AN219528

PSoC 6 MCU Low-Power Modes and Power Reduction Techniques

Author: Brian Lee

Associated Part Family: All PSoC® 6 MCU parts

Related Documents: 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. 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 AN221774 - Getting Started with PSoC 6 MCU or 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 seven power modes that are split into system modes that affect the whole device, and standard Arm® CPU modes that affect only one CPU. The system power modes are Low-Power (LP), Ultra-Low-Power (ULP), deep sleep, and hibernate. The Arm CPU power modes are active, sleep, and deep sleep, and are available in system LP and ULP power modes. Table 1 lists the power modes in which the devices operate.

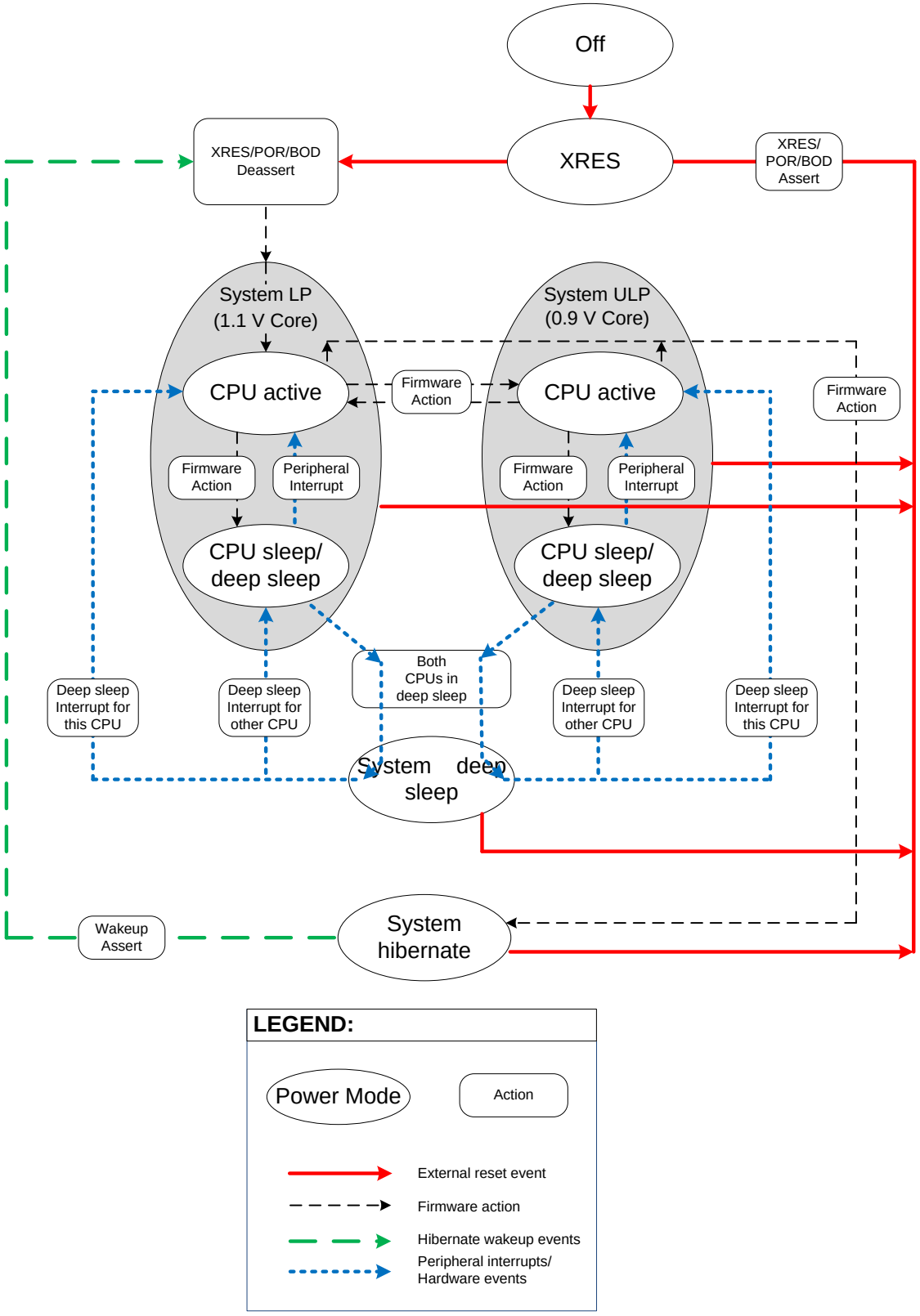
Table 1. Power Mode Description

Power Mode	Description
System LP	• All resources are available with maximum power and speed • All CPU power modes supported.
System ULP	• All blocks are available, but core voltage lowered resulting in reduced high-frequency clock frequencies. • All CPU power modes supported.
CPU active	• Normal CPU code execution • Available in system LP and ULP power modes
CPU sleep	• CPU halts code execution • Available in system LP and ULP power modes
CPU deep sleep	• CPU halts code execution. • Requests system deep sleep entry • Available in system LP and ULP power modes
System deep sleep	• Occurs when all CPUs are in CPU deep sleep • CPUs, 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
System hibernate	• CPUs and clocks are OFF • GPIO output states are frozen • Low-power comparator, RTC alarm, and dedicated WAKEUP pins are available to wake up the system • Backup domain is available.

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

Figure 1. PSoC 6 MCU Device Power Mode Transition Diagram

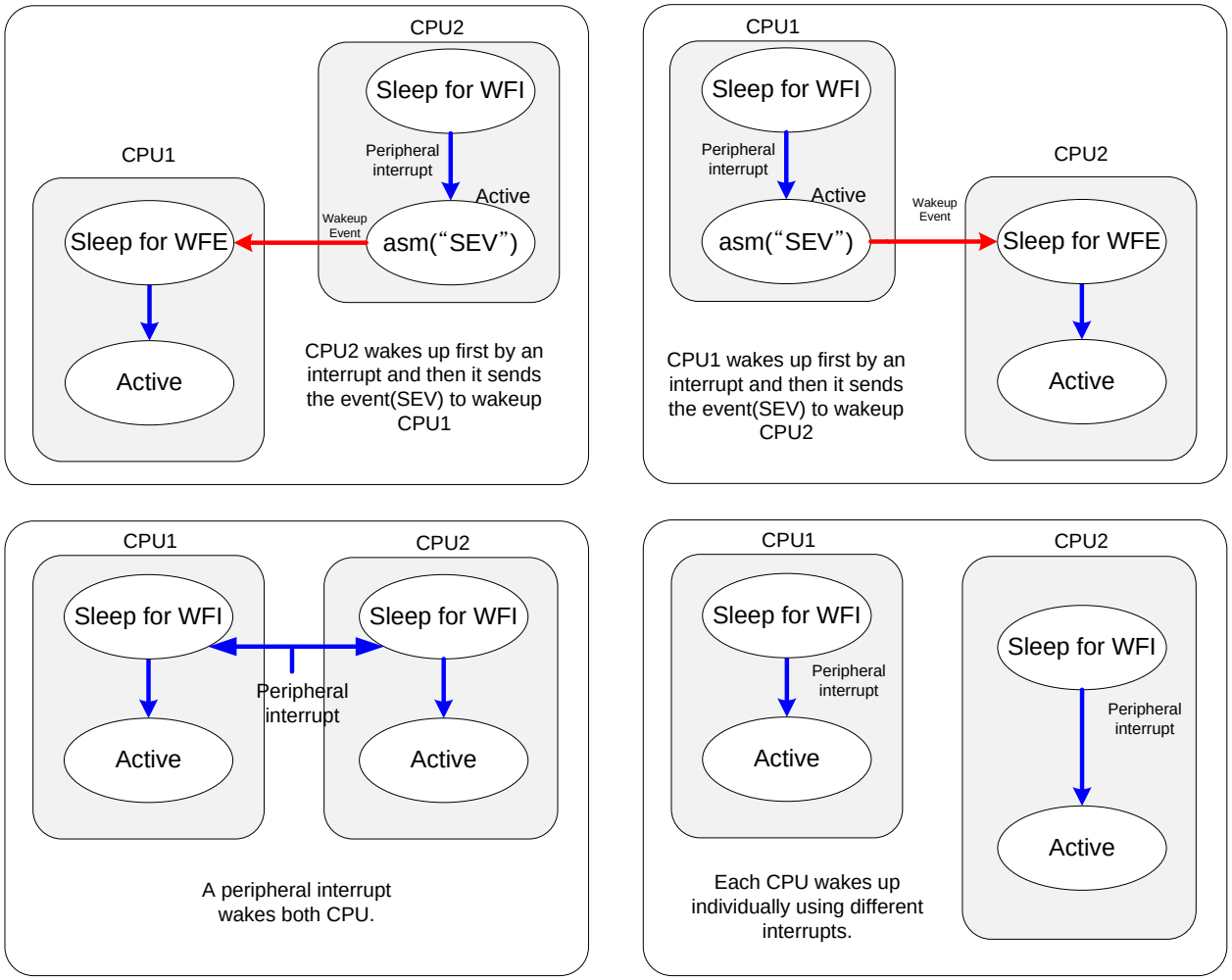


2.2 Core Sleep and Wakeup Instructions

Arm Cortex® CPUs 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 CPU executes `__WFI`, the CPU goes to sleep and stays in sleep until any interrupt is asserted. Wait-for-Event (`__WFE`) is similar to `__WFI`, but it wakes up when the wakeup event is received instead of an interrupt. Set Event (`__SEV`) is used for waking up other CPUs in sleep mode because of a `__WFE`. Deep sleep uses the same instructions for sleep and wakeup, but the SLEEPDEEP bit[2] of the 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 CPU Sleep and Wakeup Cases



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

2.3 Subsystem Availability and Power Consumption

2.3.1 Subsystem Availability

Each subsystem resource works differently in system 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. Because 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 reduces power consumption without performance degradation. Table 2 lists sample case scenarios of power modes. In some examples, only a few power modes are used effectively.

Table 2. Sample Case Scenarios of Power Modes

Power Modes	Wearable Device	Air Conditioner	Remote Controller	Thermometer
System LP CPU active	GUI interaction by user	Motor run	–	Communicates over BLE
System ULP CPU active	Processes heartbeat	–	Sends command	Reads temperature Updates result on LCD
System ULP CPU sleep	–	–	–	–
System ULP CPU sleep	Analog block detects heartbeat	–	–	–
System 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)
System hibernate	Low battery – Does nothing Resets device when charger is plugged in	–	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 registers 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 firmware transitions for power modes. SysPm provides the default five transition functions for CPU sleep, CPU deep sleep, system hibernate, system LP, and system ULP.

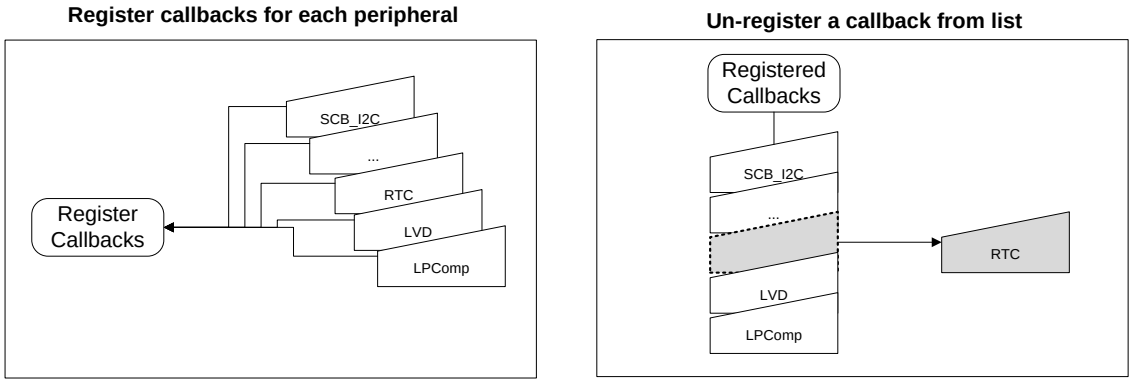
The power mode changing functions provide four different callback operations to execute a necessary action for each peripheral:

- `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 the rollback action.

The SysPm driver 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. You can choose to register the defined peripheral callback or can make a custom callback. The SysPm transition function executes the registered callbacks sequentially. The first registered function is executed first.

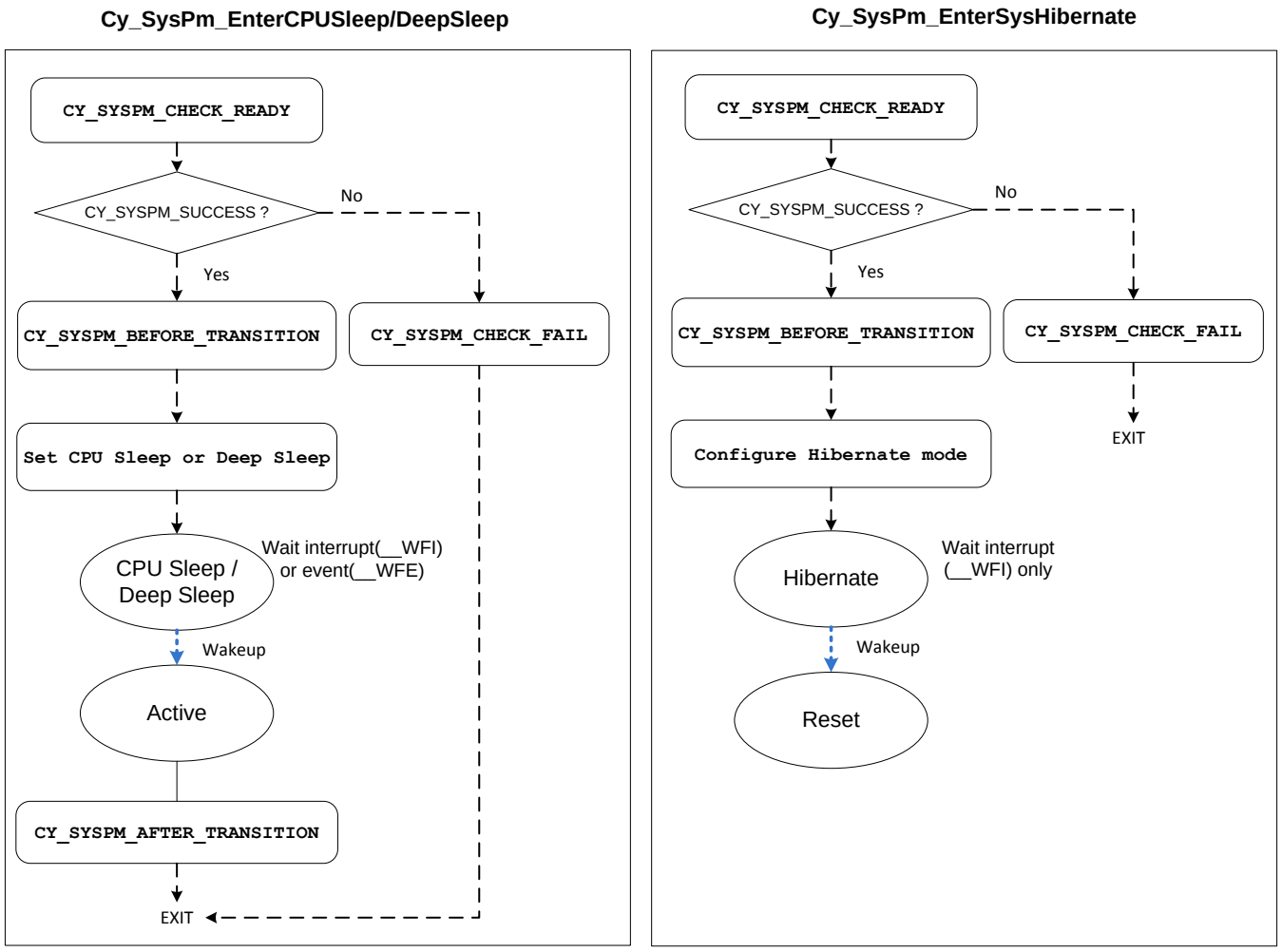
For more information on callback registration and implementation, see Appendix C, and D.1 CE219881 - PSoC 6 MCU Switching between Power Modes, which is the mode transition example for active, LPACTIVE, sleep, LPSLEEP, and deep sleep.

Figure 3. Power Mode Callback Registration and Un-registration



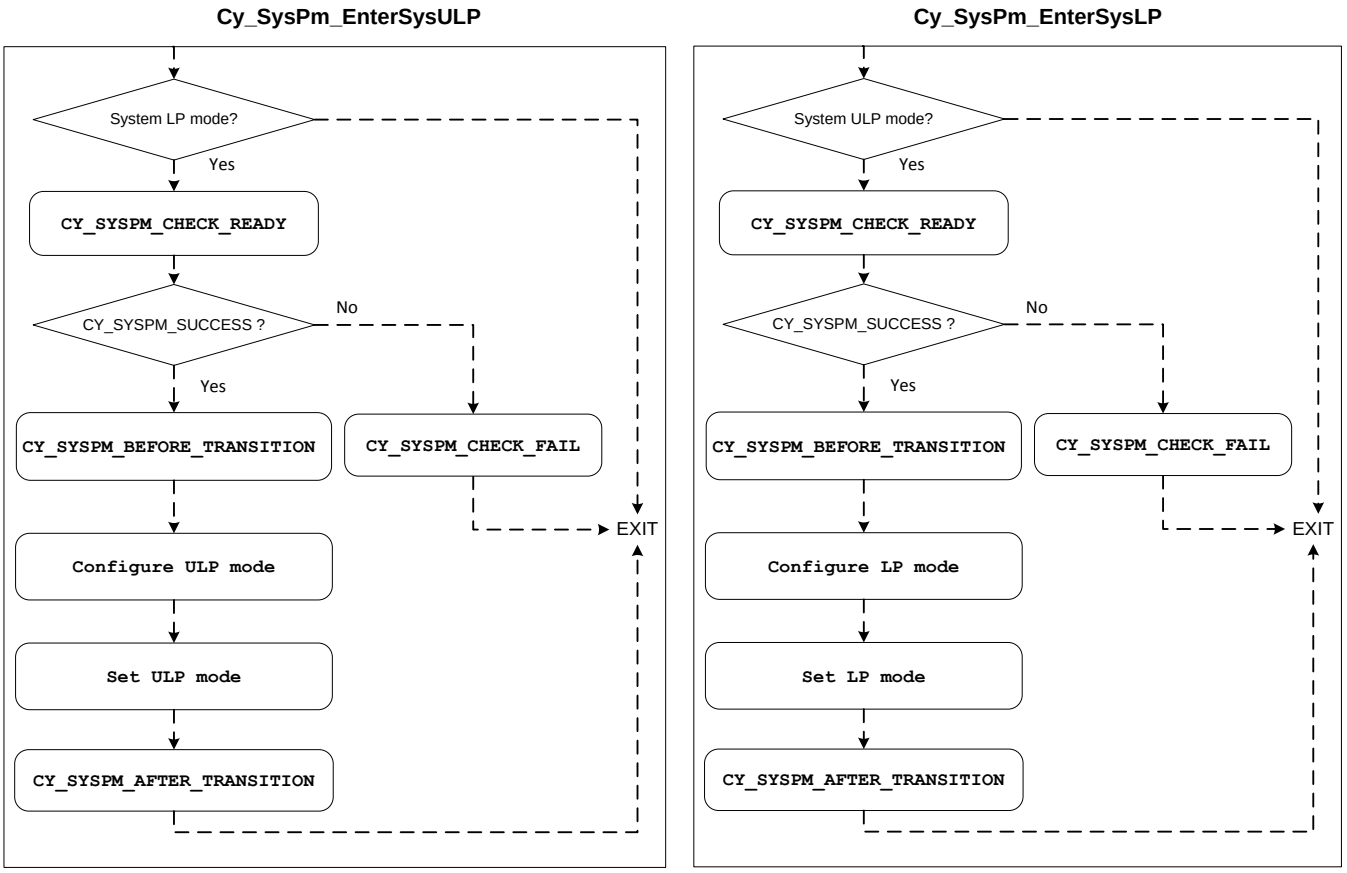
By calling the mode transition function, the device starts to transition with four callback operations. The CPU sleep, and CPU deep sleep modes use Arm sleep instructions. Code execution stops and waits for an interrupt during the CPU sleep power mode. Figure 4 shows the CPU waiting for a wakeup source by calling sleep instruction: `WFI()` or `__WFE()`. So, the actual mode transition to system deep sleep is done after the sleep instruction is executed. After wakeup, the device automatically transitions to CPU active.

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



In the system LP and system ULP modes, all system resources keep running. So, entering LP and ULP mode are done by configuring the power mode control register, and there is no delay or wait for 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 ULP 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), or single input single output (SISO) buck for core power, as listed in Table 3. The LDO can provide up to 300 mA in high-current mode (normal) and 25 mA in low-current (LP) mode. The buck regulator can provide up to 20 mA for one output and 30 mA combined output. The SIMO buck regulator gives a better efficiency under normal load conditions. Once switched to SIMO, it is not possible to switch back to LDO without resetting the device.

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

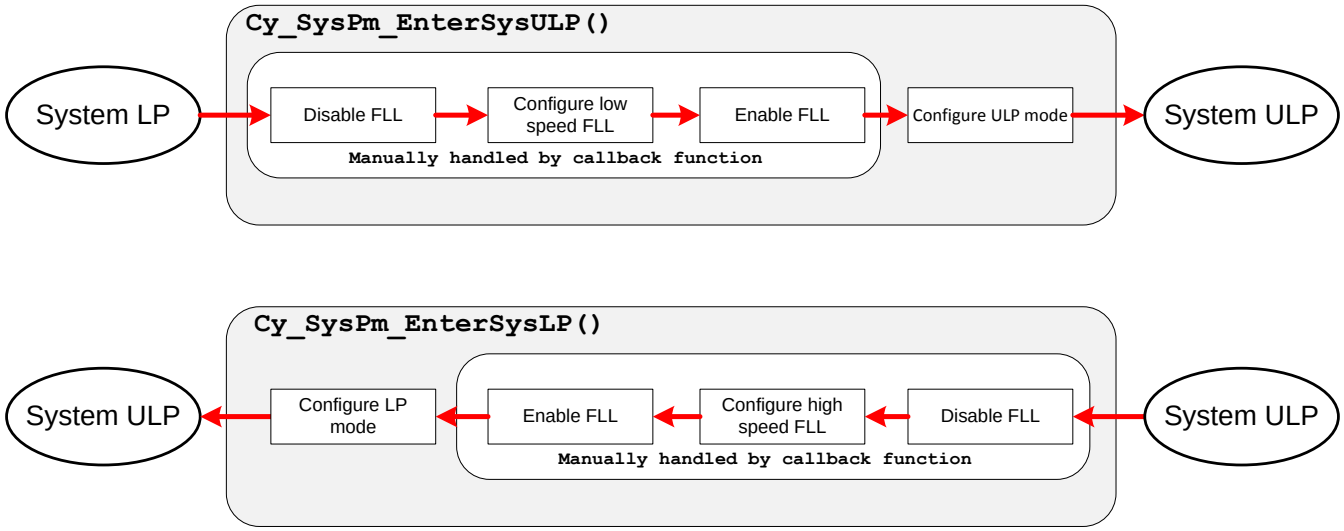
	Output	Max Load	Max Clock Frequency
LDO	0.9 V	25/300 mA	50 MHz for Cortex-M4 (CM4) 25 MHz for Cortex-M0+ (CM0+)
	1.1 V	25/300 mA	Allow maximum supportable clock frequency
Buck	0.9 V	20 mA	50 MHz for CM4 25 MHz for CM0+
	1.1 V	20 mA	Allow maximum supportable clock frequency

3.2 ULP Mode Clock

Transition to ULP 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 ULP mode as described earlier, so the clock configuration should be adjusted based on the regulator output when entering or exiting ULP 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 LP and ULP using PDL functions with the registered callback function. Because of the clock limitation of ULP mode, either the frequency-locked loop (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 their own callbacks to handle the changing frequency. CE219881 – PSoC 6 MCU Switching between Power Modes provides an example of clock adjustment callback.

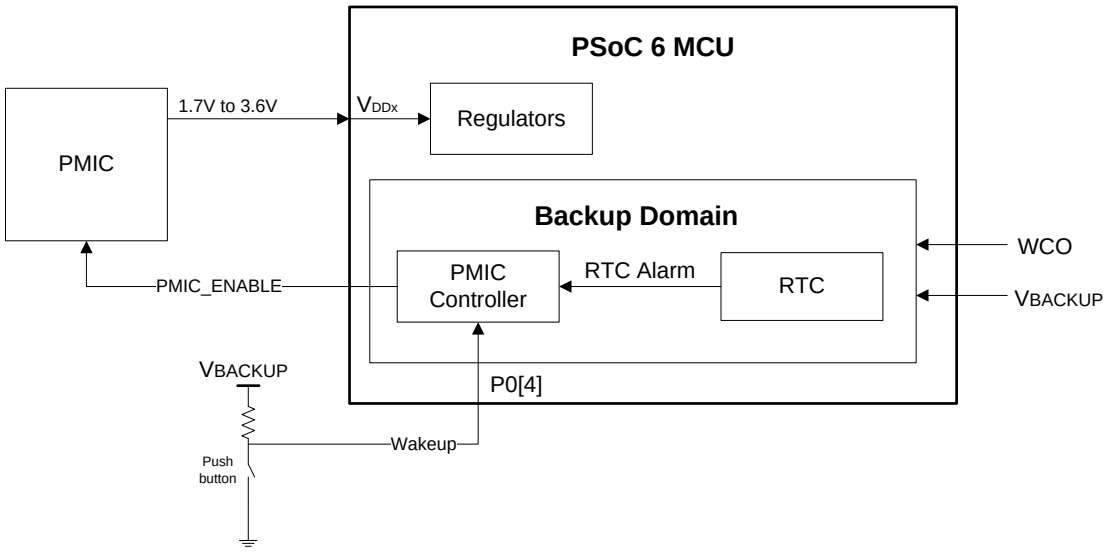
Figure 6. LP Mode Enter/Exit Transition



3.3 External PMIC Control

The PSoC 6 MCU backup power domain 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 the PMIC, but a backup supply to control PMIC keeps the backup domain alive.

Figure 7. An External PMIC Control Block Diagram



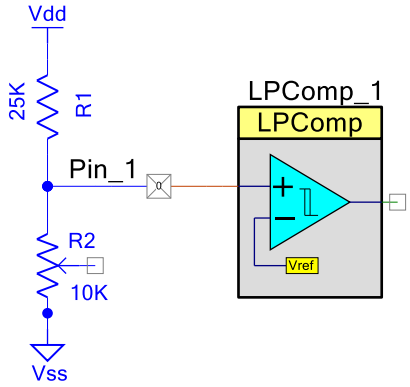
4 Other Power Saving Techniques

4.1 Use PSoC 6 MCU 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.

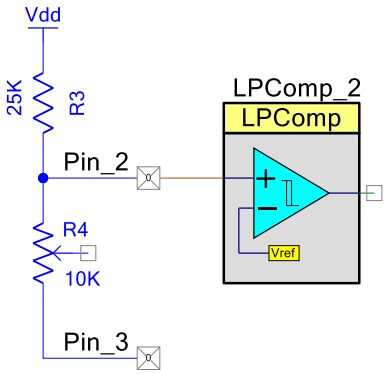
Figure 8. Typical LPComp Application



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

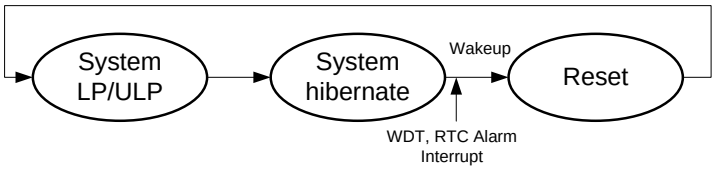
You can save power any time you offload a task from the CPU and either halt the CPU or let it do something else in parallel. The DMA engine that can be used in system LP or ULP modes to transfer data with no CPU use.

4.4 Periodic Wakeup Timers

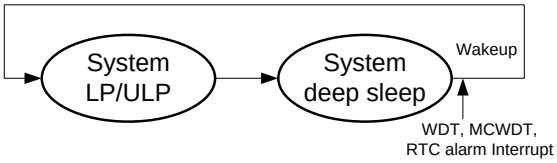
The periodic wakeup from CPU sleep mode is a traditional way to save power consumption. The average power consumption is determined by CPU active period power consumption and CPU sleep period power consumption. To achieve the best result, the sleep period should be as long as possible and the active period should be as short as possible. WDT and multi-counter WDT (MCWDT) can be good periodic wakeup sources in system deep sleep mode. The WDT can also run in system hibernate Mode. If your application needs a longer wakeup period, an RTC alarm can be a good periodic wakeup source. See D.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:

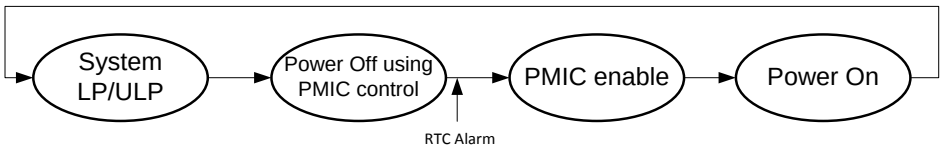
- Active and hibernate



- Active and deep sleep



- Active and Power OFF with an external PMIC

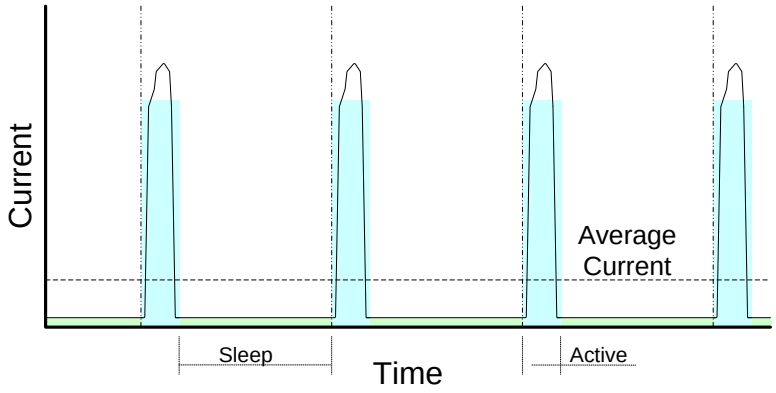


4.5 Clocks

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.

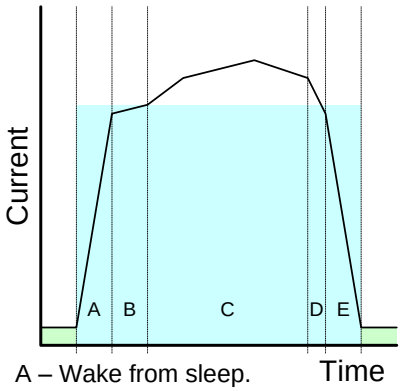
You can use CPU sleep or system deep sleep mode to reduce the power when the PSoC device is idle, but the average current consumption is higher because of the time spent in CPU active mode. Figure 10 is a representation of the current consumption of this example with the system clocks set at 3 MHz.

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 CPU active mode for less time. Figure 11 is a representation of CPU active mode timing, broken up into tasks.

Figure 11. Analysis of Tasks in CPU active Mode at 3 MHz

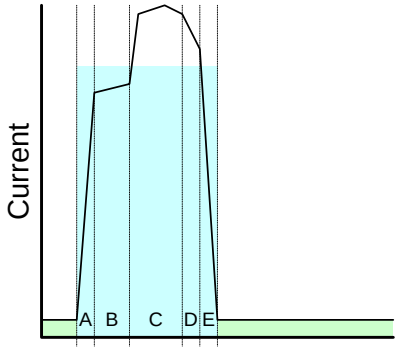


A – Wake from sleep.
B – Read sensor data.
C – Manipulate data.
D – Transmit result.
E – Go back to sleep.

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.

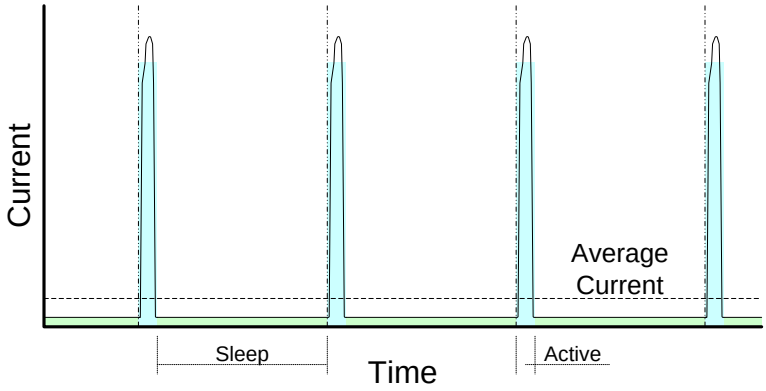
At some point, the benefit of a shorter active time is overcome by the energy required to drive the clocks at a higher rate. Assume that the optimal speed is 12 MHz, as Figure 12 shows. With a 12-MHz clock, the time spent in Active mode is about half the time spent with a 3-MHz clock. Figure 13 shows 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 external circuitry when the PSoC device is in all low-power modes. 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 system hibernate mode, the GPIO drive modes and data registers are automatically "frozen." They should be reconfigured to a known state before being "unfrozen" after a wakeup reset to allow their states to change by calling `Cy_SysPm_IoUnfreeze()`. See Section D.2 CE218129 - PSoC 6 MCU Wakeup from Hibernate Using a Low-Power Comparator for a code example on hibernate and I/O 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-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 Modes Summary

A.1 Power Modes and Wakeup Source

Table 4. Power Modes and Wakeup Source

System Power Mode	MCU Power Mode	Description	Entry Conditions	Wakeup Sources	Wakeup Action
LP	Active	Primary mode of operation. 1.1-V core voltage. All peripherals are available (programmable). Clocks at their maximum frequencies.	Reset from external reset, brownout, power on reset system and hibernate mode. Manual register write from system ULP mode. Wakeup from CPU sleep or CPU deep sleep while in system LP mode. Wakeup from system deep sleep after entered from LP mode.	Not applicable	N/A
	Sleep	1.1-V core voltage. One or more CPUs in sleep mode (execution halted). All peripherals are available (programmable). Clocks at their maximum frequencies.	In system LP mode, CPU executes WFI/WFE instruction with deep sleep disabled	Any interrupt to CPU	Interrupt
	Deep sleep	1.1-V core voltage. One CPU in deep sleep mode (execution halted). Other CPU in active or sleep mode. All peripherals are available (programmable). Clocks at their maximum frequencies.	In system LP mode, CPU executes WFI/WFE instruction with deep sleep enabled	Any interrupt to CPU	Interrupt
ULP	Active	0.9-V core voltage. All peripherals are available (programmable). Clock frequencies are limited.	Manual register write from system LP mode. Wakeup from CPU sleep or CPU deep sleep while in system ULP mode. Wakeup from system deep sleep after entered from ULP mode.	Not applicable	N/A
	Sleep	0.9-V core voltage. One or more CPUs in sleep mode (execution halted). All peripherals are available (programmable). Clock frequencies are limited.	In system ULP mode, CPU executes WFI/WFE instruction with deep sleep disabled	Any interrupt to CPU	Interrupt
	Deep sleep	0.9-V core voltage. One CPU in deep sleep mode (execution halted). Other CPU in active or sleep mode. All peripherals are available (programmable). Clock frequencies are limited.	In system ULP mode, CPU executes WFI/WFE instruction with deep sleep enabled	Any interrupt to CPU	Interrupt
Deep sleep	Deep sleep	All high-frequency clocks and peripherals are turned off. Low-frequency clock (32 kHz) and low-power analog and digital peripherals are available for operation and as wakeup sources. SRAM is retained (programmable).	Both CPUs simultaneously in CPU deep sleep mode	GPIO interrupt, low-power comparator, SCB, CTBm, watchdog timer, and RTC alarms	Interrupt
Hibernate	N/A	GPIO states are frozen. All peripherals and clocks in the device are completely turned OFF except low-power comparator and backup domain. Wakeup is possible through WAKEUP pins, XRES, low-power comparator (programmable), and RTC alarms (programmable). Device resets on wakeup.	Manual register write from LP or ULP modes	WAKEUP pin, low-power comparator, watchdog timer[b], and RTC[a] alarms	Reset

a. RTC (along with WCO) is a part of the backup domain and is available irrespective of the device power mode. RTC alarms are capable of waking up the device from any power mode.

b. Watchdog timer is capable of generating a hibernate wakeup.

Appendix B. Subsystem Availability

B.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	System Power Modes							
	LP		ULP		Deep Sleep	Hibernate	XRES	Power Off with Backup
	CPU Active	CPU Sleep/ Deep Sleep	CPU Active	CPU Sleep/ Deep Sleep				
Core functions								
CPU	On	Sleep	On	Sleep	Retention	Off	Off	Off
SRAM	On	On	On	On	Retention	Off	Off	Off
Flash	Read/Write	Read/Write	Read only	Read only	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
Peripherals								
SMIF	On	On	On	On	Retention	Off	Off	Off
UDB	On	On	On	On	Off	Off	Off	Off
SAR ADC	On	On	On	On	Off	Off	Off	Off
CTBm	On	On	On	On	On (lower GBW)[1]	Off	Off	Off
LPCMP	On	On	On	On	On[1]	On[2]	Off	Off
TCPWM	On	On	On	On	Off	Off	Off	Off
CSD	On	On	On	On	Retention	Off	Off	Off
BLE	On	On	On	On	Retention	Off	Off	Off
LCD	On	On	On	On	On	Off	Off	Off
SCB	On	On	On	On	Retention (I^2C/SPI wakeup available)[3]	Off	Off	Off
GPIO	On	On	On	On	On	Freeze	Off	Off
Watchdog timer	On	On	On	On	On	On	Off	Off
Multi-Counter WDT	On	On	On	On	On	Off	Off	Off
Resets								
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	On[4]	Off	Off
Backup domain								
WCO, RTC, alarms	On	On	On	On	On	On	On	On

1. Low-power comparator and CTBm may be optionally enabled in system deep sleep mode to generate wakeup.
2. Low-power comparator may be optionally enabled in hibernate mode to generate wakeup.
3. Only one SCB with deep sleep support is available in the deep sleep power mode; other SCBs are not available in the deep sleep power mode.
4. Watchdog interrupt can generate a hibernate wakeup. See the “Watchdog Timer” chapter of the TRM for details.

Appendix C. Callback Function Example

C.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);
```

C.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 D. Code Examples

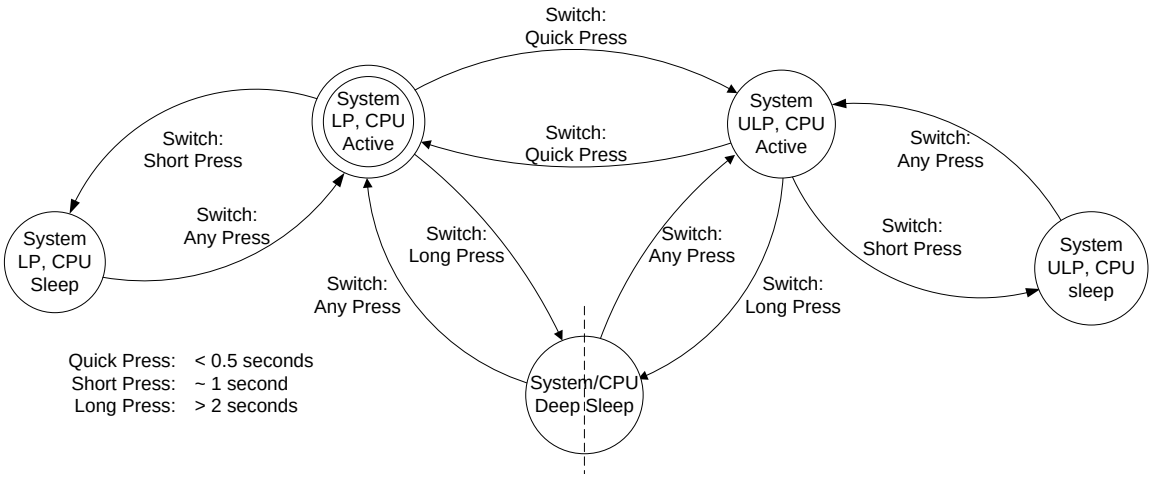
The following code example is included with this application note to demonstrate PSoC 6 MCU power modes and power reduction techniques.

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

This code example shows how to enter and exit system LP and ULP power modes, and transition the CPU from CPU active to sleep or deep sleep. Once in either mode, the example also shows how to wake up and return to one of the system LP/ULP and CPU active modes.

The project uses a button switch to transition among 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.

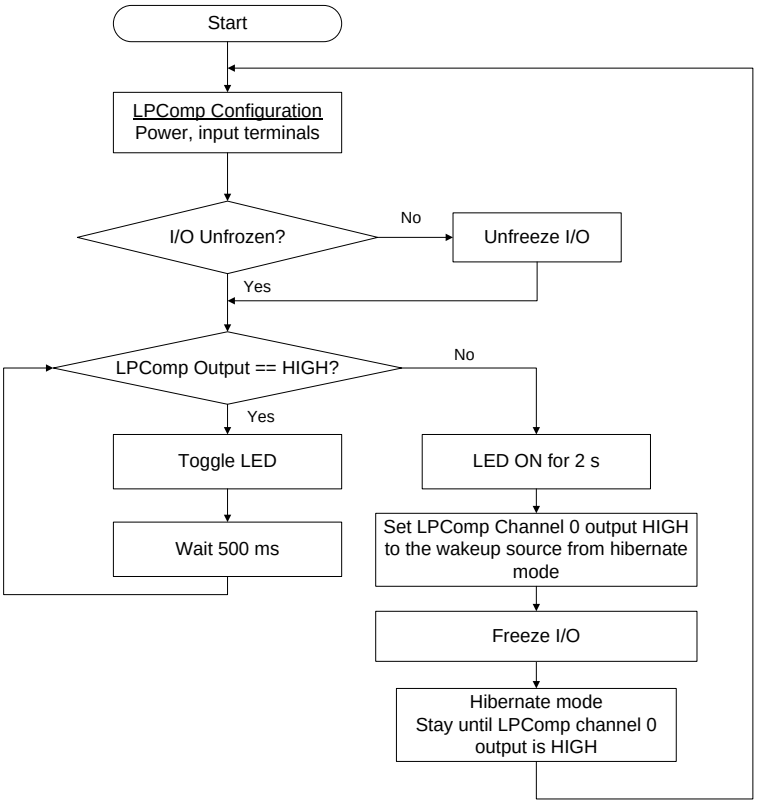
Figure 14. Power Mode State Machine



D.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 system hibernate power mode transition. It teaches you how to handle the GPIOs before and after system 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.

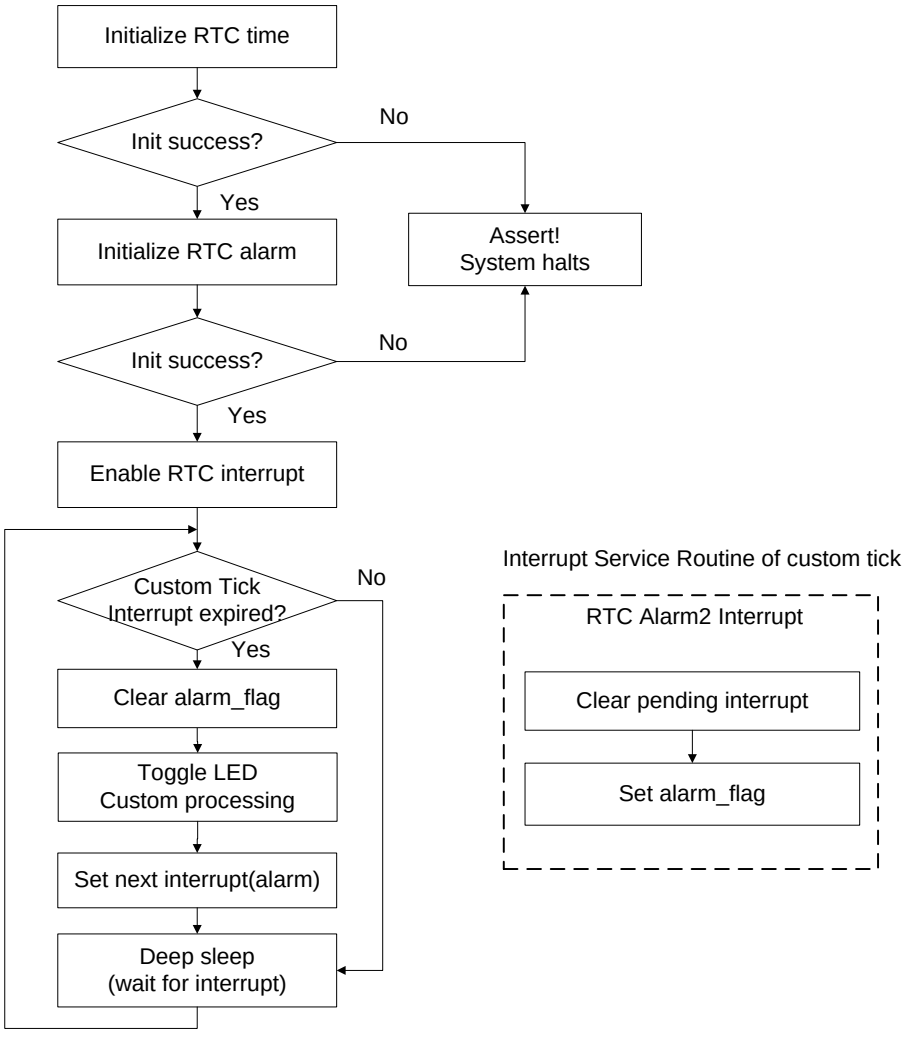
Figure 15. Wakeup from system hibernate Mode Using LPComp Input



D.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 system LP and deep sleep modes 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.
*A	6166204	GJV	05/05/2018	Updated power mode names to match TRM names; CPU active, CPU, sleep, CPU deep sleep, system LP, system ULP, system deep sleep, system hibernate.

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 MCU

Cypress Developer Community

Community | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

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