PSoC 4 Low-Power CapSense Design

Authors: Vijay K M

Associated Project: Yes

Associated Part Family: PSoC® 4 Series

Software Version: PSoC Creator™ 4.2

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

More code examples? We heard you.

To access an ever-growing list of hundreds of code examples, using either ModusToolbox™ IDE or PSoC® Creator™, visit our GitHub repository. You can also explore the Cypress video training library here.

AN210998 teaches you how to design low-power CapSense® applications with PSoC 4 devices. It also explains how to compute the power consumption for a CapSense application and provides hardware, firmware, and system-level guidelines to minimize power consumption.

Contents

1 Introduction

Designs based on capacitive sensing are increasingly used in a wide variety of battery-powered and portable electronic appliances. Power consumption is a key parameter in the design of these portable appliances, where battery life is of primary importance. This application note explains how to design low-power CapSense applications with PSoC 4 devices. In doing so, it discusses the factors that affect power consumption in a CapSense system and covers the hardware, firmware, and system-level considerations for a low-power design.

This document assumes that you are familiar with the PSoC 4 architecture, PSoC 4 CapSense, and application development for PSoC 4 using the PSoC Creator™ Integrated Design Environment (IDE). For an introduction to PSoC 4, read AN79953 – Getting Started with PSoC 4. If you are new to PSoC 4 CapSense, refer to AN64846 – Getting Started with CapSense and AN85951 – PSoC 4 CapSense Design Guide. If you are new to PSoC Creator, see the PSoC Creator home page.

The initial sections of this application note cover the factors affecting power consumption. Later sections cover the hardware, firmware, and system-level considerations for low-power design. Finally, an example project is described, which demonstrates how to achieve a low-power CapSense design.

2 Introduction to Low-Power Design

This section explains the importance of the power budget requirement in the system definition. It then gives an overview of the factors that affect power consumption in a CapSense controller such as active time, sleep time, operating voltage, operating frequency, peripheral, and GPIO activity. This discussion is divided into factors that influence active time and those that influence instantaneous current.

2.1 Power Budget Requirement

The power budget requirement is one of the main ingredients in the system definition of a low-power capacitive sensing system. The initial definition of target power consumption provides the system designer with a starting point for hardware and firmware development.

The power budget can be derived from battery capacity and battery life. Selection of a battery capacity depends on physical form factor, cost, and size. To determine the power budget, the designer needs to know the battery life, that is, how long the system is required to operate without replacing batteries or recharging them. The battery life requirement varies based on the application. For some applications such as mobile/portable gadgets, the battery life requirement could be specified in days. For other systems that are sealed or placed in remote areas, the required battery life may be specified in months or years. In such systems, battery replacement would be very difficult, placing a stringent requirement on the power budget. Once the required battery life and the largest battery capacity (that can be used in the system) are established, an average current can be calculated. See the datasheet of the selected battery for information on battery capacity.

The average current (power budget) for a battery-powered embedded system can be calculated using the following equation:

$$\text{Maximum Average Current} = \frac{Battery\ Capacity[mA{-}H]}{Required\ Battery\ Life\ [H]}$$

The system tasks should be designed such that the total average current of the system is less than the maximum average current specified by the power budget. Average power consumption is important in battery-powered applications to achieve long battery life.

2.2 Factors Affecting Average Current

The main factors that determine the power consumption of the CapSense system is the time the device spends in active mode versus inactive modes and the current (in active and inactive modes). In active mode, the CapSense hardware scans the sensor to detect any human interaction; in inactive modes, the device is put into sleep. In typical applications, the CapSense controller does not need to be in active mode all the time. The sensors are scanned periodically by the CapSense hardware. When the sensor is scanned, the device is in the Active mode. When the sensor is not being scanned, the device is in the inactive mode.

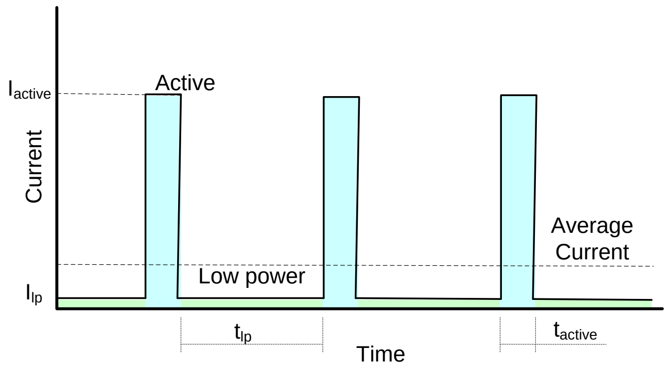
PSoC 4 supports low-power modes to reduce the device average power consumption. To reduce the average power, the device should be put in the lowest-power mode (deep-sleep) during the inactive mode. Some of the PSoC 4 devices support hibernate and stop modes that consume less power than deep-sleep mode. However, these modes cause a software reset when woken up by an interrupt, which requires special handling when CapSense is used. Therefore, it is recommended to use deep-sleep mode for most of the systems.

Since the CapSense hardware is disabled in deep-sleep mode, the device must wake up periodically to scan for human interaction. You can use the watchdog timer (WDT) in PSoC 4 to wake up the device from deep-sleep mode at periodic intervals.

Even during the CapSense scan (active mode), the device can be put into sleep mode, as CPU intervention is not required between the start and end of the scan. If the firmware does not have any additional tasks apart from waiting for the scan to finish, you can put the device into Sleep mode after initiating a scan to save power. When the CapSense hardware completes the scan, it generates an interrupt to wake up the CPU from sleep mode and return to active mode.

A system optimized for low average power spends most of the time in the lower power modes, while ensuring reliable operation. Figure 1 and Equation 1 illustrate the relationship between average current and active time.

Figure 1. Timing Diagram of a Typical Low-Power System



If the device periodically wakes up from a low-power mode and spends a fixed amount of time in the Active mode, you can calculate the approximate average current consumed by the device as follows:

$$\text{Average current } (I_{AVE}) = \frac{((t_{active} * I_{active}) + (t_{lp} * I_{lp}))}{t_{active} + t_{lp}} \qquad \text{Equation 1}$$

t_{active} = Time spent in Active mode

I_{active} = Active mode current

t_{lp} = Time spent in low-power mode

I_{lp} = Low-power mode current

The average power consumed by the device can be calculated as follows:

$$P_{AVE} = V_{DD} \times I_{AVE} \qquad \text{Equation 2}$$

As shown in Equation 2, the power consumption depends on the operating voltage and average current. Operating voltage is a major influencing factor for power consumption. Power consumption is directly proportional to operating voltage. Therefore, you should consider reducing the device operating voltage if possible.

As shown in Equation 1, the average current consumption can be reduced by reducing the active time and the current (in Active and low-power modes).

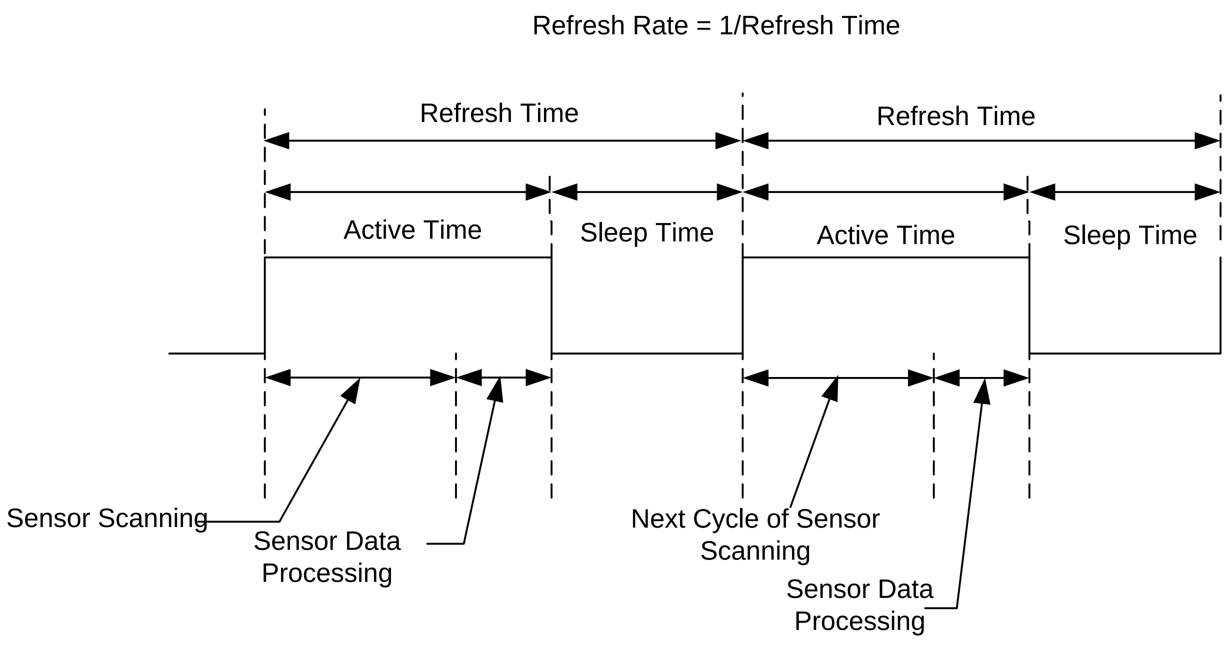
2.2.1 Factors Affecting Active Time

The following are some factors that affect the active time and therefore the overall power consumption.

- **Number of sensors:** The total scan time of sensors increases with an increase in the number of sensors. For a fixed refresh rate, if the sensor scan time increases, the duration in which the device is in sleep mode decreases. This results in increased power consumption by the device.

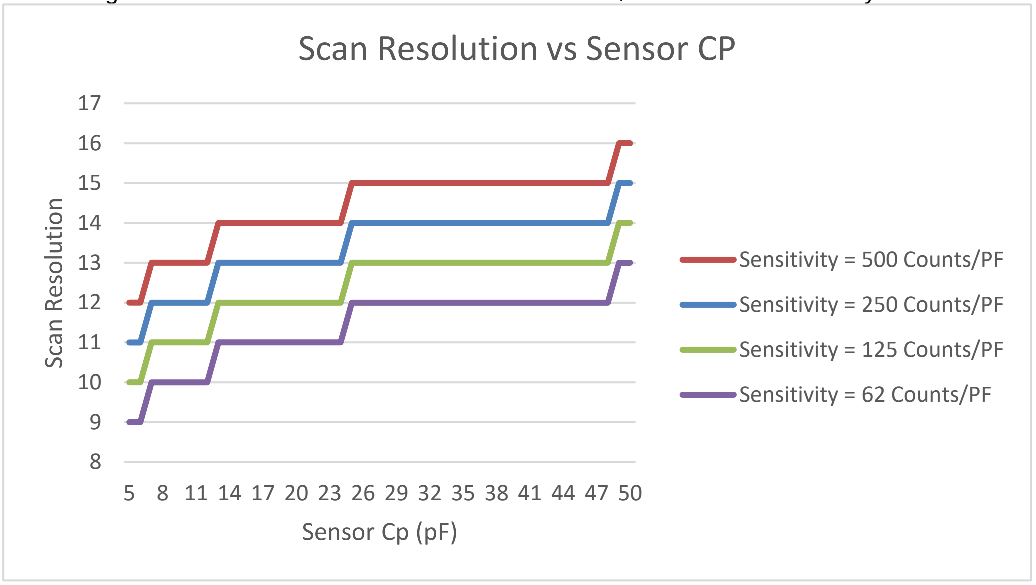
- **Refresh rate:** The refresh rate is the number of active-sleep cycles per second. This parameter affects the duration for which the device is in Sleep mode for a fixed active time, as shown in Figure 2. The average power consumption can be reduced by decreasing I_{AVE}, as shown in Equation 1. I_{AVE} can be decreased by increasing the sleep time. The lower the refresh rate, the lower the power consumption. However, a low refresh rate reduces the response time. Setting the refresh rate too long could cause the end-user experience to be poor. Therefore, there is a tradeoff between response time and power consumption. A general rule of thumb is to keep the refresh rate to a minimum of 50 Hz to detect fast finger taps and provide acceptable end user experience. The refresh rate can be further reduced by dynamically changing the refresh rate depending on whether the user is touching the sensor or is away for a longer time. In this case, the sensor is scanned at slower refresh rate when the user is not near the sensor and once the user touches the sensor, the refresh rate is increased to detect fast finger taps. The Example Project demonstrates how a user can optimize response time and reduce power consumption.

Figure 2. Relation between Refresh Rate and Sleep Time



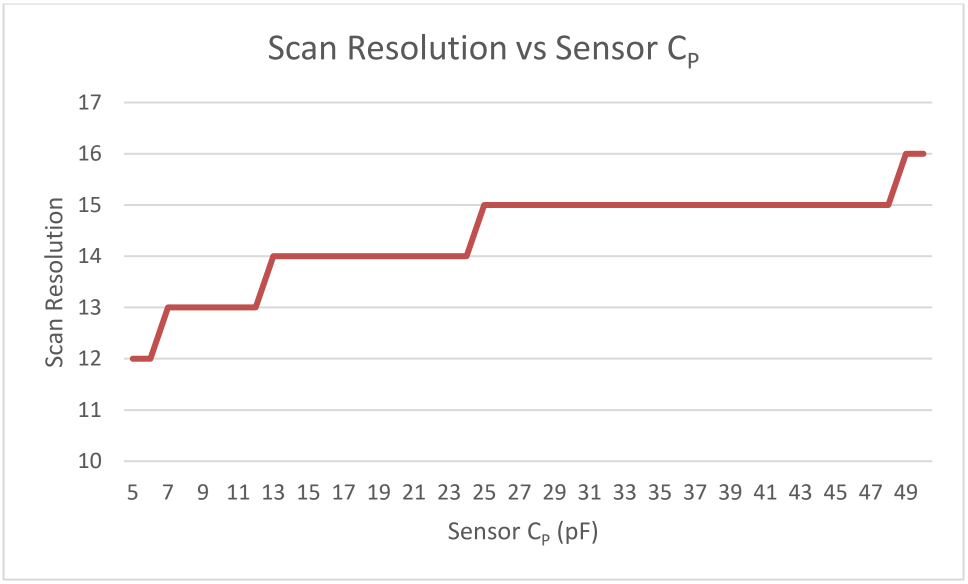
- **Scan Resolution/Sensitivity:** Scan resolution is the resolution of the raw count counter. The sensitivity parameter is similar to the scan resolution parameter and is defined as signal counts achieved per 1 pF of finger capacitance (C_F). These factors are explained in detail in the Firmware Design Considerations section.

Figure 3. Sensor Scan resolution versus Sensor C_P for Different Sensitivity values



- **Parasitic capacitance (C_P) of sensor:** The C_P of the sensor directly affects the scan time of the sensor. Figure 4 shows the effect of sensor C_P on the sensor scan resolution parameter. With a larger C_P value, the sensor has to be scanned at a higher scan resolution to achieve the required signal when compared to a sensor with a lower C_P. A higher scan resolution results in a longer sensor scan time (for a fixed modulator clock), which results in increased active time and increased average power consumption. Therefore, C_P of the sensor should be minimized by following the layout best practices. The Hardware Design Considerations for Low Power section in this application note provides guidelines to minimize the sensor C_P.

Figure 4. Sensor Scan Resolution versus Sensor C_P for 500 Count/pF Sensitivity



- **Proximity distance:** Proximity sensors requires scanning the sensor at a high scan resolution (15 or 16 bits) to achieve a large proximity-sensing distance. A high scan resolution results in a longer scan time and increases the device active time, which leads to a higher power consumption. Therefore, a larger proximity distance requires higher power consumption.

2.2.2 Factors Affecting Active and Low-Power Mode Current

Average current depends on the instantaneous current consumption in the Active and low-power modes. The following factors influence the instantaneous currents in these modes:

- **Operating frequency:** The power consumption of the device in active mode is directly proportionally to the operating frequency. Since the CPU processes the CapSense data after the scan is completed, higher operating frequency completes the processing in a short duration. However, for systems that do not require fast data processing, the CPU frequency can be reduced to a lower value to reduce the power consumption during the CPU active time. Therefore there is a trade-off between active time and active current across different frequencies.
- **Unused peripherals:** Power ON the peripherals only when they are needed by the application. Keep them in low-power states or OFF when they are not needed.
- **Unused GPIOs:** Unused GPIOs should be configured for a defined level (input with internal pull-up), and floating inputs should be avoided to reduce power consumption.
- **Host communication:** In some systems, the host communicates with the CapSense controller using a serial protocol (I^2C, SPI, or UART). In a typical CapSense application, the CapSense device enters Sleep mode after the sensor scanning is completed. The device will not enter Sleep mode or it will exit from Deep-Sleep mode if there are any transactions with the host processor (over I^2C/SPI/UART). Therefore, frequent transactions increases the average power consumption. To reduce it, implement a dedicated host interrupt pin and initiate the transaction only when the CapSense controller sends a host interrupt pulse to the host.

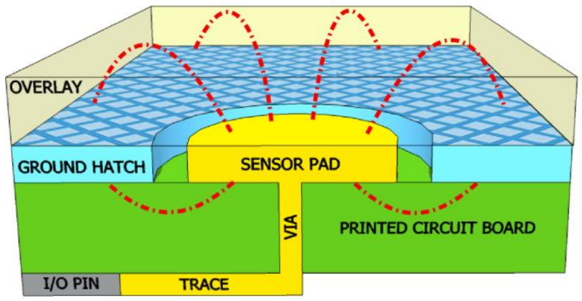
3 Hardware Design Considerations for Low Power

In a CapSense application, capacitive sensors are formed by the traces of a printed circuit board (PCB) or flex circuit. A good hardware design ensures that the design is robust and helps in achieving low average power consumption. Poorly designed CapSense hardware can lead to increased power consumption that cannot be fully compensated by firmware algorithm or sensor tuning. The following sections explain the hardware design considerations to achieve low average power.

3.1 Parasitic Capacitance

Sensor parasitic capacitance needs to be reduced to reduce power consumption. In a CapSense self-capacitance system (CSD), the sensor capacitance measured by the controller is called “Cs.” When a finger is not on the sensor, Cs equals the parasitic capacitance (C_P) of the system. This parasitic capacitance is a simplification of the distributed capacitance that includes the effects of the sensor pad, the overlay, the trace between the CapSense controller pin and the sensor pad, the vias through the circuit board, and the pin capacitance of the CapSense controller, as shown in Figure 5. The main components of C_P are therefore the trace capacitance and the sensor capacitance. C_P is related to the electric field around the sensor pad. Increased C_P increases power consumption and reduces sensitivity.

Figure 5. C_P and Electric Field



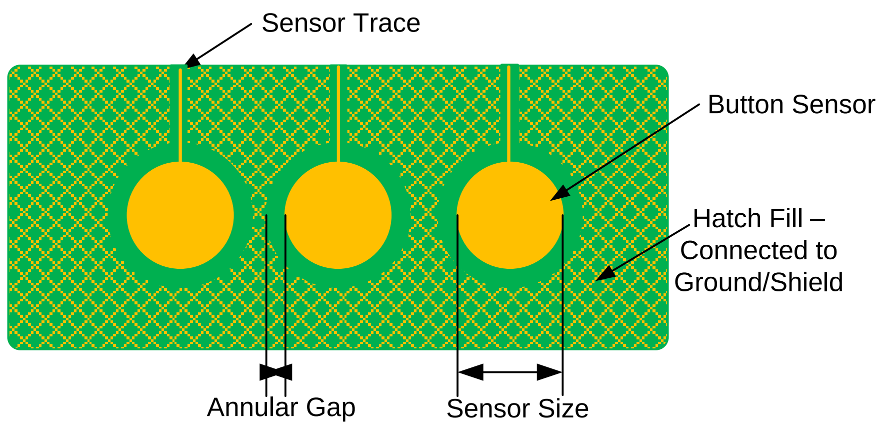
When a finger touches the sensor surface, it forms a simple parallel plate capacitor with the sensor pad through the overlay. The resulting capacitance is called “finger capacitance, Cf.” Cf is a simplification of a distributed capacitance that includes the effects of the human body and the return path to the circuit board ground.

The capacitive sensing controller converts the capacitance measured at its input pin to digital counts using the CSD method. The controller uses the digital counts to identify increases in sensor capacitance due to a finger touch.

To accurately detect a finger touch, the scan resolution parameter in CSD has to be tuned to maintain a specific sensitivity level. If the C_P is high, the scan resolution of the sensor needs to be increased in order to achieve a reliable sensitivity. A higher resolution results in a longer conversion time (scan time) as shown in Equation 3, which increases the average power consumption of a capacitive sensing application. To reduce power consumption, reduce the sensor(s) C_p so that you can set a lower scan resolution.

- **Sensor size:** The smaller the sensor size (Figure 6), the lower the C_P. Lower C_P requires smaller scan resolution to achieve the required signal and hence results in lower average power consumption. However, if the sensor size is too small for the overlay thickness and sensitivity parameter, a finger touch might not be detected. Use an optimum sensor size based on the overlay thickness requirement and sensitivity parameter settings. Refer to AN85951 – PSoC 4 CapSense Design Guide.

Figure 6. CapSense Sensor Layout



- **Trace:** The trace length and width adds to sensor C_P and should be minimized. Placing the chip closer to the sensor minimizes the trace length and hence helps to reduce C_P.
- **Annular gap:** An increase in the air gap between the sensor and ground decreases noise immunity. So there has to be a tradeoff between power consumption and noise immunity (based on system requirements) to achieve an optimum C_P. Refer to Getting Started with CapSense for details on the effect of annular gap on sensor CP and sensitivity.
- **Hatch ground fill:** Use a hatched ground plane instead of a solid ground plane to reduce the C_P. It is recommended that you use hatched ground planes surrounding the sensor and on the bottom layer of the PCB, below the sensors.
- **Shield:** To reduce the sensor C_P, drive the hatch fill in the top and bottom layer with a driven shield signal. Use a hatch fill of 0.17-mm (7-mil) trace and 1.143-mm (45–mil) grid in the top layer and a hatch fill of 0.17-mm (7-mil) trace and 1.778-mm (70–mil) grid in the bottom layer and drive it with a driven shield signal.
- For the proximity sensor, the C_P of the sensor is minimized by using a loop sensor.
- In the FPC, to minimize C_P, have a hatch fill only in the top layer. The hatch fill can be connected to either ground or a driven shield signal.

3.2 Overlay

Overlay is a nonconductive material that serves as the touch surface for the CapSense sensors. Overlay material and thickness determine the finger capacitance, C_F. The finger capacitance is directly proportional to the dielectric constant of the overlay material and inversely proportional to the overlay thickness, as shown in Equation 4. If the finger capacitance is low, the signal is low. To compensate for that, the resolution needs to be increased, which increases the scan time and thus the power consumption.

$$C_F = \frac{\varepsilon_0 \, \varepsilon_r \, A}{d} \qquad \text{Equation 3}$$

Therefore, for low power, decrease the overlay thickness and use an overlay material with a high dielectric constant. Dielectric constants of some common overlay materials are listed in Table 1. Materials with dielectrics between 2.0 and 8.0 are well suited to capacitive sensing applications.

Table 1. Dielectric Constants of Common Materials

Material	ε_r
Air	1.0
Formica®	4.6–4.9
Glass (Standard)	7.6–8.0
Glass (Ceramic)	6.0
PET Film (Mylar®)	3.2
Polycarbonate (Lexan®)	2.9–3.0
Acrylic (Plexiglass®)	2.8
ABS	2.4–4.1
Wood Table and Desktop	1.2–2.5
Gypsum (Drywall)	2.5–6.0

4 Firmware Design Considerations

This section explains the firmware techniques to reduce power consumption while still ensuring reliable operation. The main intent is to optimize the tasks such that the device spends most of the time in the lower power modes. The PSoC 4 low-power modes allow you to reduce overall power consumption while retaining essential functionality.

4.1 Manual Tuning Versus Auto Tuning

The CapSense CSD method is a combination of hardware and firmware techniques. Therefore, it has several hardware and firmware parameters that are required for proper operation. These parameters should be tuned to achieve an optimum performance. Most of the capacitive touch solutions in the market must be manually tuned. Cypress provides a unique feature called SmartSense (also known as Auto-tuning) for PSoC 4 CapSense.

SmartSense is a firmware algorithm that automatically sets all parameters to optimum values. SmartSense auto-tuning reduces design cycle time and provides a stable performance across PCB variations, but requires additional RAM and CPU resources to allow run-time tuning of CapSense parameters.

Manual tuning requires some effort to tune optimum CapSense parameters, but allows strict control over characteristics of Capacitive sensing system, like response time and power consumption. Manual tuning allows strict control over the sensor scan-time, which can help the application engineer tune for the lowest power possible.

The power consumption between various tuning modes increases in the following order:

Manual Tuning ≤ SmartSense (Hardware parameters) < SmartSense (Full Auto-Tune)

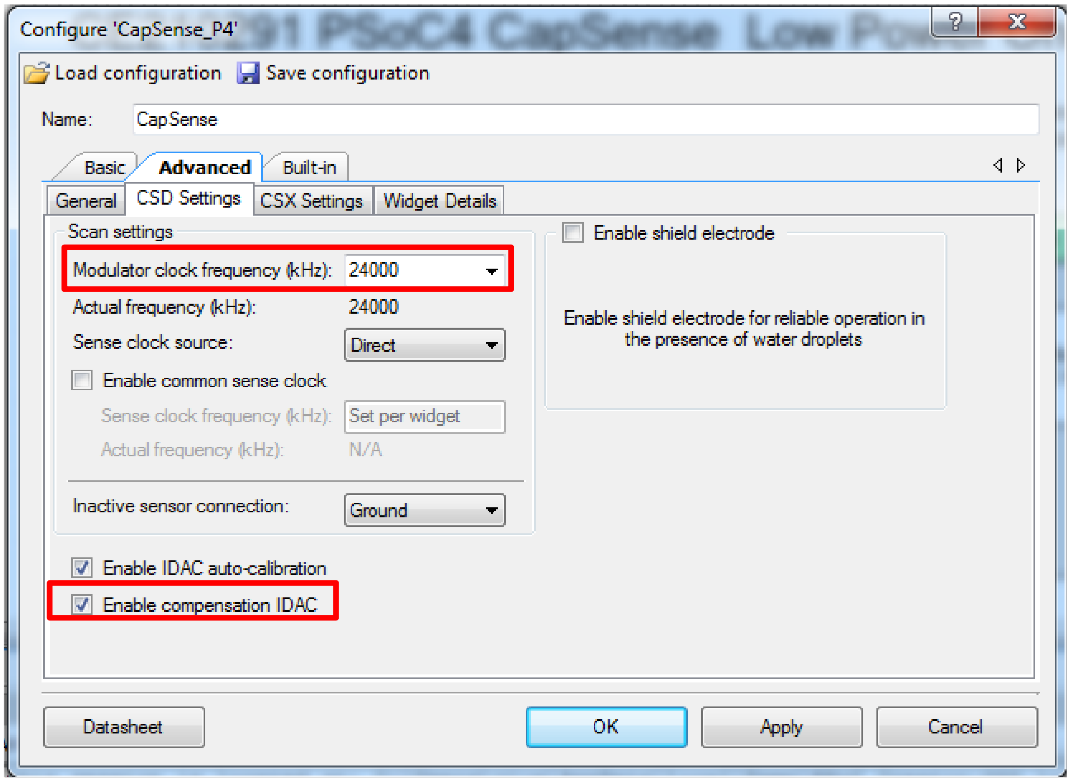
4.2 CapSense Component Settings for Low Power

The designer can tune the configuration parameters in the CapSense Component to reduce the average power consumption. By configuring the Component parameters to optimum values, one can reduce the scan time, which directly lowers the time spent in Active mode, increasing Sleep time and lowering power consumption. In the CapSense Component three key parameters can be configured to reduce average power consumption namely, Compensation IDAC, Modulator frequency, and Scan resolution.

Enable Compensation IDAC: Enabling the compensation IDAC reduces the power as the system can use lesser resolution to achieve the same signal. A lower resolution leads to reduced scan time and to decreased power consumption. The Compensation IDAC is available the Advanced Tab/CSD Settings as shown in Figure 7. It is recommended to enable Compensation IDAC as long as it is not needed for general-purpose DAC requirements and the sensor parasitic is higher than approximately 20 pF.

Modulator frequency: Higher Modulator frequency leads to lower scan time, therefore, reducing power. This parameter is available in the Advanced Tab/CSD Settings as shown in Figure 7.

Figure 7. Advanced Tab/CSD Settings in CapSense Component



Name: CapSense

Modulator clock frequency (kHz): 24000

Actual frequency (kHz): 24000

Sense clock source: Direct

☐ Enable common sense clock

Sense clock frequency (kHz): Set per widget

Actual frequency (kHz): N/A

Inactive sensor connection: Ground

☑ Enable IDAC auto-calibration

☑ Enable compensation IDAC

☐ Enable shield electrode

Enable shield electrode for reliable operation in the presence of water droplets

Scan Resolution/Sensitivity: Scan resolution is the resolution of the raw count counter. The duration for which sensor is scanned is directly proportional to scan resolution parameter, as shown in Equation 3. For every one bit of resolution increase, the scan time doubles.

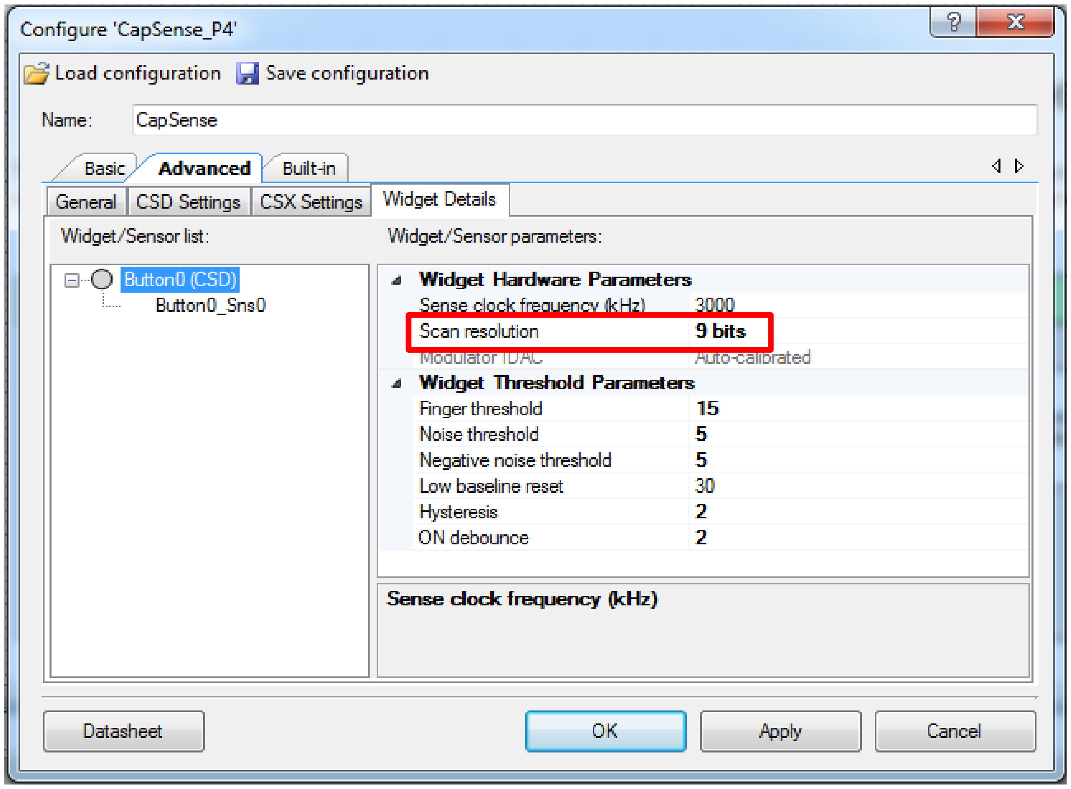
$$Sensor\ scan\ time = \left(2^{\text{Resolution}} - 1\right) \Big/ \text{Modulator Clock Frequency} \qquad \text{Equation 4}$$

Where modulator clock frequency is the clock frequency used by the raw count counter.

The sensitivity parameter is similar to the scan resolution parameter and is defined as signal counts achieved per 1 pF of finger capacitance (C_F). This parameter is used instead of scan resolution when the tuning mode is by using SmartSense. The sensitivity parameter increases or decreases the strength of the sensor signal. As shown in Figure 3, the sensitivity parameter directly affects the scan resolution and scan time of the sensor, which increases with an increase in the sensitivity parameter value. As a result, the higher the sensitivity parameter value, the higher the power consumption.

When using manual tuning, the lower the resolution, the lower will be the scan time and therefore, lower power consumption. However, a lower resolution affects touch response and SNR. Therefore, it is a trade-off for the design engineer to consider. This parameter is available in the Advanced Tab/ Widget Details settings as shown in Figure 7. When using SmartSense, the sensitivity parameter decides the sensor scan time and power consumption. Lowering the sensitivity setting lowers the power consumption. This parameter should be set such that the SNR is greater than 5:1.

Figure 8. Advanced Tab/Widget Details Settings in CapSense Component



Name: CapSense

Widget/Sensor list:

- Button0 (CSD)
  - Button0_Sns0

Widget/Sensor parameters:

Widget Hardware Parameters	
Sense clock frequency (kHz)	3000
Scan resolution	9 bits
Modulator IDAC	Auto-calibrated
Widget Threshold Parameters	
Finger threshold	15
Noise threshold	5
Negative noise threshold	5
Low baseline reset	30
Hysteresis	2
ON debounce	2

Sense clock frequency (kHz)

4.3 Dynamically Switching Power Modes

The average power consumption of the device can be reduced by dynamically switching device power modes.

4.3.1 Use Deep-Sleep Mode Between Scans

You should use the PSoC 4 Deep-Sleep mode to considerably reduce the power consumption of a CapSense design. You can reduce the average power consumption by ensuring that the system is in the lowest power mode (Deep-Sleep) between the CapSense scans.

The CapSense hardware is disabled in Deep-Sleep mode. Therefore, the device must wake up frequently to scan for touches and to perform any other operations required by the system. You can use the WDT in PSoC 4 to wake up the device from Deep-Sleep mode at frequent time intervals.

Typical applications use one of the following methods to wake up the device from Deep-Sleep mode.

Periodic Scan Using WDT

You can use the WDT to put the PSoC 4 device into Deep-Sleep for a specified amount of time and occasionally wake it up to scan the CapSense sensors. This technique is useful since the human interaction with user interface (UI) systems is generally slow, and the CapSense inactivity during the Deep-Sleep interval will not be recognizable by the user.

The frequency at which the system wakes up for a CapSense application is determined by the human response time. Normally, a valid finger touch can last from several tens of milliseconds to several hundred milliseconds. For such a system, PSoC 4 need not wake up every millisecond and scan for a finger touch. The wake-up interval determines the CapSense scan rate; it is application-specific.

GPIO Interrupt

PSoC 4 can wake up from low-power modes with an interrupt generated by the rising edge, the falling edge, or both edges of a digital input signal to a GPIO. Using the GPIO interrupt, a host processor, a transducer, or a user button can wake up the PSoC device by providing a trigger on the GPIO. GPIO-based wakeup consumes the lowest possible current in Deep-Sleep mode.

I^2C Address Match

The Serial Communication Block (SCB) block in the PSoC 4 device has an I^2C function that can operate as a master or a slave. When the I^2C is configured as a slave, it can wake up the PSoC device from Deep-Sleep mode. This method allows a remote master to wake up the target PSoC device through an I^2C address match. The PSoC device can then complete the required operations in Active power mode, send the results to the I^2C master, and return to Deep-Sleep mode.

The I^2C address match allows the device to be in any power mode, regardless of the power mode or operating mode of the host controller. No sync is needed between the host and the slave (PSoC 4). The slave can wake up only when needed, complete the action, and go back to Deep-Sleep. It also helps to avoid unwanted wakeup events when the host is communicating with other devices on the I^2C bus.

Refer to AN90799 – PSoC 4 Interrupts for details on how to configure these interrupt sources.

4.3.2 Enter and Remain in Sleep Mode During CapSense Scan

A CapSense scan is nonblocking. CPU intervention is not required between the start and end of a CapSense scan. You can put the device into Sleep mode after initiating a scan to save power. When the CapSense hardware completes the scan, it generates an interrupt to return the device to Active power mode. Figure 9 and Figure 10 illustrate these concepts.

Figure 9. Timing Diagram in the Absence of Touch

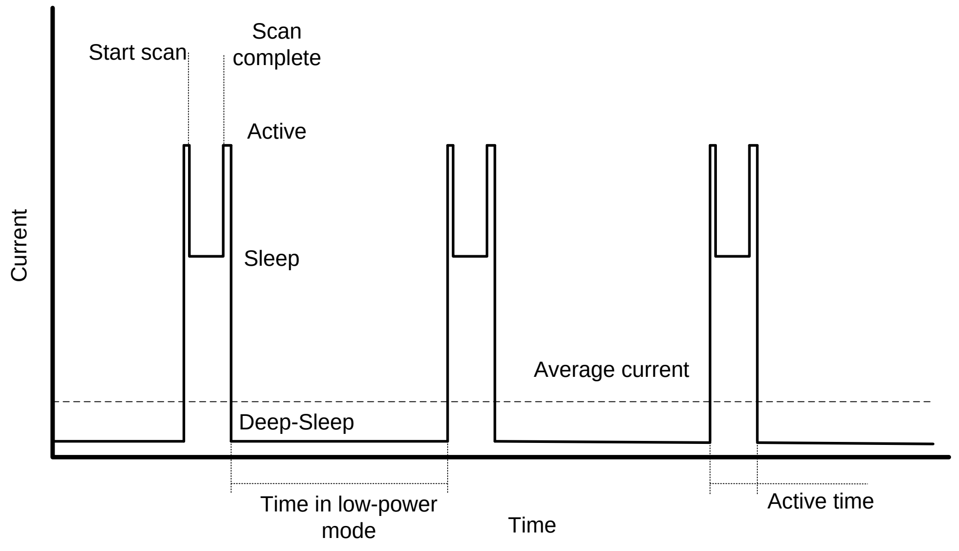
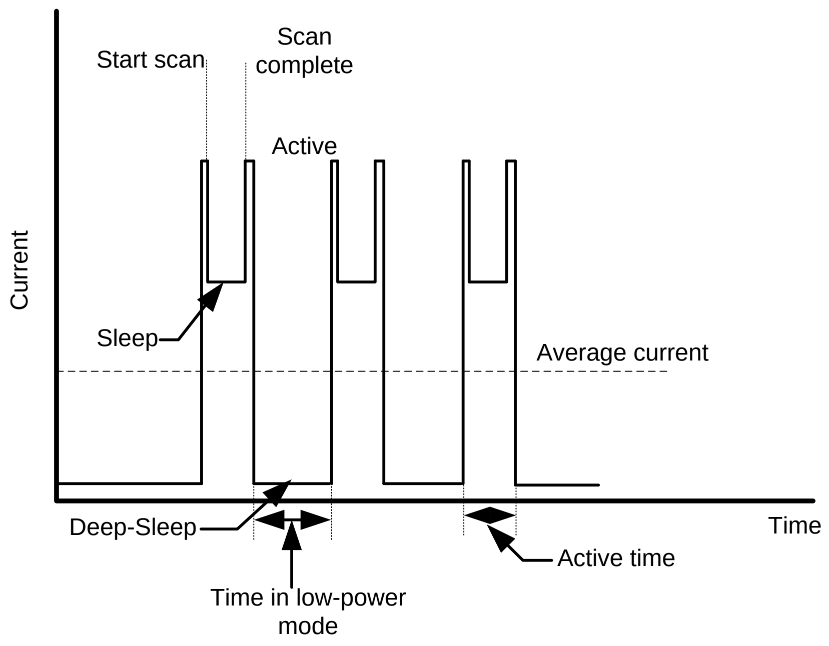


Figure 10. Timing Diagram with Touch



4.4 Refresh rate

As explained in the Factors Affecting Active Time, refresh rate is a parameter that can be configured in firmware to optimize power consumption and performance. The Example Project demonstrates how user can optimize response time and reduce power consumption.

5 System-Level Considerations for Low Power

5.1 Ganged Sensor Scanning

Ganged sensor sampling can be used to decrease the average power consumption. In this method, during the standby mode of the system, all the physical sensors that can wake up the system are connected together to form a single, virtual ganged sensor in the design. Scanning only the ganged sensor consumes less time than scanning all the sensors; therefore, the capacitive controller can be in Sleep mode for a longer time, thereby reducing the average power consumed.

If any physical sensors are touched, the sensor capacitance of the ganged sensor increases and a touch is detected. However, the specific button that was touched during standby mode cannot be determined while sensing a touch on the ganged sensor.

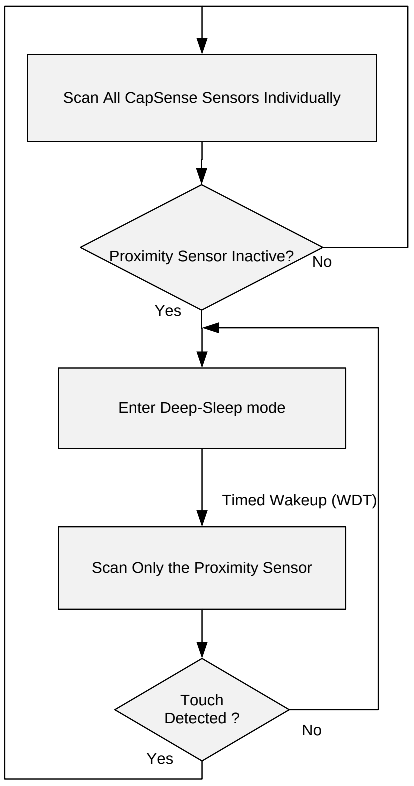
To detect the button that was touched during standby mode, the capacitive controller should wake up and move into its Active mode. The physical sensors need to be disconnected from the ganged sensor and scanned individually to identify the touched sensor.

In this method, the ganged sensor helps to optimize the average power by combining multiple physical sensors into a single, virtual ganged sensor, which ensures that the capacitive controller reverts to Active mode only when a touch is detected. After the capacitive controller moves to Active mode, if no finger touch is detected on the UI panel for a certain period of time, then the capacitive controller should revert to the ganged sensor mode. The code example CE210290 - PSoC4 CapSense Low Power Ganged Sensor demonstrates how to implement a ganged slider to reduce average power consumption.

5.2 Wake-on-Approach Using Proximity Sensor

As the number of sensors in the design increases, the device has to spend more time in Active mode to scan all the sensors. This, in turn, increases the average power consumption. If a design has many sensors, you should include a separate proximity sensor loop that surrounds the sensors. This method does not require any physical touch to wake up the system. When the device wakes up from Deep-Sleep mode, only scan this proximity sensor to reduce the total scan time and therefore the average power consumption. If the proximity sensor is active (when the user's hand approaches the UI panel), the device must stay in Active mode and scan other sensors. If the proximity sensor is inactive, the device can return to Deep-Sleep mode. Figure 11 illustrates this process.

Figure 11. Wake-on Approach Using Proximity Sensor

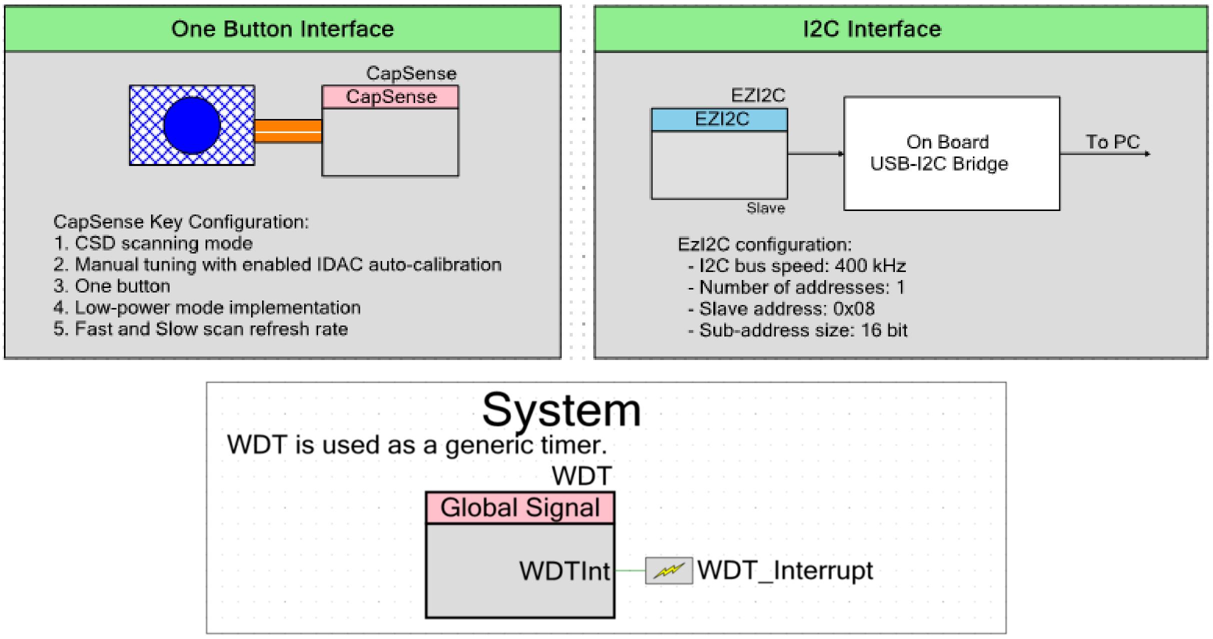

Scan All CapSense Sensors Individually
Proximity Sensor Inactive?
No
Yes
Enter Deep-Sleep mode
Timed Wakeup (WDT)
Scan Only the Proximity Sensor
Touch Detected ?
No
Yes

Additionally, to reduce power, the backlight on the UI panels can be controlled using the proximity sensor. Whenever a capacitive controller is in standby mode (no sensor active), the backlight can be turned off to indicate the inactive mode of the equipment. Once a user's hand approaches the panel and the proximity sensor detects the same, the backlight can be turned on, aiding the user in touching the correct buttons. This also helps reduce the overall power consumed by the end system.

6 Example Project

This code example CE210291 – PSoC4 CapSense® Low Power One Button showcases how to reduce average power consumption in a CapSense system without compromising response time. The CapSense Component is configured with a one-button widget. The project uses the CSD manual tuning mode with IDAC auto-calibration enabled. The EZI2C Component sends the sensor data and button touch position information to the external host/tuner via I^{2}C. The Global Signal Reference Component is configured as a watch dog timer (WDT), which is used to set the refresh rates.

Figure 12. Component Schematics



CapSense Key Configuration:
1. CSD scanning mode
2. Manual tuning with enabled IDAC auto-calibration
3. One button
4. Low-power mode implementation
5. Fast and Slow scan refresh rate

EzI2C configuration:
- I2C bus speed: 400 kHz
- Number of addresses: 1
- Slave address: 0x08
- Sub-address size: 16 bit

WDT is used as a generic timer.

The code example implements two refresh rates: 45[1] Hz in Active mode and 8 Hz in Look-For-Touch (LFT) mode. These refresh rates are chosen based on the power budget and to achieve optimum response time. Initially the device will be in LFT mode, where the sensor is scanned at 8 Hz, as shown in Figure 13. When a touch is detected, the device switches to Active mode, where the sensor is scanned at 45 Hz to detect touch fast finger tap, as shown in Figure 14. If touch is not detected for a specific duration, the device changes the refresh rate to 8 Hz to reduce average power consumption. In Active mode, the average power consumption measured is approximately 20 µA and in Look-For-Touch mode the average power consumption is approximately 6 µA

[1] 45 Hz refresh rate is chosen to meet the average current consumption of 20 uA in active mode and still ensure that the button response is optimum.

Figure 13. Timing Diagram in Active-Scan Mode

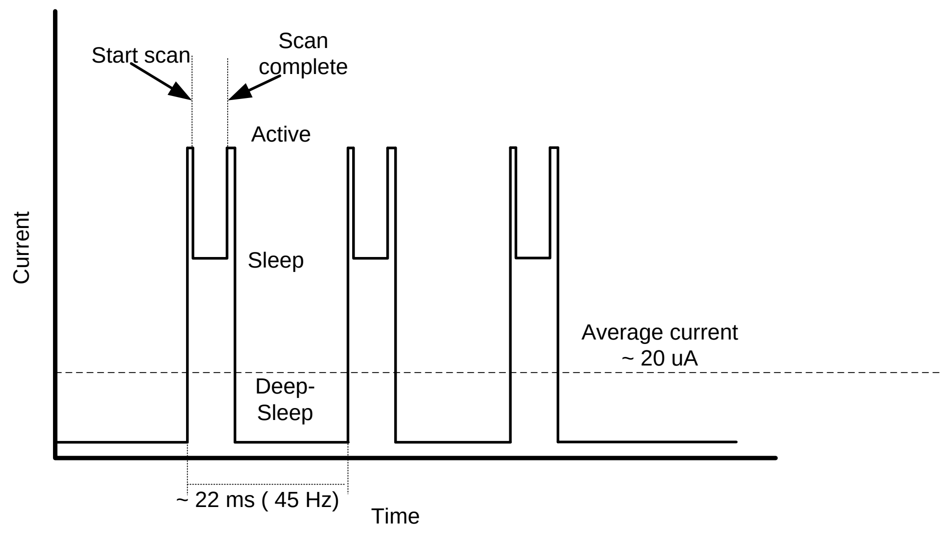
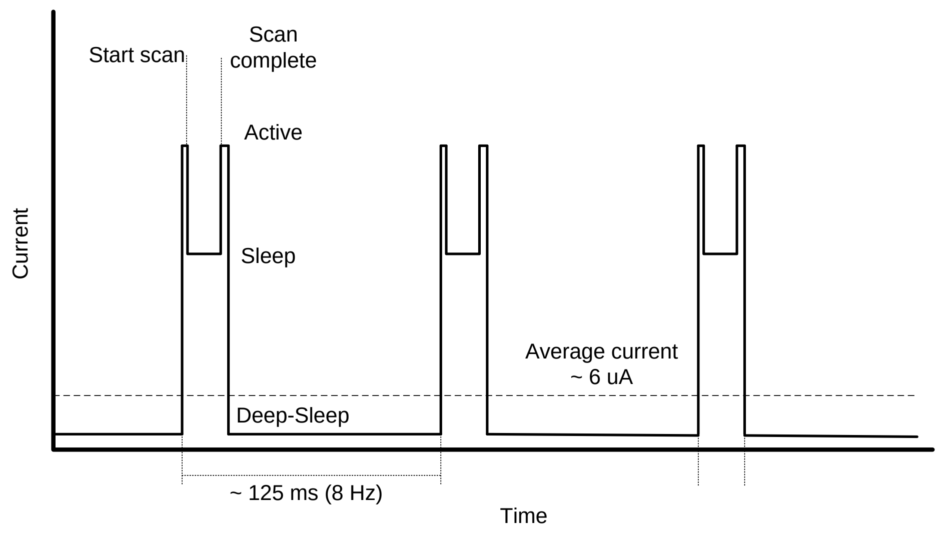


Figure 14. Timing Diagram in LFT Mode



The code example is designed for the PSoC 4000S and associated CY8CKIT-041-40xx kit. The design is easily portable to other PSoC 4 devices and kits, typically by changing only the device and Component pin assignments. Refer to the CE210291 documentation for more details on the firmware implementation.

7 Average Current Measurement

The example project can be used to measure the average current consumption and observe the current profile. The instantaneous current consumed by the device is not a steady value but varies depending on the state of the chip, which dynamically changes with power mode transitions. Therefore, it is difficult to measure each individual instantaneous current with a handheld multimeter because the duration of these current bursts is very short. Therefore, you should use a multimeter that provides the option to set the "aperture" of the measurement. The aperture is the period "T" during which the multimeter measures the instantaneous currents, integrates them, and then displays the average current for the period "T". The respective kit user guide and the code example document provides details on how to measure the current consumption of the device. The following steps summarize this procedure:

1. Build the example project and program the hex file into the kit.

2. Connect a multimeter across the current measurement jumper of the kit using a multimeter such as the Agilent 34411A 6 1/2 digital multimeter.
3. Set the aperture for the measurement based on the scan period. If the exact aperture is not available, then set it to an integral multiple of the scan period.
4. Measure the current. The current measured is the average current over one interval.

8 Summary

AN210998 introduced the hardware and firmware considerations for designing low-power CapSense systems. The example project associated with it demonstrates how to achieve very low power consumption while maintaining a good touch response in applications based on CapSense.

Power consumption can make the difference between a good idea and a successful design. By taking advantage of the many power-saving features available in PSoC 4, you can optimize your design and help ensure that it consumes the lowest amount of power possible.

9 Related Resources

These application notes and the code example give you more information relating to topics that are not fully discussed here.

- AN79953 - Getting Started with PSoC 4
- AN85951 - PSoC® 4 and PSoC® 6 MCU CapSense® Design Guide
- AN64846 - Getting Started with CapSense®
- AN86233 - PSoC® 4 Low-Power Modes and Power Reduction Techniques
- AN77900 - PSoC® 3 and PSoC 5LP Low-Power Modes and Power Reduction Techniques
- CE95288 - CapSense Low Power with PSoC 4
- CE210291 - PSoC4 CapSense Low Power One Button
- CE210290 - PSoC4 CapSense Low Power Ganged Sensor
- CE195286 – PSoC® 4 CapSense Tuner

About the Author

Name: Vijay Kumar Marrivagu

Title: Principal Systems Engr

Background: Several years of experience in digital design and validation.

A Appendix A. Power Mode Summary

This section explains in detail the power modes available in PSoC 4 devices. PSoC 4 features three modes of operation: Active, Sleep, and Deep-Sleep. These power modes differ in power consumption and peripheral availability.

A.1 Active

Active is the primary operating mode of the device and the default mode at boot. In this power mode, the high-frequency internal clock (HFCLK), which is derived from the internal main oscillator (IMO) is on, and the CPU and all the peripherals are operational unless they are specifically disabled by firmware. Active mode typically consumes more power than other modes, as all the blocks in the chip are operational.

Active mode can transition to any other mode. Any valid interrupt or reset event from the other modes returns the PSoC 4 device to Active mode.

A.2 Sleep

Sleep mode is a low-power mode similar to Active mode in that all the peripherals in the device are functional. The clock to the CPU, system clock (SYSCLK), is off during Sleep mode. The CPU can wake up from Sleep mode via interrupts. In this mode, all peripherals such as CapSense or the I^{2}C interface can operate while the CPU is off to reduce the device's power consumption.

A.3 Deep-Sleep

Deep-Sleep is the lowest power mode available in the PSoC 4 device. In this mode, almost all peripherals including CPU, IMO, CapSense, and TCPWM are either off or in a retention state.

Basic peripherals such as I^{2}C, WDT, and GPIO are operational in this mode. The I^{2}C block can wake up the device from Deep-Sleep mode via an address match, depending on the configuration. The ILO clock source is also functional in Deep-Sleep mode. A WDT implemented in the clock block enables periodic wakeup from Deep-Sleep.

Refer to the device datasheet for more detailed power numbers.

A.4 Mode Entry and Wakeup Sources

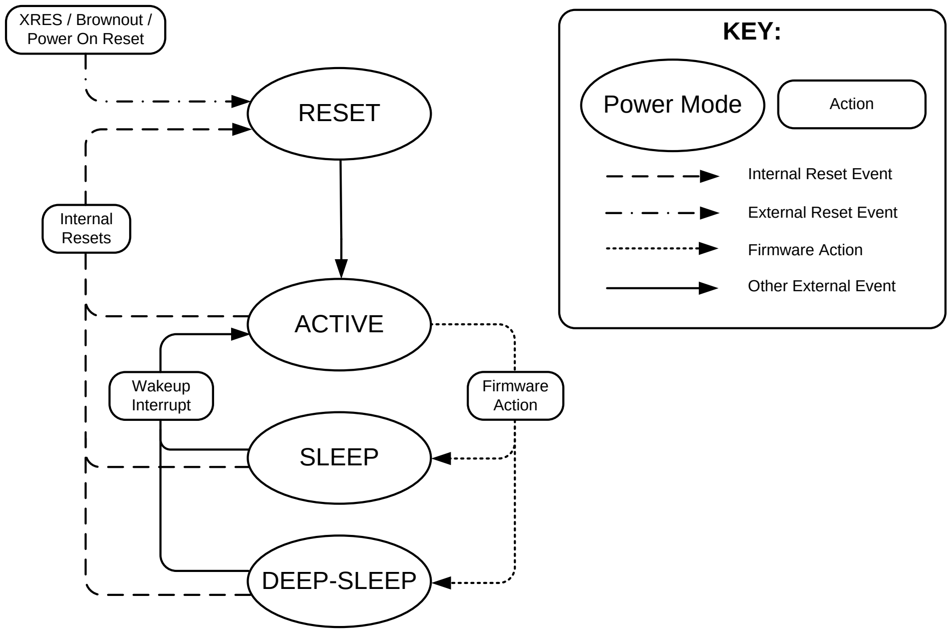
PSoC Creator provides a set of functions called "application programming interfaces (APIs)" that abstract the register-level operations to enter the low-power modes. Table 2 shows the APIs to enter the low-power modes. It also identifies wakeup sources to exit the low-power mode and return to Active mode. Figure 12 shows the possible transitions between power modes.

See the PSoC 4 Architecture TRM for more details on these low-power modes and wakeup sources. Refer to AN90799 – PSoC 4 Interrupts for details on the interrupts available in PSoC 4.

Table 2. Mode Transition Details

Low-Power Mode	API to Enter Low-Power Mode	Wakeup Source	Wakeup Action
Sleep	CySysPmSleep()	Any interrupt source	Interrupt
		Any reset source	Reset
Deep-Sleep	CySysPmDeepSleep()	GPIO interrupt	Interrupt
		I^{2}C address match	Interrupt
		Watchdog timer	Interrupt/Reset
		XRES (external reset pin), brownout	Reset

Figure 15. Power Mode Transitions



B Appendix B: Glossary

Term	Definition
Capacitive sensor	A conductor and substrate, such as a copper button on a PCB, that reacts to a touch or an approaching object with a change in capacitance.
CapSense	Cypress's touch-sensing user interface solution.
Compensation IDAC	A programmable constant current source used by CSD to compensate for excess sensor parasitic capacitance. This IDAC is not controlled by the sigma-delta modulator in the CSD block, unlike the modulation IDAC.
CSD (CapSense Sigma Delta)	A Cypress-patented method of performing self-capacitance (also called self-cap) measurements for capacitive sensing applications. In CSD mode, the sensing system measures the self-capacitance of an electrode, and a change in the self-capacitance is detected to identify the presence or absence of a finger.
Driven shield	A technique used by CSD to enable liquid tolerance in which the shield electrode is driven by a signal that is equal to the sensor switching signal in phase and amplitude.
Electrode	A conductive material such as a pad or a layer on a PCB, ITO, or FPCB. The electrode is connected to a port pin on a CapSense device and is used to serve as a CapSense sensor or to drive specific signals associated with CapSense functionality.
Ganged sensors	The method of connecting multiple sensors together and scanning them as a single sensor. Used to increase the sensor area for proximity sensing and to reduce power consumption. To reduce power when the system is in low-power mode, all the sensors can be ganged together and scanned as a single sensor, which takes less time than scanning all the sensors individually. When the user touches a sensor, the system can transition into Active mode, where it scans all the sensors individually to detect which sensor is activated. PSoC supports sensor ganging in firmware; that is, multiple sensors can be connected simultaneously to AMUXBUS for scanning.
Hatch fill, hatch ground, or hatched ground	When designing a PCB for capacitive sensing, a grounded copper plane should be placed surrounding the sensors for good noise immunity. But a solid ground increases the sensor's parasitic capacitance, which is undesirable. Therefore, the ground should be filled in a special hatch pattern. A hatch pattern has closely placed, crisscrossed lines that look like a mesh, and the line width and spacing between two lines determine the fill percentage. For liquid tolerance, this hatch fill, referred to as a "shield electrode," is driven with a shield signal instead of ground.
IDAC (current output digital-to-analog converter)	A programmable constant current source available in PSoC that is used for CapSense and ADC operations.
Linear slider	A widget consisting of more than one sensor arranged in a specific linear fashion to detect the physical position (in a single axis) of a finger.
Manual tuning	The manual process of setting (or tuning) CapSense parameters.
Modulator clock	A clock source that is used to sample the modulator output from a CSD block during a sensor scan. This clock is also fed to the raw count counter. The scan time (excluding preprocessing and post-processing times) is given by (2N – 1)/Modulator Clock Frequency, where N is the scan resolution.
Modulation IDAC	Modulation IDAC is a programmable constant current source whose output is controlled (ON/OFF) by the sigma-delta modulator output in a CSD block to maintain the AMUXBUS voltage at VREF. The average current supplied by this IDAC is equal to the average current drawn by the sensor capacitor.
Mutual capacitance	The capacitance associated with an electrode (such as TX) with respect to another electrode (such as RX).
Noise (CapSense noise)	The variation in the raw count when a sensor is in the OFF state (no touch), measured as peak-to-peak counts.
Overlay	A nonconductive material, such as plastic or glass, which covers capacitive sensors and acts as a touch surface. The PCB with the sensors is placed directly under the overlay or is connected through springs. The casing for a product often becomes the overlay.
Parasitic capacitance (Cp)	Parasitic capacitance is the intrinsic capacitance of the sensor electrode contributed by the PCB trace, sensor pad, vias, and air gap. It is undesirable because it reduces the CSD sensitivity.
Proximity sensor	A sensor that can detect the presence of nearby objects without any physical contact.

Term	Definition
Radial slider	A widget consisting of more than one sensor arranged in a specific circular fashion to detect the physical position of a finger.
Raw count	The unprocessed digital count output of the CapSense hardware block that represents the physical capacitance of the sensor.
Scan resolution	The resolution (in bits) of the raw count produced by the CSD block.
Scan time	The time to complete a scan of a sensor.
Refresh rate	Refresh rate is defined as the frequency at which the CSD block scans the sensor. Refresh rate decides the minimal possible time from the finger touch until it is reported. The maximum Refresh rate will be limited by the Sensor Scan Time.
Self-capacitance	The capacitance associated with an electrode with respect to circuit ground.
Sensitivity	The change in raw count corresponding to the change in sensor capacitance, expressed in counts/pF. The sensitivity of a sensor is dependent on the board layout, overlay properties, sensing method, and tuning parameters.
Sense clock	A clock source used to implement a switched capacitor front end for the CSD sensing method.
Shield electrode	The copper fill around sensors to prevent false touches due to the presence of water or other liquids. The shield electrode is driven by the shield signal output from the CSD block. See Driven shield.
Signal-to-noise ratio (SNR)	The ratio of the sensor signal, when touched, to the noise signal of an untouched sensor.
SmartSense Auto-Tuning	A CapSense algorithm that automatically sets the sensing parameters for optimal performance after the design phase and continuously compensates for system, manufacturing, and environmental changes.
Tuning	The process of finding the optimum values for the hardware and software or threshold parameters required for CapSense operation.
Widget	A UI element in the CapSense Component that consists of one sensor or a group of similar sensors. Button, proximity sensor, linear slider, radial slider, matrix buttons, and touchpad are the supported widgets.

Document History

Document Title: AN210998 – PSoC 4 Low-Power CapSense Design

Document Number: 002-10998

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	5148966	VJYA	02/24/2016	New application note
*A	5688065	AESATMP8	04/19/2017	Updated logo and Copyright.
*B	6525109	VJYA, AJYA	04/05/2019	Updated to PSoC Creator 4.2

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