THIS SPEC IS OBSOLETE

Spec No: 001-59286

Spec Title: LOW POWER CAPSENSE(R) DESIGN USING CY8C22X45 - AN59286

Sunset Owner: Subbarao Lanka (SLAN)

Replaced by: None

Low Power CapSense® Design using CY8C22x45

Author: Pushek Madaan
Associated Project: No
Associated Part Family: CY8C22x45
Software Version: PSoC Designer™ 5.3
Related Application Notes: None

If you have a question, or need help with this application note, contact the author at pmad@cypress.com.

The power consumption of the PSoC® programmable system-on-chip depends on a number of parameters. This application note explains the factors affecting power consumption and describes how those parameters can be optimized to design a low power CapSense® solution using CY8C22x45 family of PSoC devices.

OBSOLETE

Introduction

Technology that can provide stable performance, such as CapSense buttons, is gaining importance. This is because manufacturers prefer devices that provide comfort, convenience, and innovation. CapSense buttons are considered better than traditional mechanical buttons because of their simple mechanical structure, longer product life, and friendly interface.

A number of CapSense applications are used in portable devices. However, one of the main concerns in system design is power consumption. To design an effective system, you must understand the characteristics of PSoC and use features that help to reduce power. This application note reviews the relevant features of PSoC and describes how to design a low power CapSense system based on the CY8C22x45 PSoC family.

Global Resources related to Power Consumption

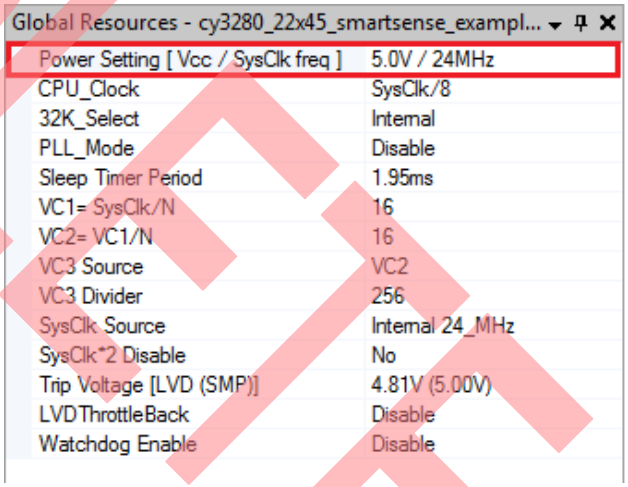
Almost every programmable part of PSoC can affect the system's power consumption. The global parameters that have a major impact on power consumption are: *Power Setting* and *CPU Clock*. The following sections explain these and other global parameters that affect power consumption.

The test values are based on the PSoC CY8C22545, without placing any digital or analog user module. The source code has a infinite loop that keeps the CPU busy. All GPIO pins are set to "High Z Analog" drive mode to minimize current paths into and out of the device. It also disables the GPIO input to internal digital logic by disabling the Schmitt trigger.

Power Setting (V_{CC} / SysClk freq)

This parameter selects the nominal operation voltage and system clock frequency (SysClk freq), from which the internal clocks (V1, V2, V3, and CPU clock) are derived.

Figure 1. Power Setting - Global Resources



User can decide the operating voltage and SysClk freq based on the design requirements. Table 1 shows the typical power supply current for various Power Settings. Other global parameters are set to the respective default values.

Table 1. Typical Supply Current with Various Power Setting Values

Vdd (V)	SysClk (MHz)	Idd (mA)
5	24	8.24
5	6	3.03
3.3	24	4.70
3.3	6	1.94

CPU Clock

This parameter selects a CPU clock in the range of 93.75 kHz to 24 MHz. It is a function of SysClk/N, where N = 1, 2, 4, …, 256. N is set to 1, 2, 4, and 8 for most applications. Table 2 lists the typical power supply current with various CPU Clock settings. Other global parameters are set to the respective default value.

Table 2. Typical Supply Current with Various CPU Clock Settings

CPU Clock	Idd (mA)			
	Vdd = 5 V		Vdd = 3.3 V	
	SysClk 24 MHz	SysClk 6 MHz	SysClk 24 MHz	SysClk 6 MHz
SysClk/1	11.33	7.92	N/A	4.59
SysClk/2	9.56	3.43	5.61	2.21
SysClk/4	8.68	3.14	5.11	2.03
SysClk/8	8.24	3.03	4.70	1.94
SysClk/16	8.02	2.97	4.64	1.95
SysClk/32	7.91	2.94	4.55	1.90
SysClk/128	7.84	2.92	4.50	1.90
SysClk/256	7.81	2.91	4.49	1.90

VC1, VC2, VC3 Source, and VC3 Divider

The internal clocks derived from the SysClk - VC1, VC2, and VC3 - can be connected to digital or analog blocks. The values of VC1, VC2, and VC3 can also influence the system's power consumption. It is recommended that the unused clock dividers (VC1, VC2, and VC3) should be set to maximum value to minimize power. Table 3 lists the current values for a few VC1, VC2, and VC3 settings.

Table 3. Typical Supply Current with Various VC1, VC2 and VC3 Settings

SysClk = 24 MHz; CPU_Clk = SysClk/8	Idd (mA)	
	Vdd = 5 V	Vdd = 3.3 V
VC1 = 16; VC2 = 16; VC3 Source: VC2; VC3 = 256	8.20	4.72
VC1 = 1; VC2 = 16; VC3 Source: VC2; VC3 = 256	8.63	4.97
VC1 = 16; VC2 = 1; VC3 Source: VC2; VC3 = 256	8.21	4.73
VC1 = 16; VC2 = 16; VC3 Source: VC2; VC3 = 1	8.20	4.72
VC1 = 16; VC2 = 16; VC3 Source: SysClk/1; VC3 = 256	8.34	4.80
VC1 = 1; VC2 = 1; VC3 Source: SysClk/1; VC3 = 1	8.95	5.15

SysClk*2 Disable

This parameter is an optional internal clock derived from SysClk using a frequency doubler. This parameter can be used to completely shut down the internal circuitry used for generating SysClk*2 clock, which can reduce power consumption. Therefore, if the project does not require SysClk*2, this function must be disabled to save power. Table 4 shows the current consumption when this feature is enabled or disabled.

Table 4. Typical Supply Current when SysClk*2 is disabled/enabled

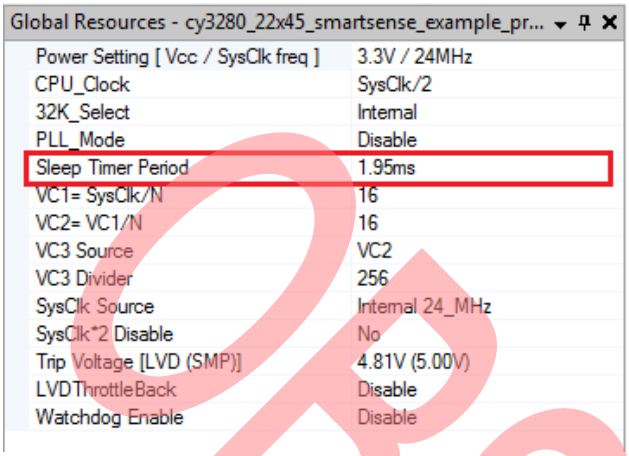
SysClk = 24 MHz, CPUClk = SysClk/8	Idd (mA)	
	Vdd = 5 V	Vdd = 3.3 V
Enable SysClk*2	8.24	4.70
Disable SysClk*2	6.77	3.95

Sleep Mode

The system can be sent to sleep mode to save power. If all the peripherals are disabled, the sleep mode current is at a maximum of 6.5 µA with a 3.3 V supply. The longer the sleep mode duration, lower is the average system current consumption. The Sleep duration is set using the Sleep Timer Period setting in the Global Resources window as shown in Figure 2.

OBSOLETE

Figure 2. Sleep Timer Period in Global Resources



Sleep mode can be enabled by the following two lines of code.

```
INT_MSK0  |=  INT_MSK0_SLEEP;  /*enable  the
sleep timer interrupt*/

M8C_Sleep; /*put PSoC in sleep mode:*/
```

In this state, the CPU is stopped at an instruction boundary. The 24/48 MHz oscillator (IMO), the Flash memory module, and bandgap voltage reference are powered down. The only blocks that remain in operation are the 32 kHz oscillator (external crystal or internal), PSoC blocks clocked from the 32 kHz clock selection, and the supply voltage monitor circuit.

Analog PSoC blocks have individual power down settings that are controlled by firmware, independently of the sleep state. Continuous time analog blocks may remain in operation, since they do not require a clock source. Typically, switched capacitor analog blocks will not operate, since the internal sources of clocking for these blocks are stopped. AN47310 – PSoC 1 Power Savings Using Sleep Mode discusses in detail about the sleep mode operation, wake up sources, power saving techniques and special sleep mode considerations.

To monitor low voltage conditions and power on reset, the bandgap and LVD blocks are periodically re-enabled during sleep.

The system can wake up from sleep as a result of an interrupt or reset event. The sleep timer can provide periodic interrupts to allow the system to wake up, poll peripherals, or perform real time functions, and then go to sleep again. The GPIO (pin) interrupt, supply monitor interrupt, analog column interrupts, and timers clocked externally or from the 32 kHz clock are examples of asynchronous interrupts. They can also be used to wake up the system. Table 5 lists the Sleep Interval settings and Sleep Period.

Table 5. Sleep Interval Settings and Sleep Period

Sleep Interval OSC_CR[4:3]	Sleep Timer Clocks	Sleep Period (nominal)	Watchdog Period (nominal)
00b (default)	64	1.95 ms	6 ms
01b	512	15.6 ms	47 ms
10b	4096	125 ms	375 ms
11b	32768	1 s	3 s

Consecutive instances of the M8C_Sleep command can make PSoC sleep for multiple standard sleep intervals. For example:

```
M8C_Sleep; M8C_Sleep; M8C_Sleep;
```

This will result in a sleep interval of 375 ms if the Sleep Timer Period is set to 125 ms.

Follow these steps to minimize power consumption during sleep mode:

1. In digital modules, APIs are available to stop a digital module. You can call the Stop function of each User Module API to disable the digital function or clear the enable bit [bit0] of the Digital Block Control Register 0 with the assembler code.

2. In the analog function, you must manually disable the function/blocks because the analog User Module API Stop function cannot disable all the related analog modules. For example, the CSD_Stop API cannot disable its VDAC module. Therefore, the VDAC enable bit [bit6] of the VDAC_TRIM register must be cleared by the assembler code. Refer to the *CY8C22x45 CY8C21345 PSoC® Programmable System-on-Chip™ Technical Reference Manual (TRM)*(*http://www.cypress.com/?rID=37728*) for more details about register definitions.

3. If possible, configure drive mode of all GPIO pins as “High Z Analog” mode. For example, set Port 0 to Hign Z Analog mode and output low.

```
M8C_SetBank0;
mov reg[PRT0DR], 00h
mov reg[PRT0DM2], ffh
M8C_SetBank1;
mov reg[PRT0DM0], 00h
mov reg[PRT0DM1], ffh
```

4. Clear post interrupts before going to Sleep Mode with the following code:

```
M8C_SetBank0
M8C_ClearWDTAndSleep
mov reg[INT_VC],00h
```

OBSOLETE

5. Select longer sleep intervals and wake up only by the interrupt from the sleep timer. For example, set the interval timer to 1 second.

```
or reg[OSC_CR0], 18h
```

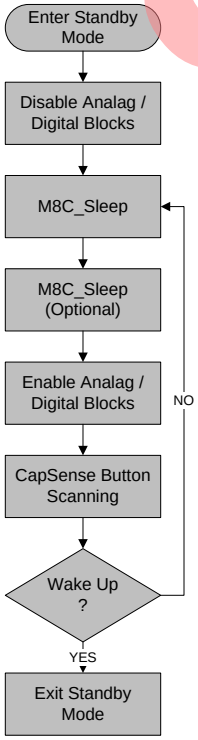
6. Set Sleep System Duty Cycle (PSSD) to low radio so that POR and LVD consume less power. For example, set PSSD to 1/512.

```
mov reg[ECO_TR] , 40h
```

Sleep Mode with CapSense Module

Portable equipments that have CapSense on PSoC always require the standby function. If no button is operated for some time, the system goes into standby mode. In the standby mode, the system scans either one or multiple CapSense buttons at a fixed frequency. If the available CapSense button is pressed, the system is woken up and all the tasks resume. Figure 3 shows the flowchart in standby mode.

Figure 3. Standby Mode Flowchart



The average current in standby mode is calculated by Equation 1.

$$I_{average} = \frac{t_{active} * I_{active} + t_{sleep} * I_{sleep}}{t_{active} + t_{sleep}}$$

Equation 1

where:

- t_{active} = time taken to scan CapSense and other instructions
- I_{active} = the average current in active mode
- t_{sleep} = sleep mode duration
- I_{sleep} = sleep mode current

From equation 1, in order to reduce the power consumption following parameters need to be optimized:

- Reduce active time
- Increase sleep mode time
- Lower the average active current

In real applications where you must guarantee CapSense sensitivity, the adjustable range of I_{active} is limited. Therefore, faster scan speed and longer sleep duration is the key to reduce system power consumption. However, the tradeoff is that longer sleep duration reduces CapSense response time.

Some of the methods which can help in reducing the scan time or increasing the sleep duration are explained below:

Faster Scanning

CY8C22x45 device family has a dedicated IP for CapSense scanning (CSD2X). This IP has two channels which can scan which can scan two sensors simultaneously, thus the total amount of time required to scan "n" number of sensors gets reduced approximately by half. Table 6 shows the time required to scan 1 sensor by a single channel and dual channel architecture for different resolutions.

Table 6: Scanning Time Comparison between Single Channel and Dual Channel

Resolution	Time in (us)	
	Single Channel (1x)	Dual Channel (2x)
9	293	146.5
10	462	231
11	804	402
12	1482	741
13	2844	1422
14	5568	2784
15	11016	5508
16	21900	10950

OBSOLETE

Background Scanning

The CSD2X IP in CY8C22x45 family supports complete scanning of sensors in hardware with 0% CPU intervention. During this time CPU can be used to do other task like updating the display or communicating the data to the host.

If no additional resources are used or time is left even after doing all the software tasks, CPU operating frequency can be reduced to the lowest frequency for the time scanning is happening in the background.

```
 /* checking for sensor scan is complete */
while(!(CSD2X_bScanComplete                    &
CSD2X_SCAN_COMPLETE))
{
/* Reduce the CPU speed to 93.75 kHz */
    OSC_CR0 |= 0x03;
}
```

This reduces the average CPU current and also reduces the active time which can then be used by increasing the sleep duration.

Wakeup Method

This section explains the methods to wake up the system.

Fixed CapSense Button Wakeup Method

Fixed CapSense button wakeup method can effectively reduce system scan button time. Instead of scanning all the buttons, the system can simply scan a single fixed button, which greatly reduces the standby average current. For example, if there are six CapSense buttons, the scan time for every CapSense button is about 0.231ms and active current is about 8mA. The sleep duration is 15.6ms and sleep current is about 5 uA. If all CapSense buttons are scanned, the average current is 0.657 mA (refer to Equation 1). If only one CapSense button is canned, the average current is reduced to 0.122 mA.

Any CapSense Button Wakeup Method

In the previous method, if there are six CapSense buttons, only one can wake up the system. In some cases, the system requires any of the six CapSense buttons to wake up the system. The PSoC specific internal analog bus method can combine all CapSense buttons into a ‘big button’. In the system standby mode, only this ‘big button’ needs to be scanned. This helps to determine whether there is a finger touch on any button. Regardless of which CapSense button is touched, the system can be woken up. Prior to system wakeup, all CapSense buttons are connected to the internal analog bus at the same time. Each time, CapSense can scan the status of all the buttons. After wakeup, all CapSense buttons connect with the internal analog bus individually. You need six CapSense scans to scan six buttons. You can distinguish which button is pressed by the API function. In this method, the system's average current is the same as that in the fixed button method.

Proximity Wakeup Method

Cypress finger proximity is a mature technology. This method is based on Any CapSense Button Wakeup Method. In the standby mode, a ‘big button’ is scanned. Unlike Any CapSense Button Wakeup Method, it is not the finger touching but finger proximity that wakes up the system. This method can speed up the system’s corresponding speed on the buttons and enhance the product features.

Summary

Application designs that consume less power are important for portable devices. This application note gives a detailed analysis of PSoC power consumption, firmware design flowchart with CapSense buttons, and system wakeup methods. The flexibility of PSoC enables you to design a CapSense system that has a long battery life.

About the Author

Name:	Pushek Madaan
Title:	Applications Engineer Staff
Contact:	pmad@cypress.com

OBSOLETE

Document History

Document Title: Low Power CapSense® Design using CY8C22x45 - AN59286

Document Number: 001-59286

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	2876840	ROBC	02/10/2010	New application note.
*A	3909052	PMAD	02/20/2013	Updated Power Setting (V_{CC} / SysClk freq) under Global Resources related to Power Consumption: Updated contents in the section, removed the table “Global Resources” and added Figure 1. Updated Sleep Mode under Global Resources related to Power Consumption: Updated contents in the section, added Figure 2. Updated Sleep Mode with CapSense Module under Global Resources related to Power Consumption: Added Faster Scanning, Background Scanning. Updated in new template.
*B	4821882	DCHE	07/03/2015	Obsolete document.

OBSOLETE

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

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/usb
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

OBSOLETE

PSoC and CapSense are registered trademarks of Cypress Semiconductor Corp. "Programmable System-on-Chip" and PSoC Designer are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

Phone : 408-943-2600
Fax : 408-943-4730
Website : www.cypress.com

© Cypress Semiconductor Corporation, 2010-2015. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.