## Smart Lighting System

## Author: Ben Ellis

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Objectives**

1) Utilize interrupts, I/O, and timers on the MSP432

2) Implement a power efficient lighting system that turns on street lights when vehicles are in the area

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Overview**

This application has a real-world implication. It simulates a motion detecting lighting system to reduce power consumption by only turning on street lights when cars are in the area. When no cars are in the area a timer will begin and once the timer as reached a specified time interval the street lights will be turned when no cars are in the area.

**Description**

This application functions with the interrupt, I/O, and timer technologies built in to the MSP432. Using these technologies, a power efficient lighting system can be created using a PIR sensor as the input and an LED as the output. When the PIR sensor detects motion, it sends a digital 1 signal to the MSP432 which triggers an interrupt to turn on the LED. When the sensor no longer detects motion a timer begins and counts for approximately 5 seconds. If the sensor detects motion during those five seconds the timer is reset and begins counting again from the start with the LED remaining on. If the timer reaches the 5 second time frame an interrupt is triggered and a check to see if the sensor is on will be evaluated and if the sensor is providing a digital 0 then the LED will be turned off. This system design reduces power consumption by turning on street lights only when cars are in the area.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Equipment**

This application requires the following parts:

1 MSP432 LaunchPad with USB cable

3 Female to Female Jumper Wires

1 PIR motion sensor \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Schematic/Peripheral Connections**

**Diagram

Description automatically generated**

The output port of the PIR sensor labeled OUT connects the pin P1.5 on the MSP432 while the port GND of the sensor connects to a GND pin on the board and 5V on the board connects to the port labeled VCC on the sensor. The LED at pin P1.0 is built-in on the board so there is no need to include any additional wiring.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Programming Helps**

**#include** "msp.h"

**#define** TIMER\_PERIOD 15000000 //produces a 5 second count interval

/\*\*

\* This program is designed to reduce power consumption of street lights

\* by using I/O motion sensors, timers, and interrupts to detect when a

\* car is in the area and then turn off the street light after a

\* five second interval and when the sensor is outputting a low voltage

\*/

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

P1->DIR |= BIT0; //set pin P1.0 as output

P1->DIR &= ~BIT5; //set pin P1.5 as input

P1->OUT &= ~BIT0; //turn pin P1.0 off

P2->DIR |= BIT2; //set up pin P2.2 as output

P2->OUT &= ~BIT2; //turn pin P2.2 off

TIMER32\_1->LOAD = TIMER\_PERIOD;//set the timer32 count value to cause an

//interrupt five seconds after the sensor

//produces a low zero from no traffic.

TIMER32\_1->CONTROL = 0XE2; //turn on timer and config control reg below

//ENABLE: 1 start the timer

//MODE: 1 periodic mode

//IE: 1 enable interrupts

//PRESCALE: 000 no clock divider

//SIZE: 1 for 32-bit timer

//ONESHOT: 0 wrapping mode

P1->IE |= BIT5; //enable the interrupt for pin P1.5

P1->IFG &= ~BIT5; //clear the interrupt flag for pin P1.5

NVIC->ISER[0] = 0x02000000; //enable Timer32\_1 interrupts

NVIC->ISER[1] |= 0x8; //enable port 1 interrupts

\_enable\_interrupts();

**while** (1); //wait for an interrupt

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Interrupt handler for port 1. When

\* the PIR sensor detects car traffic/motion

\* an interrupt is triggered to turn on

\* the street light or in our case the led

\* at pin P1.0. This interrupt resets the five

\* second interval

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **PORT1\_IRQHandler**(**void**)

{

uint32\_t status;

status = P1->IFG; //get the interrupt status for port 1

P1->IFG &= ~BIT5; //clear the interrupt for port 1, pin 5

**if**(status & BIT5) //there is traffic on the street

{

P1->OUT |= BIT0; //turn on the street light

TIMER32\_1->CONTROL = 0X00; //stop the timer

TIMER32\_1->CONTROL = 0XE2; //restart the timer

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Interrupt handler for the timer32 module.

\* When the five second time interval has been

\* reached this interrupt will be triggered

\* and if the PIR sensor is outputting a low

\* voltage then the street light will be turned

\* off

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**void** **T32\_INT1\_IRQHandler**(**void**)

{

TIMER32\_1->INTCLR = 0; //clear the timer interrupt flag

**if** ((P1->IN & BIT5) == 0x00) //There is no traffic on the street

{

P1->OUT &= ~BIT0; //turn off the street light

//P2->OUT |= BIT2; used for debugging purposes

}

}

**Test Plan and Results**

The test procedure for this application includes phase one of observing the result displayed on the LED on pin P1.0. If the LED turns on when motion is placed before the PIR sensor then it can be concluded that the interrupt for port 1 is working correctly and is interfacing as expected, with the sensor. If the LED is not turning on when motion is present in front of the sensor then the code needs to be examined carefully to first identify if the output pin for P1.0 is initialized correctly in the directional control register. If the LED is still not turning on then the IE and ISER control registers need to be examined to verify that the interrupts are enabled for port P1.5. The second phase includes evaluating the LED at pin P1.0 once more, but this time the LED needs to turn off when the sensor detects no motion and the timing interval has been reached. If the LED does not turn off then the first thing to be examined is the if statement inside the timer interrupt handler. If the LED is still not turning on then the blue LED at pin P2.2 is used to toggle on and off to verify the interrupt flag is being raised when the timing interval has been reached. If the blue LED is not being toggled on and off then the CONTROL, LOAD, and ISER control registers need to be examined to make sure the timer module was initialized correctly.

At the conclusion of the debugging process the system performed with a 100% functionality. When motion was detected the LED at pin P1.0 turned on and when the timer reached the 5 second time frame with no motion detected from the sensor the LED was turned off. When motion is constantly being detected in front of the sensor the LED remains on until motion is no longer detected and the timer reaches its limit.

I struggled the most with getting the interrupts to work correctly. I first wanted to get the LED to turn on correctly and that was easy but I wanted to get the LED to turn off and I tried using the same signal from the sensor but using a not gate and when the signal went low on the sensor I would have a high signal sent to another port to trigger an interrupt to turn off the LED. I ended up putting that aside and focused on the timer32 module. Reading from the textbook was a little challenging, but after reading it enough times I realized that everything was done using the LOAD and CONTROL registers with an interrupt handler function. The timer interrupt helped clear up my problem of turning off the LED and the system performed as expected.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Project Video Explanation**

[**https://youtu.be/Rz51Z6EEenM**](https://youtu.be/Rz51Z6EEenM)

**Project Video Demonstration**

[**https://youtu.be/vnqcRLIDypI**](https://youtu.be/vnqcRLIDypI)