Applications IP Library **White Paper**

Capacitive Sensing Multimedia Board

Author: Chris Hammer
Associated Project: Yes
Associated Part Family: CY8C21x34
PSoC Designer Version: 4.3, Service Pack 2

Abstract
This White Paper describes the PSoC project and board design for a capacitive sensing multimedia board.

Introduction

In the consumer electronics world, “cool” design is an important factor in a product’s success. PSoC Capsense allows for “cool” designs by replacing mechanical buttons with a sleeker, more refined interface.

One application for PSoC Capsense that is rapidly growing is on multimedia boards for laptops.

The majority of today’s laptops include multimedia capabilities. Access to these functions as well as general-purpose functions are often located on a multimedia strip near the laptop’s keyboard.

Figure 1: Example of mechanical multimedia buttons on a laptop

With PSoC’s capacitive sensing technologies, these functions can be implemented without mechanical buttons resulting in a sleeker look.

This white paper describes a reference design for a PSoC Capsense multimedia board.

Overview: The Example Board

While the details on how to layout a PSoC Capsense board are beyond the scope of this paper (see Application Note AN2292 – Layout guidelines for PSoC Capsense) there are three key design features of this board that are essential to maximizing the sensors sensitivity.

First, minimize ground fill below the sensors is minimized. Ground fill below the sensors effectively shunts the electric field lines from the sensors reducing sensitivity.

Second, the ground fill on the top of the board is spaced significantly away from the sensors (**Figure 2**).

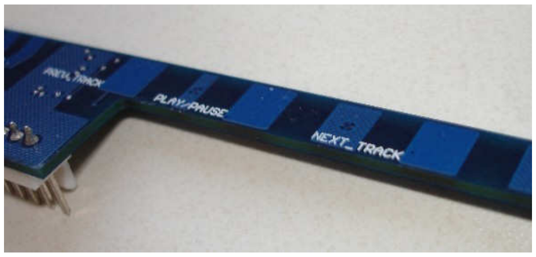


Figure 2: Image of the top of the Multimedia board. Notice the gaps (Dark blue) between the ground fill and the sensors (lighter blue)

Similar to having ground fill below the sensors, ground fill that is placed close or next to a sensor shunts the electric field lines reducing sensitivity.

Finally, the PSoC is placed as close as possible to the slider sensors. Slider sensors typically have much lower sensitivity and therefore are more adversely affected by parasitic capacitance. By placing the PSoC as close as possible to the slider sensors, the parasitic capacitance for each slider sensor is reduced. The reduced parasitic capacitance helps improve the signal to noise ratio and therefore the performance of the slider.

The Firmware

The general flow of the firmware is straightforward – scan the sensors, determine sensor status, communicate to host and repeat.

There are a few caveats to note. First, the project includes an I2C *bootloader*. This allows for in-system programming via the I2C bus between the master and the multimedia board. This is extremely handy when developing the firmware as it allows the firmware in the board to be updated without having to remove it from the system for programming (see Application Note AN2273 I2C Bootloader for PSoC).

Second, in addition to reporting the status of the various sensors, the multimedia board stores data from the sensors that is available for a master device to retrieve via I2C. This allows for easier debugging when developing firmware.

<u>The Sensors</u>

The Multimedia board consists of seven buttons and a 7-segment slider. The function/title of the buttons are DVD, Mute, Previous Track, Play/Pause, Next Track, Stop and WiFi. The functions were chosen as examples and are easily modified or renamed to serve other functions.

The slider is used to control the volume of the laptop speakers. However, it could also be used for other functions such as controlling the brightness of the LCD. For this board the slider resolution has been set to 100 meaning that, depending on where a finger is on the slider, the slider position can take on any value between 0 and 100.

<u>The User Modules</u>

The firmware uses two PSoC User Modules (UMs for short) – CSD for capacitive sensing and EzI2C for communication to the master controller over I2C.

CSD stands for Capsense Sigma-Delta and allows for easy implementation of capacitive sensors such as buttons and sliders. How CSD works is beyond the scope of this paper but more information on CSD can be found in the CSD User Module datasheet. This user module will be used to implement the multimedia board's seven buttons and slider.

The EzI2C user module allows for easy implementation of I2C communications with the multimedia board acting as a slave device. This user module implements I2C in a similar fashion as a 256Byte I2C EEPROM. This means that in addition to the multimedia board's I2C address, there are addressable registers in the Multimedia board that can be written to or read by the I2C Master. For more details on the EzI2C user module see the EzI2C user module datasheet that is included in PSoC Designer 4.3 or later.

With the above user modules, the rest of the firmware design is relatively simple and straightforward.

Using the CSD user module, each sensor on the multimedia board is scanned. Based on the results of the scan, it is determined whether a user is actively touching one of the sensors. The results, whether a sensor is active or not, is then stored in an EzI2C register where it can be read by the master or host.

<u>I2C Registers</u>

This design makes use of 68 8-bit I2C registers (**Table 1**). While this may seem like a lot of registers, 56 of them are used to store CSD data for each sensor (28 each for wRawCount[] and wBaseline[]). These registers are useful for debugging and tuning the buttons and slider sensitivity.

Table 1: Capsense Multimedia Board I2C Registers and Addresses

Address (hex)	Register Name
0x00	**bControl**
0x01	**bButtons**
0x02 … 0x03	**wCentroid** *(2 Bytes)*
0x04	**bRev**
0x05 … 0x0B	**bFingerThrshld[]** *(7 Bytes)*
0x0C … 0x27	**wRawCount[]** *(28 Bytes)*
0x28 … 0x43	**wBaseline[]** *(28 Bytes)*

Seven of the I2C registers (bFingerThrshld[]) store the finger threshold for each of the buttons. With these seven registers, the master or host can control the finger thresholds of the seven buttons. This allows for run-time adjustments in button sensitivity.

The register bRev is used to identify the version of the firmware. This is essential when developing production firmware.

The registers wCentroid, bButtons and bControl are were most of the action takes place. The wCentroid registers hold the finger position of the slider if the slider is pressed. Since the finger position is represented by two bytes (a WORD), two I2C registers are used to store the value.

Each bit in the bButtons (this is read only register) register represents the on/off status of a particular sensor or slider (**Table 2**). A '1' indicates that that particular button has been pressed. Conversely, a '0' means that that particular button is off.

Table 2: The bButtons Register: On/Off status of the buttons and slider.

bButtons Register	
Bit 0	DVD Button
Bit 1	Mute Button
Bit 2	Previous Track Button
Bit 3	Play/Pause Button
Bit 4	Next Track Button
Bit 5	Stop Button
Bit 6	WiFi Button
Bit 7	Slider Active

Finally, the bControl register is used to control the multimedia board. By setting various bits in this register, the master or host can place the multimedia board in low power mode and/or force a baseline reset (Table 3).

The first bit (bit 0) in the bControl register is set by the multimedia board whenever a button or the slider is pressed or activated. The idea is for the master or host to read this bit periodically to determine if any of the sensors has been pressed since the last time the host checked. A '1' means at least one of the sensors is active.

Table 3: The bControl Register

bControl Register	
Bit 0	Activation Bit
Bit 1	Standby Mode
Bit 2	Force Baseline Reset
Bit 3	*reserved*
Bit 4	*reserved*
Bit 5	*reserved*
Bit 6	*reserved*
Bit 7	*reserved*

The second bit (bit 1) in the bControl register is controlled by the master. By default, this bit is set to '0' but if the host sets this bit to '1' the multimedia board will enter low power mode. Low power mode will be discussed in the next section.

Finally, the third bit (bit 2) is also controlled by the master. Normally, this bit is set to '0'. When the host sets this bit to '1', the multimedia board will reset the baselines for all the sensors in the multimedia board. For a description of what baselines are see the CSD datasheet.

Low Power Mode

In this mode, also called "standby", the multimedia board will use *sleep* to reduce current consumption.

In standby mode, the multimedia board will:

1) Go to sleep
2) Wake-up and scan the DVD button
3) Determine if the DVD button is pressed
4) Go back to sleep if the DVD button is not pressed.

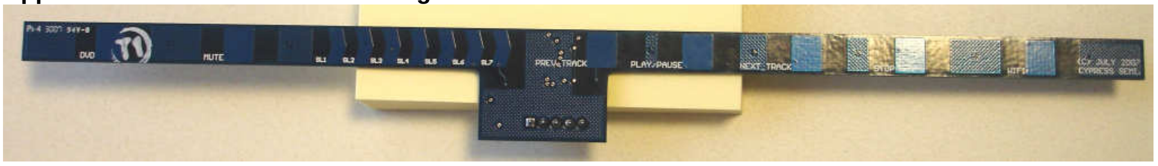
If the DVD button is pressed while the multimedia board is in standby mode, the board will return to normal operation and set the standby bit in the bControl register to '0'.

Conclusion

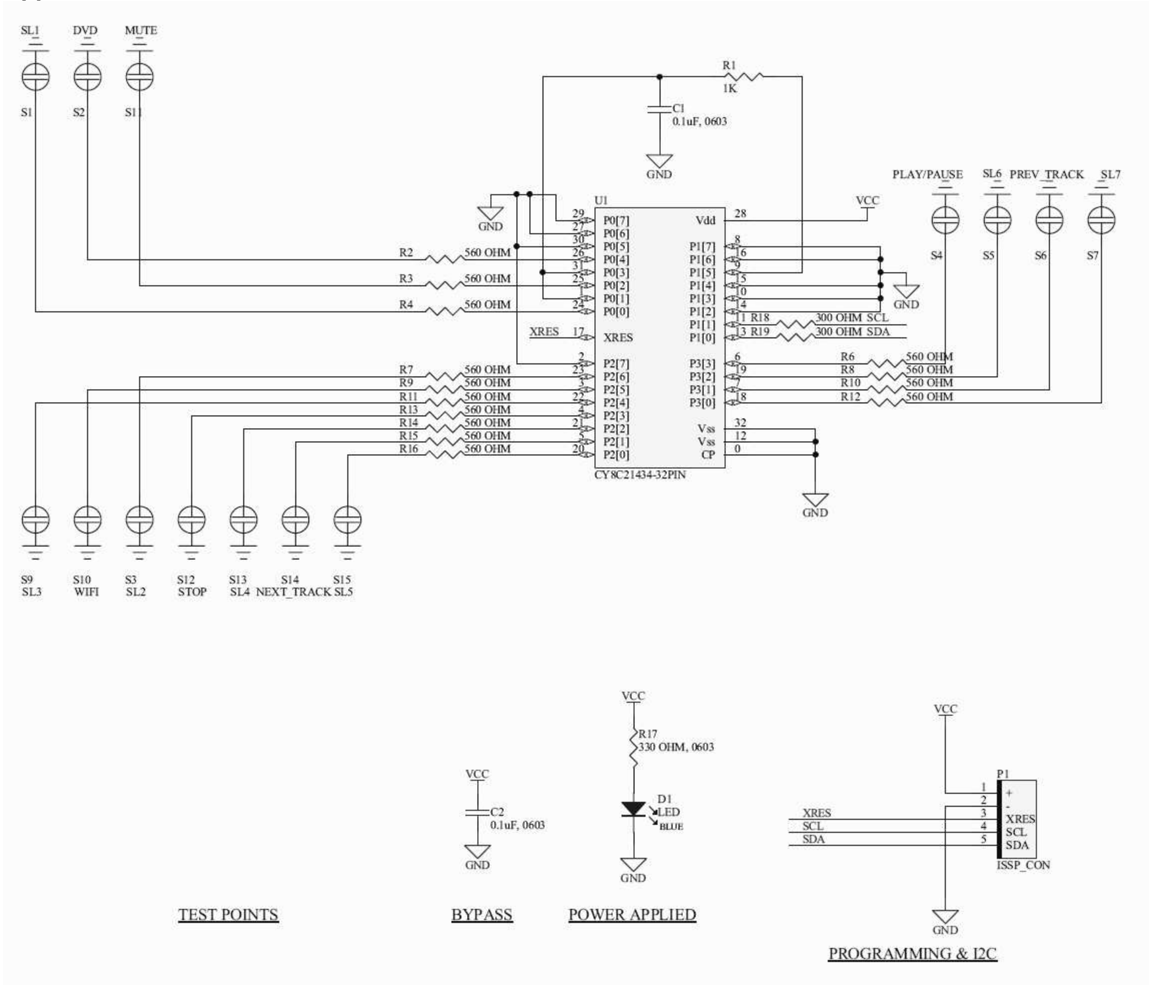
The firmware and board layout accompanying this document are meant as a reference for design development of a PSoC Capsense multimedia board.

PSoC Capsense allows a designer to easily replace mechanical buttons with sleek design and improved functionality. This is especially true for multimedia functionality on laptops or other personal devices.

Appendix A: Multimedia Board Image



Appendix B: Schematic



Appendix C: I2C Register Map

Addr (hex)	Register Name	Definition/Description/Function		
00	**bControl** *(register 0)*	**Control Register**		
		Bit 0	Activation Bit	1 = Sensor(s) is/are active 0 = no sensor is active
		Bit 1	Standby Mode	1 = standby mode (low power) 0 = normal mode
		Bit 2	Force Baseline Reset	1 = Force Baseline Update 0 = Normal Operation
		Bit 3	*reserved*	*Reserved*

		Bit 4	*reserved*	*Reserved*
		Bit 5	*reserved*	*Reserved*
		Bit 6	*reserved*	*Reserved*
		Bit 7	*reserved*	*Reserved*
01	**bButtons** *(register 1)*	**Status of the Buttons and Slider**		
		Bit 0	DVD Button	1 = DVD button activated 0 = DVD button not activated
		Bit 1	Mute Button	1 = Mute button activated 0 = Mute button not activated
		Bit 2	Previous Track Button	1 = Previous Track button activated 0 = Previous Track button not activated
		Bit 3	Play/Pause Button	1 = Play/Pause button activated 0 = Play/Pause button not activated
		Bit 4	Next Track Button	1 = Next Track button activated 0 = Next Track button not activated
		Bit 5	Stop Button	1 = Stop button activated 0 = Stop button not activated
		Bit 6	WiFi Button	1 = WiFi button activated 0 = WiFi button not activated
		Bit 7	Slider Active	1 = Slider activated 0 = Slider not activated
02	**wCentroid** *(MSB - register 2)*	Finger position on the slider (most significant byte)		
03	**wCentroid** *(LSB - register 3)*	Finger position on the slider (Least significant byte)		
04	**bRev** *(register 4)*	Firmware revision number		
05	**bFingerThrshld[0]**	DVD button finger threshold		
06	**bFingerThrshld[1]**	Mute button finger threshold		
07	**bFingerThrshld[2]**	Previous Track button finger threshold		
08	**bFingerThrshld[3]**	Play/Pause button finger threshold		
09	**bFingerThrshld[4]**	Next Track button finger threshold		
0A	**bFingerThrshld[5]**	Stop button finger threshold		
0B	**bFingerThrshld[6]**	WiFi button finger threshold		
0C	**wRawCount[0]**	DVD Button raw counts (2 bytes)		
0E	**wRawCount[1]**	Mute Button raw counts (2 bytes)		
10	**wRawCount[2]**	Previous Track Button raw counts (2 bytes)		
12	**wRawCount[3]**	Play/Pause Button raw counts (2 bytes)		
14	**wRawCount[4]**	Next Track Button raw counts (2 bytes)		
16	**wRawCount[5]**	Stop Button raw counts (2 bytes)		
18	**wRawCount[6]**	WiFi Button raw counts (2 bytes)		
1A	**wRawCount[7]**	Slider Sensor 1 raw counts (2 bytes)		
1C	**wRawCount[8]**	Slider Sensor 2 raw counts (2 bytes)		

1E	**wRawCount[9]**	Slider Sensor 3 raw counts (2 bytes)
20	**wRawCount[10]**	Slider Sensor 4 raw counts (2 bytes)
22	**wRawCount[11]**	Slider Sensor 5 raw counts (2 bytes)
24	**wRawCount[12]**	Slider Sensor 6 raw counts (2 bytes)
26	**wRawCount[13]**	Slider Sensor 7 raw counts (2 bytes)
28	**wBaseline[0]**	DVD Button baseline counts (2 bytes)
2A	**wBaseline[1]**	Mute Button baseline counts (2 bytes)
2C	**wBaseline[2]**	Previous Track Button baseline counts (2 bytes)
2E	**wBaseline[3]**	Play/Pause Button baseline counts (2 bytes)
30	**wBaseline[4]**	Next Track Button baseline counts (2 bytes)
32	**wBaseline[5]**	Stop Button baseline counts (2 bytes)
34	**wBaseline[6]**	WiFi Button baseline counts (2 bytes)
36	**wBaseline[7]**	Slider Sensor 1 baseline counts (2 bytes)
38	**wBaseline[8]**	Slider Sensor 2 baseline counts (2 bytes)
3A	**wBaseline[9]**	Slider Sensor 3 baseline counts (2 bytes)
3C	**wBaseline[10]**	Slider Sensor 4 baseline counts (2 bytes)
3E	**wBaseline[11]**	Slider Sensor 5 baseline counts (2 bytes)
40	**wBaseline[12]**	Slider Sensor 6 baseline counts (2 bytes)
42	**wBaseline[13]**	Slider Sensor 7 baseline counts (2 bytes)

About the Author

Name: Chris Hammer
Title: Senior Application Engineer
Contact: xch@cypress.com