Application Note AN2105

Pyroelectric Infrared Motion Detector, PSoC Style

By: David Van Ess
Associated Project: AN2015.zip
Associated Part Family: CY8C25xxx, CY8C26xxx

Summary
An application is shown to build a **P**yroelectric **I**nfra-**R**ed (PIR) motion detector. This design uses no external active components to buffer, amplify and detect a moving infrared signal source.

Introduction
Applications to detect motion do so by either:

- Generating some stimulus and sensing its reflection.
- Sensing some natural signal generated by an object.

PIR sensors allow motion detectors to be built that do not require a stimulus signal. Relying only on a body's radiated infrared radiation, a passive detector is less expensive to construct.

The PSoC total system-on-a-chip integrates all the active circuitry required to construct a motion detector. This leaves only the addition of a PIR sensor and some passive components.

This Application Note:

- Describes Pyroelectric Infrared (PIR) sensor operation.
- Shows PSoC circuitry to implement a motion detector.
- Lists the software required to control the application.
- Gives details where components to construct a motion detector can be obtained.

Infrared Radiation
Infrared is the portion of the electromagnetic spectrum that falls between microwaves and visible light. Infrared has wavelengths longer than visible light but shorter than microwaves.

Humans, at normal body temperature, radiate most strongly in the infrared, at an approximate wavelength 10 μm.

To detect this signal, a transducer is required that converts the infrared signal to a form detectable with conventional circuitry.

Pyroelectric Infrared Sensors
A pyroelectric sensor is made of ceramic material that generates a surface charge when exposed to infrared radiation. As the amount of radiation changes, so does the charge. Being a high impedance signal, a FET is used to buffer this potential as shown in Figure 1:

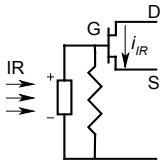


Figure 1: Pyroelectric Charge Measured with a FET

This sensor is sensitive to a wide range of radiation. To optimize for human detection, a filter window is added to limit the incoming radiation to a range of 8 μm to 14 μm. It is shown in Figure 2 with an external resistor to convert the FET current to a voltage:

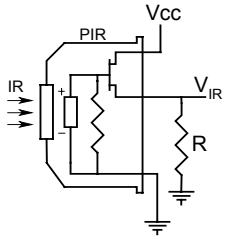


Figure 2: Single Element Pyroelectric Detector

The output voltage is a function of the amount of Infrared Radiation (IR) sensed at the input. Unfortunately, the output is also affected by vibration, radio interference, and sunlight. Figure 3 shows an improved motion sensor with dual elements:

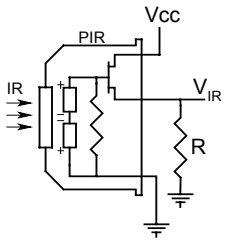


Figure 3: Dual Element Pyroelectric Detector

The sensing elements are connected such that one subtracts from the other. This arrangement causes any signal common to both elements to be canceled. A body passing in front of the sensor activates first one and then the other element while vibration, and other background signals, affect both elements simultaneously and are cancelled. The layout of the two elements allows for maximum sensitivity along a single axis. A typical dual-element layout is shown in Figure 4:

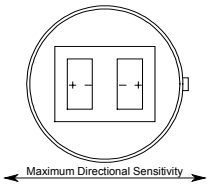


Figure 4: Typical Dual-Element Pyroelectric Layout

Each application's requirements determine how the sensor is mounted for desired directional sensitivity.

Figure 5 shows a quad element sensor that eliminates directional sensitivity:

Figure 5: Quad Element Pyroelectric Detector

The quad elements are configured as a pair of top elements subtracted from the bottom elements. The left pair is then subtracted from the right pair. Again, a common signal is canceled while movement in either axis is detected. A typical quad element layout is shown in Figure 6:

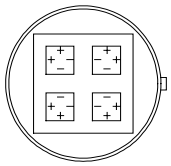


Figure 6: Typical Quad Element Pyroelectric Layout

For this example, a RE200B PIR sensor is used. Specifications are shown in Table 1:

Table 1: RE200B Specifications

Type of Sensor	Series-Opposed Dual Element
Package	TO - 5 Metal Can
Field of View	138° x-axis, 125° y-axis
Supply Voltage	$3V_{dc}$ – $10V_{dc}$
Source Voltage	$350mV_{dc}$ - $1.5V_{dc}$ VD = 5V, Rs = 47 kΩ
Signal Output	3.5 V_{pp} Measured with 72.5 dB gain exposed to 420 K blackbody.
Noise Output	90 mV_{pp} Measured with 72.5 dB gain for 20 seconds with no infrared input.
Spectral Response	5 μm ~ 14 μm
Freq Response	0.3 Hz ~ 3.0 Hz
Operating Temp.	-30°C ~ +70°C

The RE200B was selected for this example because:

- It is readily available from many sources.
- Dual-element construction makes the design less sensitive to external conditions.
- It has reasonable specifications.
- The data sheet has key phases like "for motion detection," "low-cost version," and "satisfies a customer's cost-reduction needs."

With a field-of-view well over 90, a lens is required for detecting motion any more than a few feet away. Figure 7 shows a Fresnel Lens placed in front of the PIR detector.

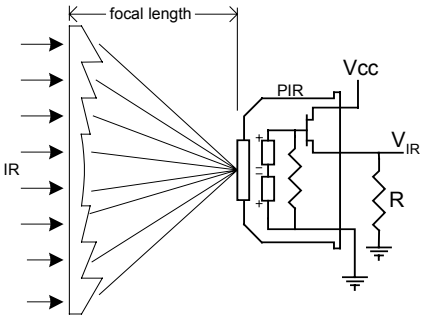


Figure 7: PIR Detector with Fresnel Lens

It is important that the lens be mounted a focal length away from the PIR detector. A Fresnel Lens is used because it is low cost, it is easy to mount, and it can be made of a plastic that does not attenuate IR signals.

A PSoC PIR Motion Detector

Now that you know more about PIR detectors, a block diagram of the motion detector is shown in Figure 8:

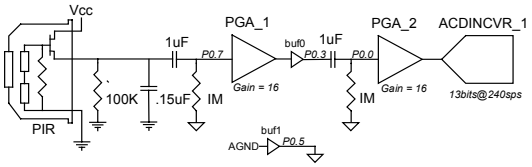


Figure 8: PIR Motion Detector Block Diagram

A 100-kΩ resistor is connected to the source of the PIR detector. A 0.15 µF capacitor is connected in parallel to the resistor. This combination of components makes a 10 Hz low-pass filter. A 0.16 Hz high-pass filter is made with a 1 µF capacitor and a 1 MΩ resistor. This shifts the signal so it is centered around **AGND**. **PGA_1** provides a gain of 16 to the signal. It is brought back out **buf0** and is high-pass coupled into the input of **PGA_2** to provide another gain of 16 for a total gain of 256. The signal is then fed to a 13-bit ADCINCVR_1 where the signal is digitized at 240 samples per second (sps). The User Module placement is shown in Figure 9:

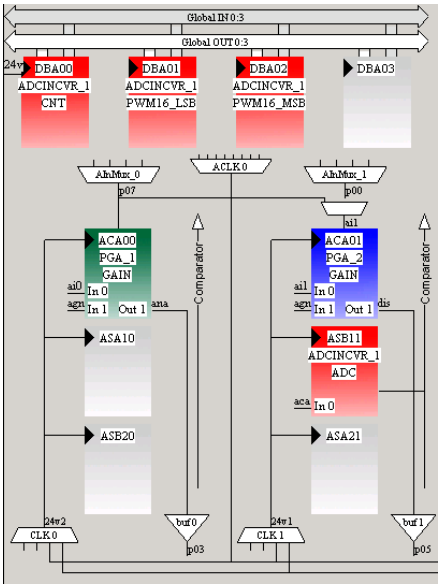


Figure 9: Motion Detector User Module Placement

The ADC requires an 8 MHz (24V1) clock. The signal is sampled, digitized and available to the CPU for detecting motion.

Signal Detection

The data from the ADC is not very useful by itself. Something has to be done with it. Figure 10 shows a block diagram of the output path:

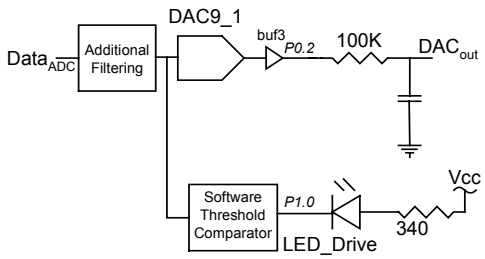


Figure 10: Motion Detector Output Block Diagram

For this project, the ADC data requires some additional digital filtering before being fed to **DAC9_1**. The DAC output allows the processed data to be viewed with an oscilloscope. The reader can view this output for motion at different ranges to get a feel for its performance. The CPU also analyzes the data to light an LED when the signal exceeds some threshold window. An LED was selected to simulate an opto-coupled triac. It could have easily driven a relay coil. The LED is programmed to light for 5 seconds after the threshold has been detected. The User Module placement, pin selection, and component values can be seen in Appendix A.

Figure 11 shows the additional filtering requirements:

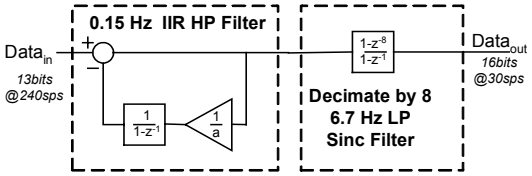


Figure 11: Motion Detector Digital Filtering

The high-pass filter is to remove any offset error from the ADC or final gain stage. The high-pass filter is an IIR filter developed in Application Note AN2099, "*Single Pole IIR Filters.*" With a 240 sps data rate, the filter's roll-off frequency is 0.15 Hz. The sinc filter is just 8 samples added together to produce a single output. It is the nature of a sinc filter that the roll-off frequency is 22% of the output rate or 6.6 Hz. Eight accumulated samples take the data range from 13 bits to 16 bits. However, the resolution enhancement is only the square root of the decimation factor (8) or 1½ bits. There may be 16 bits of resolution, but the lower 1½ bits are noise.

Now for the Code

Example Code 1 is the code required to control this project:

```
//--------------------------------------------
// Application Note AN2105
//
// PIR Motion Detector
//
// This project demonstrates a PSoC implemented
// PIR motion detector.
// Application Note AN2105 discusses the PIR
// theory of operation and also lists the
// necessary external components.
//
// Copyright 2003
// Cypress MicroSystems
//--------------------------------------------

#include "IIRfilters.h"
#include "AN2105api.h"

#define LEDPORT  PRT2DR
#define LEDOFF   0x01
#define LEDON    0x00
#define DEBUG    1
#define SECONDS  5
#define DEADTIME 1

int iTotalData, iTemp;
extern unsigned char cTimer;
unsigned char i;

void main()
{
   //Start UMs, init IIR Filter, LED off
   DAC9_1_Start(DAC9_1_FULLPOWER);
          PGA_1_Start(PGA_1_HIGHPOWER);
          PGA_2_Start(PGA_2_HIGHPOWER);
          ADCINCVR_1_Start(ADCINCVR_1_HIGHPOWER);
          SimpleHighPassInit(0);
          LEDPORT = LEDOFF;
          cTimer = 0;

   //Enable REFMUX in ACA01 to output AGND
          PGA_2_GAIN_CR2 |= 0x14;

          //Enable the Sleep Timer
          M8C_EnableIntMask( INT_MSK0, INT_MSK0_Sleep
);

          M8C_EnableGInt;

          //Setup A/D for continuous samples
          ADCINCVR_1_GetSamples(0);

          //Main loop
          while(1){
      iTotalData = 0;
      for(i = 0; i < 8; i++){
         //Obtain A/D sample, send it through
         //the filter, and decimate by eight.
         ADCINCVR_1_ClearFlag();
         while(ADCINCVR_1_fIsDataAvailable() == 0);
            iTemp = ADCINCVR_1_iGetData();
            iTemp = iSimpleHighPassFilter(iTemp);
            iTotalData += iTemp; //decimate FIR
          }

      //Check if the decimated output is
      //outside the threshold. If so, turn on
      //the LED and enable the Timer delay
      if((iTotalData < -1536)||(iTotalData > 1536)){

         //Setup the 5 second "ON time" for the
         // LED. Sleep interrupt times this.
         // Allow it to be retriggered, except
         // during dead time just after the LED
         // is turned off.
         if(cTimer > DEADTIME){
            LEDPORT = LEDON;
            cTimer = SECONDS * 8 + DEADTIME;
         }
      }
      //Clip at upper & lower limit for FIR
      if (iTotalData > 16383)iTotalData = 16383;
      if (iTotalData < -16383)iTotalData = -16384;

      //The following code is not needed for
      // functionality and is placed only so
      // that the internal digitized signal can
      // be viewed for debugging with the DAC.
      #if (DEBUG)
         iTotalData >>= 6;  //adjust for DAC range
         DAC9_1_WriteStall(iTotalData);
      #endif
   }
}
```

Code 1

The following is done at start up:

- All the User Modules are started.
- The high-pass filter is initialized.
- The LED drive is turned off.
- AGND is connected to **buf1** via the testmux in PGA_2.
- The sleeptimer is enabled for 8 Hz interrupts.
- The global interrupt is enabled.
- The ADC is started for continuous sampling.

The control loop waits for valid ADC values and then passes them through the high-pass filter. Eight values are accumulated. If this accumulated value exceeds a threshold of +/- 1536 (0x600), then the LED drive is enabled and a variable, **cTimer**, commands the sleeptimer interrupt to keep LED drive on for the next 40 sleeptimer interrupt cycles. The threshold could be changed to adjust the sensitivity and range of the detector. The data is then fed to the DAC.

Example Code 2 is the sleeptimer interrupt:

```
;;-----------------------------------------------
;   Sleep Timer Interrupt
;
;   This file decrements _cTimer which is used
;   in main.c to count a five second "ON time"
;   for the LED.
;;-----------------------------------------------
include "m8c.inc"

LEDPORT:  equ       PRT2DR
LEDOFF:   equ       0x01
DEADTIME: equ       0x01

area bss
          _cTimer:  blk       1
area text

export SleepTimer
export _cTimer

SleepTimer:
          cmp       [_cTimer], 0xFF  ;if timer FF then
done
          jz        .Exit
          dec       [_cTimer]        ;still counting
          cmp       [_cTimer], DEADTIME
          jnz       .Exit
   ;if less than DEADTIME, then turn off LED
          mov       reg[LEDPORT],LEDOFF

.Exit:
          reti
```

Code 2

The sleeptimer interrupt has four states defined by cTimer. They are:

cTimer = 0xff
This means the timer reached terminal count and stops counting.

cTimer = DEADBAND
The pulse stretching period is over, turn off the LED.

cTimer < DEADBAND
This means the LED is off. The cTimer continues to countdown. The main program uses this state to ignore any threshold crossing that happens just after the LED is suddenly turned off. This prevents the sensor circuit from retriggering itself.

cTimer = Anything Else
This means cTimer is being used to count interrupt cycles. Decrement cTimer.

To get access to this interrupt, the following section of code in "*boot.tpl*"…

```
    org 3Ch         ;Sleep Timer Interrupt Vector
        `@INTERRUPT_15`
        reti
```

… must be changed to:

```
    org 3Ch        ;Sleep Timer Interrupt Vector
        ljmp SleepTimer
        reti
```

Your project is ready to go. When operating this project, disconnect from the ICE and run from external power. Because the input signal is amplified by 256, the sensor is too sensitive to the power-supply noise to run while the ICE operates.

Operation

Tests of this circuit indoors show that when fitted with a Fresnel Lens, human motion can be reliably detected as a distance of 60 feet.

Wanna Build One?

The PIR detector, Fresnel Lens, and a machined enclosure made to mount the sensor and lens at the correct focal distance can be purchased from Glolab Corp. They can all be had for less than $20.

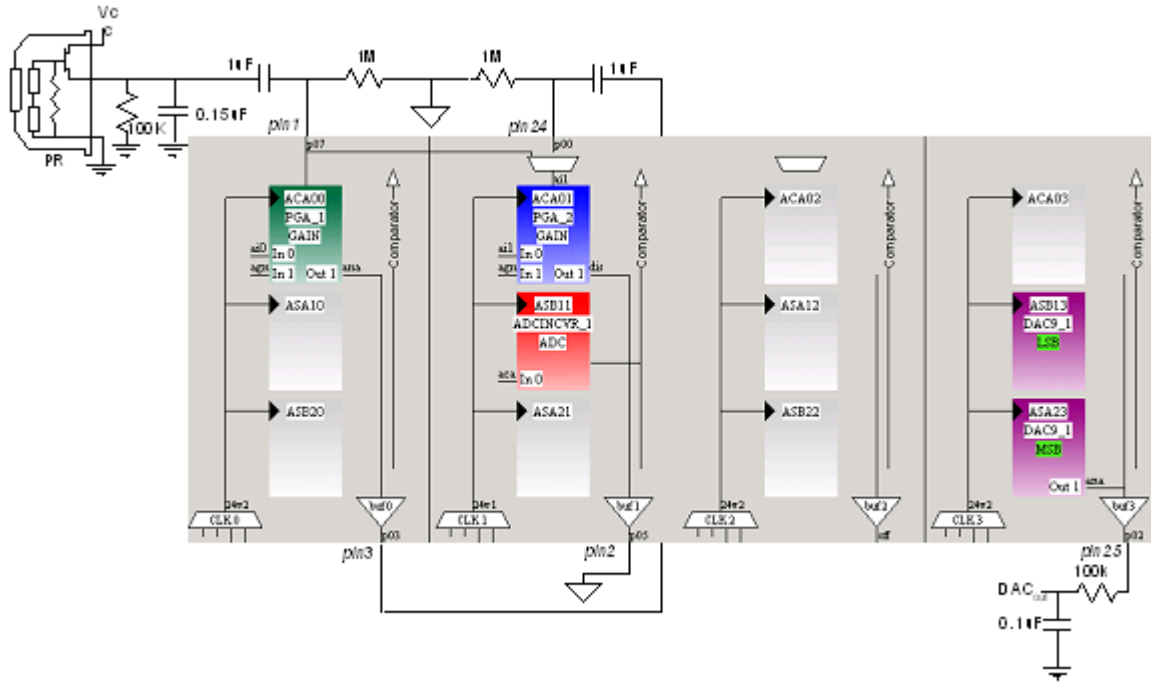
Glolab is a company that supplies electronic kits and components. They offer schematics of everything they sell and are some of the nicest people to work with. They can be reached at http://www.glolab.com.

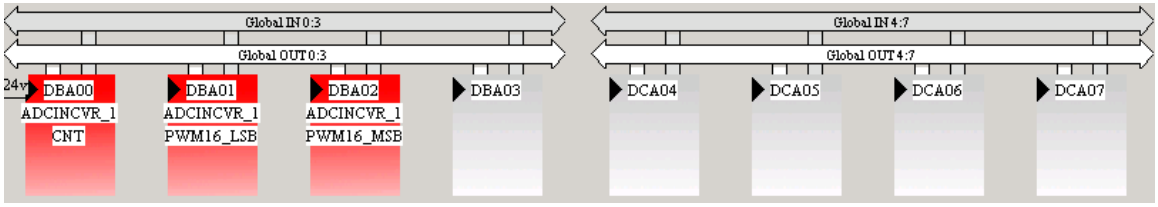
Conclusion

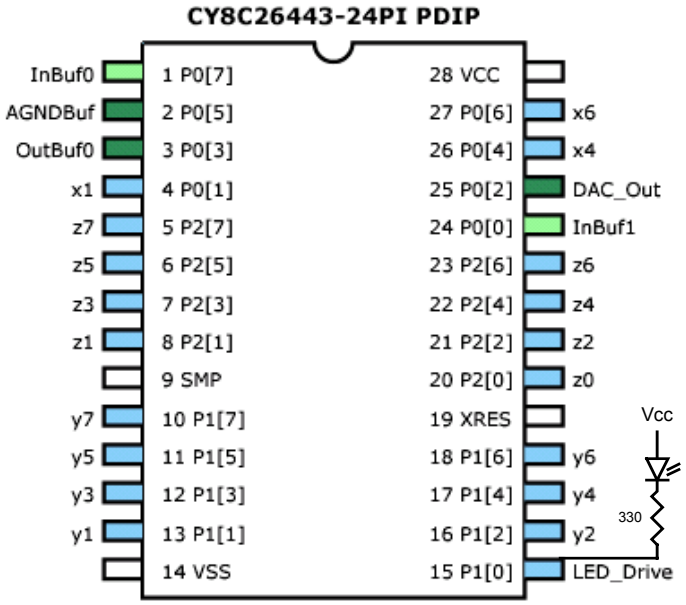
The PSoC architecture makes for simple inexpensive construction of a PIR motion detector. This can either be a product's primary function or an inexpensive feature enhancement for your present product design. How about an oven controller that can determine if a toddler is in the vicinity of the stove. Or maybe a baby monitor that can determine if a child has gotten out of bed. The possibilities are unlimited.

Appendix A

PSoC User Module Placement and Pin Interface Schematic







Cypress MicroSystems, Inc.
2700 162nd Street S.W. Building D
Lynnwood, WA 98037
Phone: 800.669.0557
Fax: 425.787.4641
http://www.cypressmicro.com/ / http://www.cypress.com/aboutus/sales_locations.cfm / support@cypressmicro.com