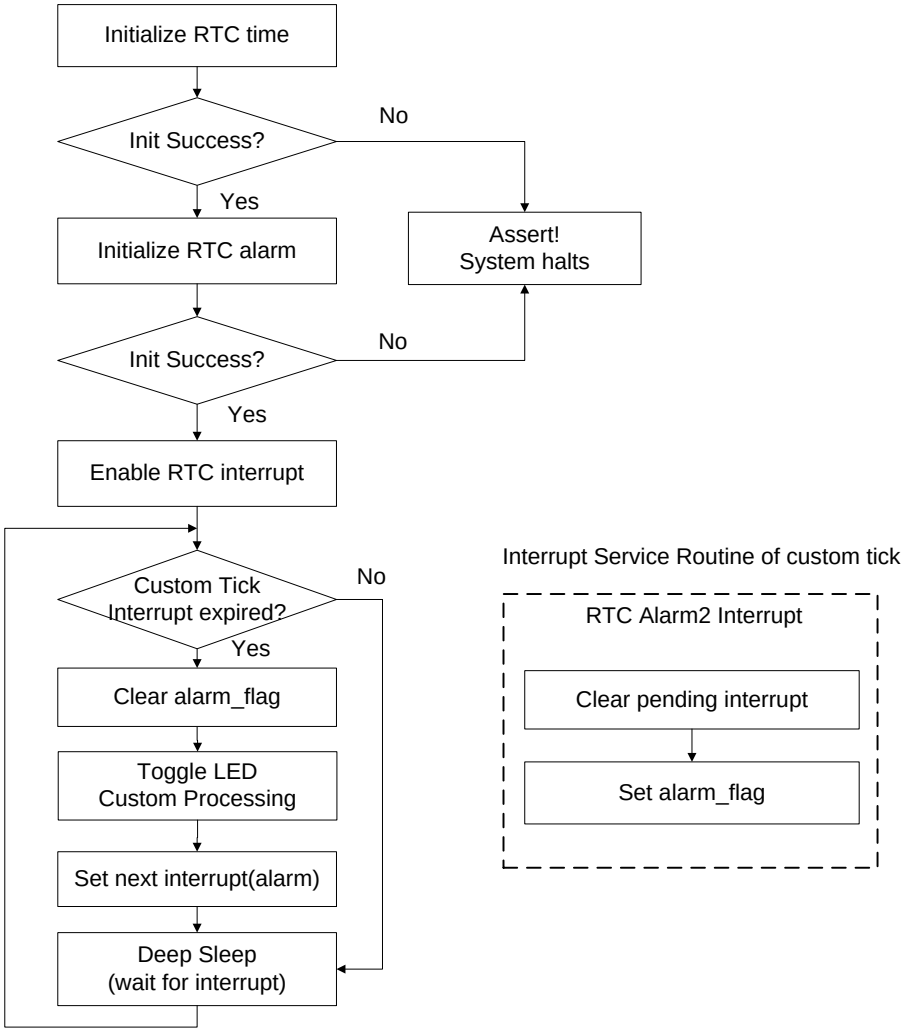
Figure 15. Wakeup from Hibernate Mode Using LPComp Input



E.3 CE218542 - PSoC 6 MCU Custom Tick Timer Using RTC Alarm Interrupt

This code example demonstrates how to configure the RTC registers for a periodic alarm interrupt using the PDL RTC driver API. The project uses Active and Deep Sleep mode to save power consumption. A GPIO output is included to toggle the LED to show the period of the interrupt. For more information, see CE218542 - PSoC 6 MCU Custom Tick Timer Using RTC Alarm Interrupt.

Figure 16. RTC Periodic Wakeup Timer using Alarm Interrupt



Document History

Document Title: AN219528 - PSoC® 6 MCU Low-Power Modes and Power Reduction Techniques

Document Number: 002-19528

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5862341	BOO	09/12/2017	New application note.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709