15-348: Embedded Systems, Fall 2022

Homework 4: Invisible Control

Due: Thursday, September 29th, 2022 by 10:00pm

Part	**Quantity**
Launchpad	1
TSOP382	1
$0.33\mu F$ Capacitor	1
220Ω Resistor	4
330Ω Resistor	1
LED	4

Introduction

This is a programming homework. You must submit your code to Gradescope by **Thursday, September 29th, 2022 by 10:00pm**. Only submit the files indicated below. You should also demo your code to the one of the course staff members on or before Monday, October 3rd during one of the available slots. You can only demo once. More instructions about the demos will be announced during class and on Piazza.

Learning Objectives:

Learn how to use input capture to measure pulse widths.

Getting Started:

1. For each task, create a new project in CCS, as explained in the tutorial on the website.
2. Submit only the source code files that you modified (i.e., not the entire project, binaries, etc).

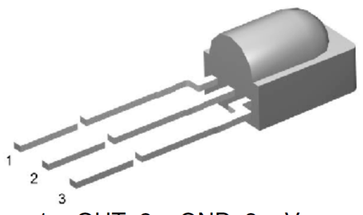
Some Important Notes:

1. You should read the entire assignment before starting. This will help you predict what your final circuit will look like, what sort of functionality your software will need, etc.
2. We expect that your implementation exactly matches the specifications indicated below.
3. Although we might be lenient with unspecified behaviors, you should justify your implementation choices.
4. If you are unsure about the specifications, you should ask questions on Piazza or directly to the course staff.
5. Reading, understanding, and eliciting the requirements and specifications is also part of the assignment.

Task 1: Detect an IR Signal (20 pts)

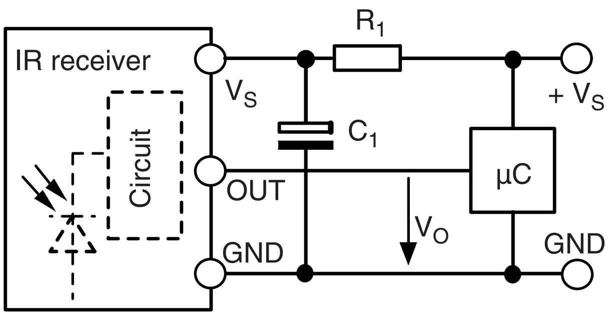
In this task, you will write a program that detects the signal emitted by a button press on a remote control using an infrared sensor. You can see full specification at http://www.sparkfun.com/datasheets/Sensors/Infrared/tsop382.pdf.

The infrared sensor has three pins as depicted below:



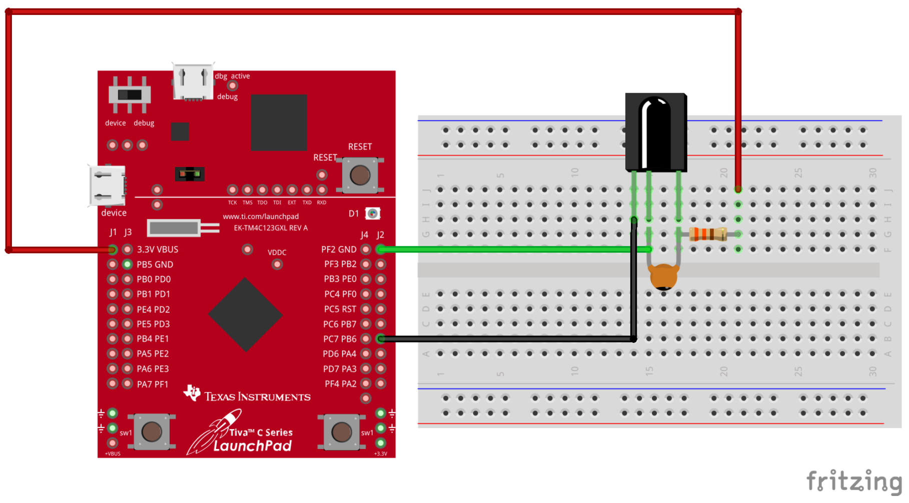
1 = OUT, 2 = GND, 3 = V_S

And the data sheet provides the following diagram for how to interface the sensor to a microcontroller:



Resistor R1 should be 330 Ω and the capacitor C1 of $0.33\mu F$.

Here is a diagram of how that wiring could look:



Next, program the controller so that it can detect the signal from the receiver. Use an LED to indicate the reception of the signal. The LED should light up every time a button in the IR remote control is pressed. The IR receiver should be connected to PB6, and the signal must be detected using the input capture interrupts.

Task 2: Decoding the IR Signal (30 pts)

In this task, you should write a program that decodes the button pressed on a remote control using an infrared sensor. The IR sensor produces a sequence of pulses for each button press from the IR remote control. The pulse widths determine which button was pressed on the remote. You can decode the signal by measuring the pulse widths as follows.

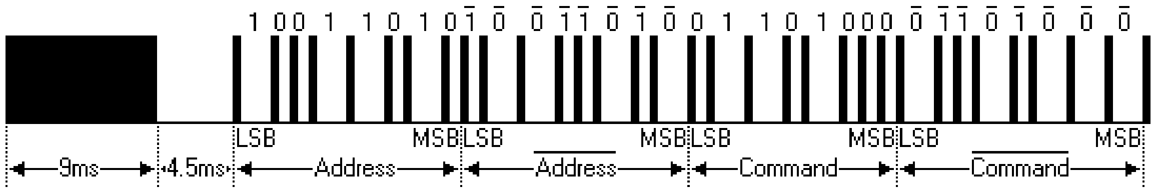
The remote uses the NEC protocol to encode the button presses: https://www.sbprojects.net/knowledge/ir/nec.php

On the receiver side, by default, the IR signal is high. The encoding of the IR command produces the following pattern at the receiver (IR sensor) side:

1. A header = 9000μs LOW followed by 4500μs HI
2. After the header, 32 bits are transmitted. Each bit has a specific pattern according to its value:
   - A BIT ZERO = 560μs LOW followed by 560μs HI
   - A BIT ONE = 560μs LOW followed by 1690μs HI

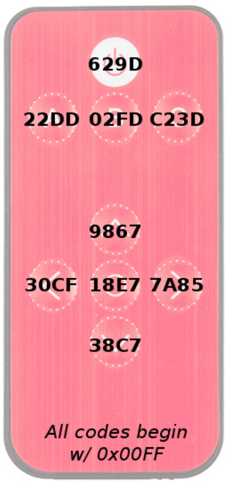
The 32 bits are interpreted as follows. The first 16 bits correspond to an address and the last 16 bits correspond to a command. Address and Command are transmitted twice. The second time all bits are inverted (bit-wise complement) and can be used for verification of the received message.

Here is an example of a command packet:


1 0 0 1 1 0 1 0 $\overline{1}$ $\overline{0}$ $\overline{0}$ $\overline{1}$ $\overline{1}$ $\overline{0}$ $\overline{1}$ $\overline{0}$ 0 1 1 0 1 0 0 0 $\overline{0}$ $\overline{1}$ $\overline{1}$ $\overline{0}$ $\overline{1}$ $\overline{0}$ $\overline{0}$ $\overline{0}$
LSB
MSB
LSB
MSB
LSB
MSB
LSB
MSB
9ms
4.5ms
Address
$\overline{\text{Address}}$
Command
$\overline{\text{Command}}$

A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9000μs LOW pulse followed by a 2250μs HI and a 560μs LOW. In this task, you can ignore (discard) the repeat commands.

The 32 bits that the IR remote transmits for each button are the following:



Button	Code
BUTTON_POWER	0x00FF629D
BUTTON_A	0x00FF22DD
BUTTON_B	0x00FF02FD
BUTTON_C	0x00FFC23D
BUTTON_UP	0x00FF9867
BUTTON_LEFT	0x00FF30CF
BUTTON_CIRCLE	0x00FF18E7
BUTTON_RIGHT	0x00FF7A85
BUTTON_DOWN	0x00FF38C7

Note that the IR remote codes shown above begin with 00FF. That is an address of 00 and its complement, FF. The code 7A85 corresponds to the > push button. This represents the command 7A, followed by its complement 85.

To show that your controller can successfully decode the IR button press, use LEDs to represent each IR button press. Use 4 LEDs as a 4-bit number to indicate which button was pressed. For instance, show 0x1 for POWER, 0x2 for `BUTTON_A`, ... , 0x9 for `BUTTON_DOWN`. (Follow the order of the table above.)

Additional Guidance

Book Bug

Example 6.5 in the course textbook is very useful for this homework, but it has an annoying bug: It incorrectly disables and enables TimerB. The proper mask for enabling/disabling both TimerA and TimerB is *not* `0x03`, it is `0x0101`. See the entry for `_CTL_R` in Table 6.1 to understand why. (And don't waste 2 hours of your life like I did trying to figure out why the example code doesn't work...)

General Tips

- Work incrementally, testing each small piece of functionality as you go. For example, your first step might be simply seeing if you can toggle an output pin from 1 to 0 and see the voltage change on the multimeter.
- If you spend too long debugging something, and can't find the problem, start over. Rewire your circuit and rewrite the code from scratch. It shouldn't take you that long the second time, and you may be surprised to find that it works.