

AN2212

Authors: Chris and Vincent Paiano

Associated Project: Yes

Associated Part Family: CY8C25xxx, CY8C26xxx

[GET FREE SAMPLES HERE](#)

Software Version: PSoC Designer™ 4.1

Associated Application Notes: None

Application Note Abstract

The PSoC® is configured with three PGAs, a two-pole band pass filter, a comparator, and a counter, for use as a battery-operated dog bark detector and ultrasonic emitter. It uses a single mini-speaker for input and output.

Introduction

It's well known that many animals are particularly sensitive to high-frequency sounds that humans are unable to hear. Many commercial pest-repelling devices based on this principle are available, most of them operating in the range of 30 to 50 kHz. The aim of this project, however, is to design a slightly different ultrasonic sound generator that could be used to train dogs.

Just imagine the possibilities - make your pet think twice before barking again in the middle of the night. Dogs tend to respond best to frequencies between 15 and 25 kHz, and older canines are less susceptible to higher tones.

This PSoC project outputs a sweeping ultrasonic signal throughout the range of 15-19 kHz to ensure effectiveness upon detecting a bark from the dog wearing the device on its collar. It is a simple matter to make it sweep all the way up to 25 kHz, but the particular speaker used in this project had no frequency response above 19 kHz.

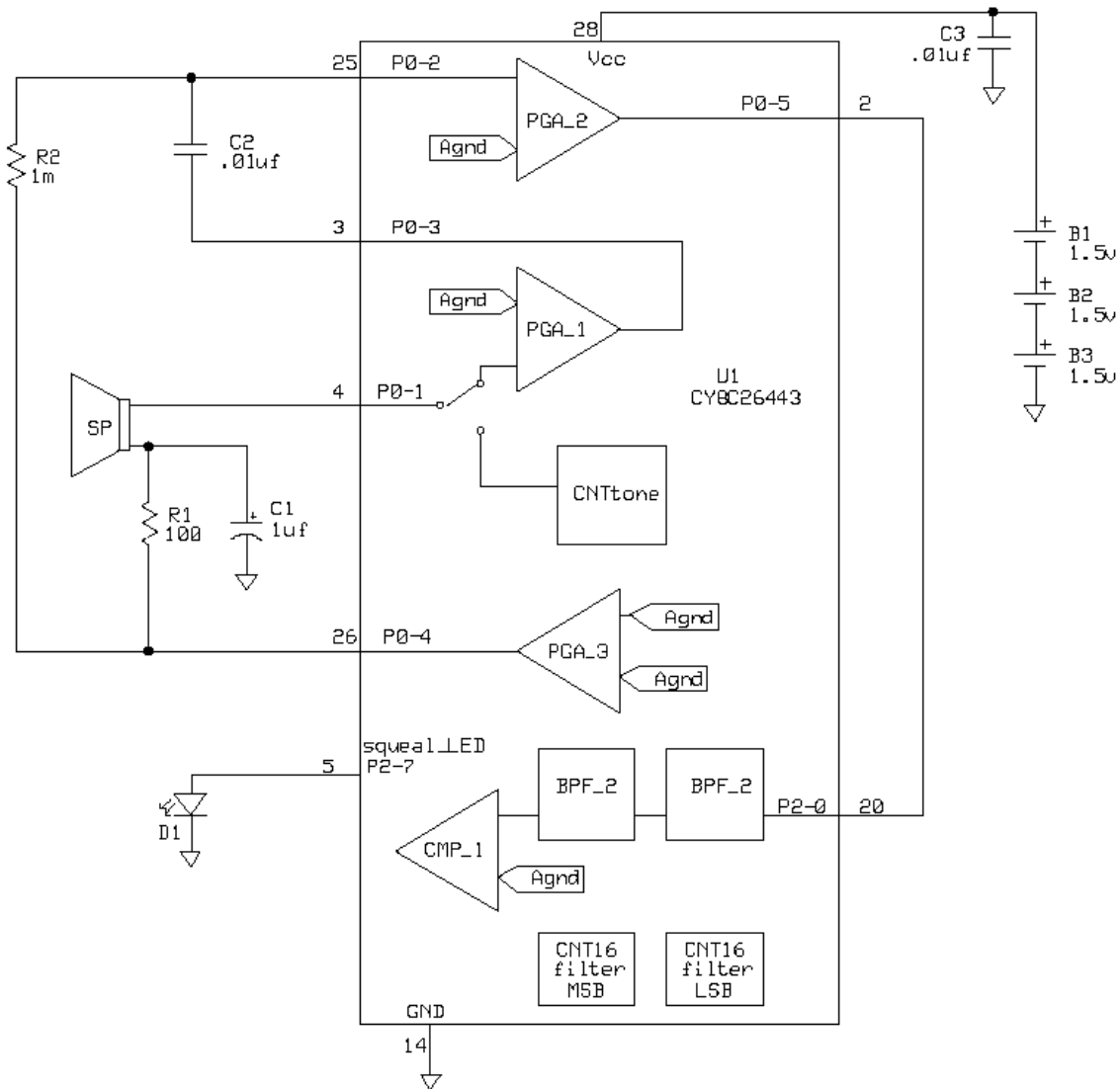
Software/PSoC Implementation

The PSoC implementation involves an interesting setup for the analog audio portion. Two stages of PGAs are used in series to apply plenty of gain to the microphone/speaker for adequate sensitivity. The output goes through a BPF2 User Module designed with the handy Filter Design Wizard (accessed by right-clicking the BPF2 module once placed), and then is connected internally to a comparator input. This comparator is, effectively, the bark detector. An interrupt is manually configured to trigger upon the rising edge of the comparator's output.

The software counts the number of comparator triggers it gets in a short period of time. If there are sufficient triggers to constitute a bark, an ultrasonic sweep from 15-19 kHz and down again is produced. In this particular implementation, the speaker is also used as the microphone – and the PSoC switches from an analog input to a strong digital output to produce the sound, and back again to listen for more. A pull-up output is supplied for an LED indicator.

The PSoC code, illustrated in Figure 2, demonstrates how all this might be achieved in practice. A flowchart of the code is shown in Figure 3, and the configuration of PSoC resources appear in Figure 4, Figure 5, Figure 6, and Figure 7.

Figure 1. Schematic



Hardware Implementation

R_1 and C_1 isolate and filter the AGND reference so that both PGAs can use it while operating as a microphone. The PGAs act as a buffer while operating as a speaker. C_2 blocks DC and allows R_2 to restore the AGND offset so maximum gain can be set on both stages. Three 1.5V batteries, AAA in the prototypes case, power the PSoC, even though it is set to 3.3V. The idle current is about 10 mA, and about 50 mA per bark for 2 seconds. The LED is sourced through an internal pull-up resistor to limit current.

Code 1. PSoC C Code

```

#include <m8c.h>           //Part-specific constants and macros
#include "PSoCAPI.h"       //PSoC API definitions for all User Modules
#include "ports.h"
#include "globalparams.h"
    #define Bit(bitNumber)      ( 1 << (bitNumber) )
    void Wait(long int); long int Counter = 0;
#define UltrasonicPeriodMax 100 //for 15kHz min freq w/24V2 clock = 24MHz/16
#define UltrasonicPeriodMin 79  //for 19kHz max freq w/24V2 clock = 24MHz/16
#define UltrasonicToneChangeDelay 200
#define BarkTriggerTimeout 100
#define NumTriggers 20        //Number of comparator edges required to constitute a bark
#define DriveModeSwitchDelay 500
#define Squeal(b) (PRT2DR = (b==0) ? (PRT2DR&0x7F) : (PRT2DR|0x80)) //P2_7
char UltrasonicPeriod;
char Barking=0;
char PrevBarkCount=0;
void UltrasonicSqueal(void);
void main()
{
    Squeal(0);
    CNT16_Filter_DisableInt();
    CNT16_Filter_Start();
    PGA_Ref_Start(PGA_Ref_HIGHPOWER);
    PGA_Audio_Start(PGA_Audio_HIGHPOWER);
    PGA_Audio2_Start(PGA_Audio2_HIGHPOWER);
    BPF2_Audio_Start(BPF2_Audio_HIGHPOWER);
    CMP_Audio_Start(CMP_Audio_HIGHPOWER);
    CNTTone_DisableInt();
    M8C_EnableIntMask(INT_MSK0,INT_MSK0_ACOLUMN_3); //Comparator interrupt
    M8C_EnableGInt;
    while(1)
    {
        if (Barking)
        {
            PrevBarkCount=Barking;
            Wait(BarkTriggerTimeout);
            if (Barking==PrevBarkCount) Barking=0;
            if (Barking>NumTriggers)
            {
                Barking=0;
                UltrasonicSqueal();
            }
        }
    }
}
void UltrasonicSqueal()
{
    Squeal(1);
    PGA_Audio_Stop();
    PRT0DM0 = PORT_0_DRIVE_0 | Bit(1);
    PRT0DM1 = PORT_0_DRIVE_1 & ~Bit(1);
    PRT0GS = PORT_0_GLOBAL_SELECT | Bit(1);
    UltrasonicPeriod=UltrasonicPeriodMax;
    CNTTone_WritePeriod(UltrasonicPeriod);
    CNTTone_WriteCompareValue(UltrasonicPeriod/2);
    CNTTone_Start();
    Wait(UltrasonicToneChangeDelay);
    while(UltrasonicPeriod>UltrasonicPeriodMin)
    {
        CNTTone_WritePeriod(--UltrasonicPeriod);
        CNTTone_WriteCompareValue(UltrasonicPeriod/2);
        Wait(UltrasonicToneChangeDelay);
    }
}

```

```

}
while(UltrasonicPeriod<UltrasonicPeriodMax)
{
    CNTTone_WritePeriod(++UltrasonicPeriod);
    CNTTone_WriteCompareValue(UltrasonicPeriod/2);
    Wait(UltrasonicToneChangeDelay);
}
CNTTone_Stop();
PRT0DM0 = PORT_0_DRIVE_0 & ~Bit(1);
PRT0DM1 = PORT_0_DRIVE_1 | Bit(1);
PRT0GS = PORT_0_GLOBAL_SELECT & ~Bit(1);
Wait(DriveModeSwitchDelay);
PGA_Audio_Start(PGA_Audio_HIGHPOWER);
Barking=0;
Squeal(0);
}
#pragma interrupt_handler BarkINT
void BarkINT()
{
    Barking++;
}
void Wait(long int ToWait){for(Counter = 0; Counter < ToWait; Counter++);}

```

Figure 2. Program Flowchart

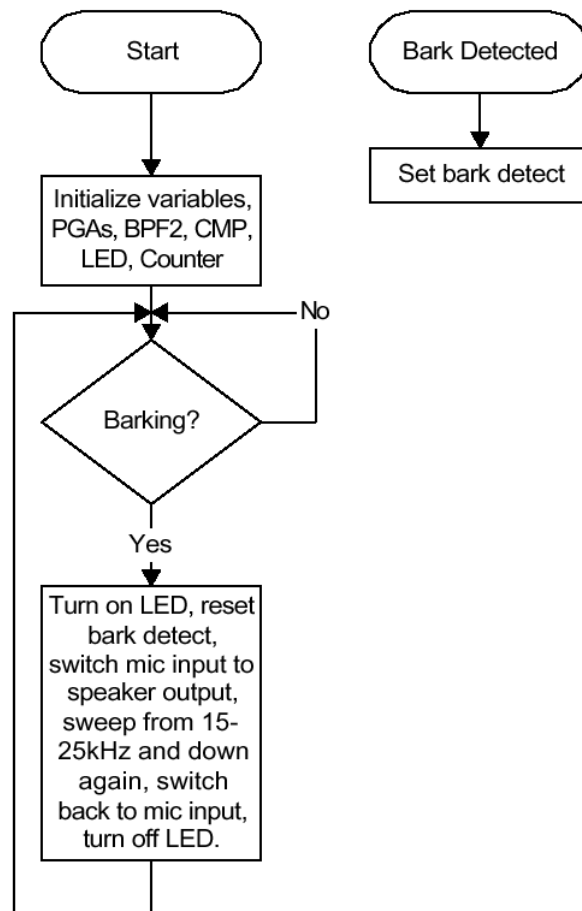


Figure 3. PSoC Pin Configuration

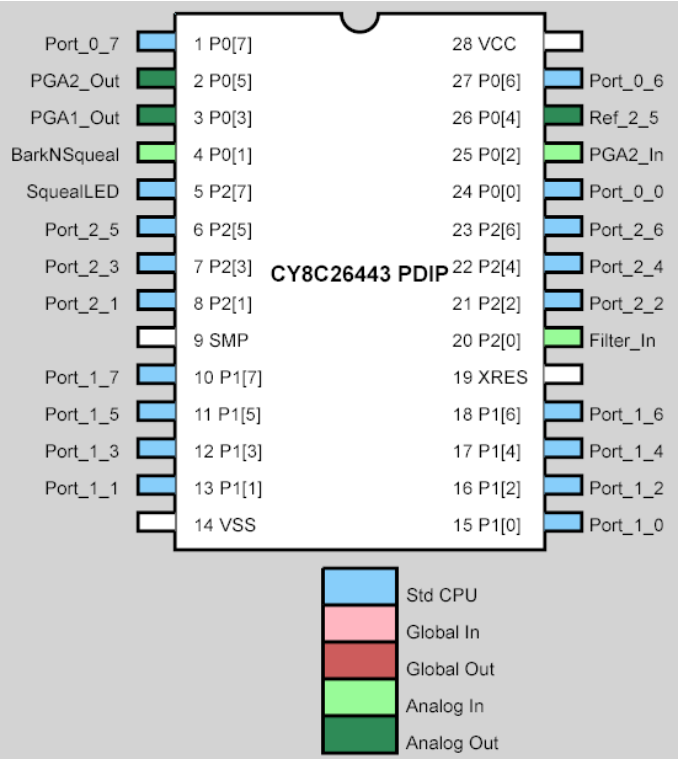


Figure 4. PSoC User Module Configuration

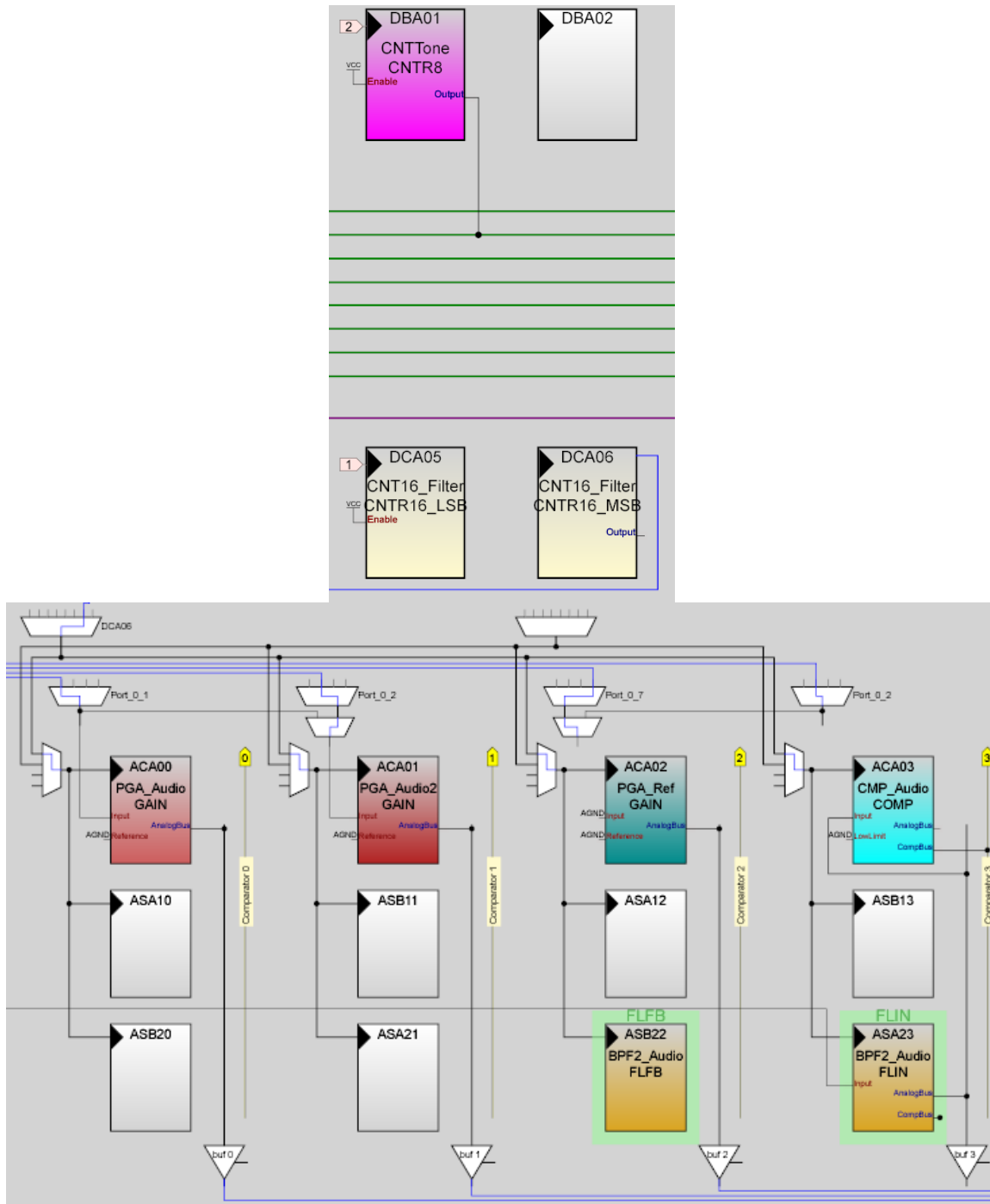


Figure 5. PSoC User Module Parameter Settings

BPF2_Audio			
User Module Parameters			
C1	20		
C2	10		
C3	10		
C4	4		
CA	32		
CB	32		
Input	Port_2_0		
AnalogBus	AnalogOutBus_3		
CompBus	DISABLE		
CNTTone			
User Module Parameters			
Clock	24V2		
Enable	High		
Output	Global_OUT_1		
Period	0		
CompareValue	0		
CompareType	Less Than Or Equal		
InterruptType	Terminal Count		

CMP_Audio			
User Module Parameters			
AnalogBus	Disable		
CompBus	ComparatorBus_3		
Input	AnalogOutBus_3		
LowLimit	AGND		
RefValue	0.562		

CNT16_Filter			
User Module Parameters			
Clock	24V1		
Enable	High		
Output	None		
Period	380		
CompareValue	190		
CompareType	Less Than Or Equal		
InterruptType	Terminal Count		

PGA_Audio			
User Module Parameters			
Gain	16.00		
Input	AnalogColumn_InputMUX_0		
Reference	AGND		
AnalogBus	AnalogOutBus_0		

PGA_Audio2			
User Module Parameters			
Gain	16.00		
Input	AnalogColumn_InputSelect_1		
Reference	AGND		
AnalogBus	AnalogOutBus_1		

Figure 6. PSoC Global Resource Settings

PGA_Ref	
User Module Parameters	
Gain	1.000
Input	AGND
Reference	AGND
AnalogBus	AnalogOutBus_2

About the Authors

Name: Chris and Vincent Paiano

Title: B.S., Computer Engineer and
Electronic Engineer

Background: 22+ years programming/computer
experience.

40+ years electronics/design and
troubleshooting experience.

Contact: psoc@chrispaiano.com
engineering@chrispaiano.com

In March of 2007, Cypress recataloged all of its Application Notes using a new documentation number and revision code. This new documentation number and revision code (001-xxxxx, beginning with rev. **), located in the footer of the document, will be used in all subsequent revisions.

PSoC is a registered trademark of Cypress Semiconductor Corp. "Programmable System-on-Chip," PSoC Designer, and PSoC Express 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
<http://www.cypress.com/>

© Cypress Semiconductor Corporation, 2004-2007. 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.