

## *Infrared Learner (Remote Control)*

By: Mehmet Zeki SONMEZ

**Associated Project:** Yes

**Associated Part Family:** CY8C25xxx, CY8C26xxx

### Summary

This project allows you to copy an IR signal from a remote controller to the RAM address of PSoC and then transmit it later. It will work for most remote control applications such as TVs, VCDs, DVDs, etc. Once you understand, you'll be able to teach different remote-controller functions to the IR learner and combine them in one device. In this project the PSoC device can learn one function. For example, TV OFF.

### Required Parts:

- **1 x IR Receiver Module (38 kHz):** One of the following can be used. ( $f_o=38$  kHz)
  - Vishay / Telefunken TSOP1238 or TSOPxx38 (Tested, OK) (**Recommended**)
  - Sanyo SPS-440 (Tested, OK)
  - Sharp IS1U60 / IS1U60L (Not Tested, Probably OK)
  - GP1U58X (Not Tested, Probably OK)
  - Any Other Module with  $f_o=38$  kHz and *inverted output*
- **1 x NPN Transistor:** General Purpose NPN Transistor (ex: BC556, 2N2222)
- **1 x LED**
- **1 x IR LED**
- **3 x 1 kOhm Resistor, 1x100 Ohm Resistor**
- **1 x 100n CAPACITOR**
- **2 x PUSH BUTTON or SWITCH**

### Introduction

There are two main stages in this project. The first of these stages is capturing/copying the signal and the second is transmitting the captured signal.

Before explaining these stages, the theory of operation will be described.

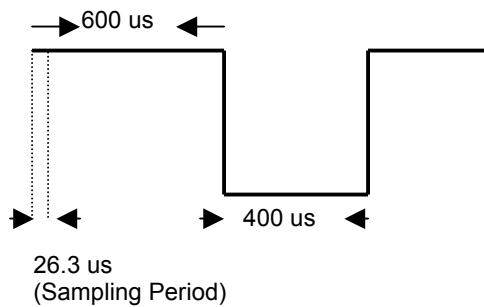
The IR receiver module produces a high level voltage (5V) when there is no incoming signal. When a signal has just come in, the IR receiver module will produce the same but inverted output of the incoming signal. Note that this signal has to operate within the defined limits of IR receiver module specified in the data sheet. Generally, TV, DVD, VCD, etc. remote controllers are appropriate with the IR receiver module.

The frequency of the IR signals from remote controllers is generally in the range of 38 kHz. Transmitting a 38 kHz signal in some limits (generally 300-800 microseconds) will produce a low level output (0V) in the receiver module.

The objective is to capture the low level (0V) and high level (5V) signals and then transmit them to the target device (TV) with a frequency of 38 kHz.

### Capturing IR Signal (First Phase)

The objective of this phase is to count every small duration (26.3 microseconds in this project) of the incoming signal until it changes its state. Then store the value of the counter to the RAM address.



**Figure 1: Example Signal Received from the IR Receiver Module**

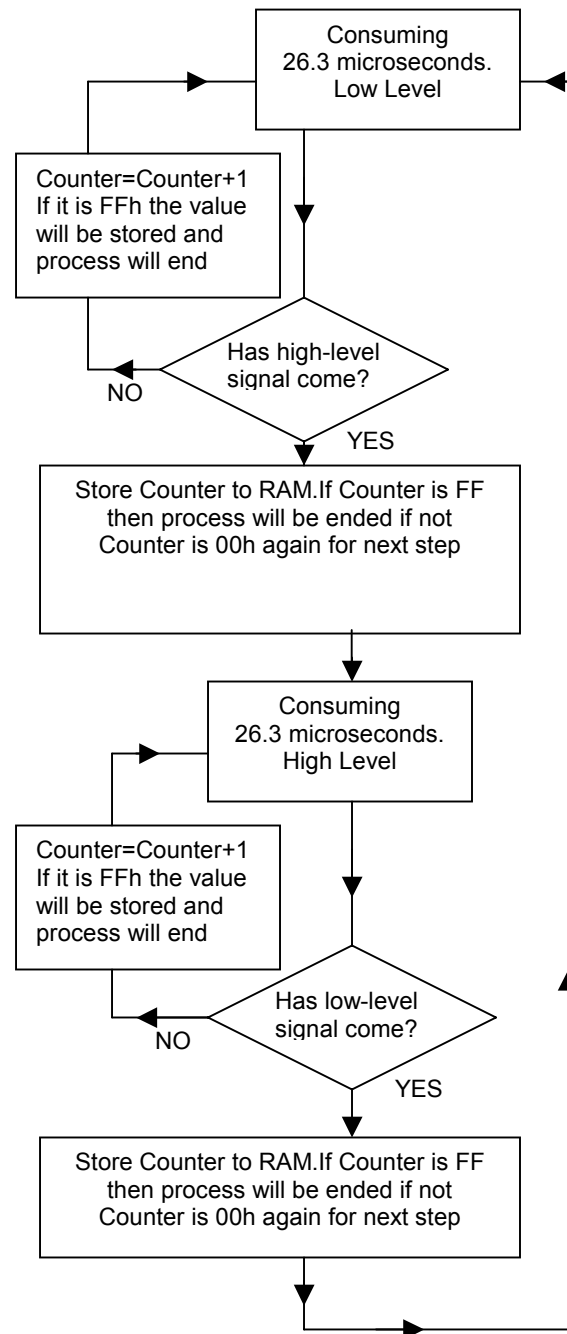
We should see the counter value at the end of the high level signal to be  $600 \text{ us} / 26.3 \text{ us} = 22$ . The counter will keep the value (22) = 16h and store it to the RAM address.

We should see the counter value at the end of the low level signal to be  $400 \text{ us} / 26.3 \text{ us} = 15 = 0Fh$ .

If the counter reaches FFh, the capturing process will end. If the counter is FFh, the duration of low level or high level signal will be  $= 255 * 26.3 \text{ us} = 6.7 \text{ milliseconds}$ , which is enough for us to end the process because standard TV remote controls don't send a signal with a pulse width of 6.7 milliseconds or greater.

A detailed description for assembler codes is included in the assembler file (*main.asm*) in the project.

If you teach the IR signal and want to teach another signal you must reset the system by pushing on the button that is connected to the X<sub>res</sub> pin. See for the schematic at the end of this document for details.



**Figure 2: Process Flow**

### Captured Signal at RAM Address

The captured signal will be stored in RAM addresses by using the Index register (X). Consider the following incoming signal from the IR receiver module:

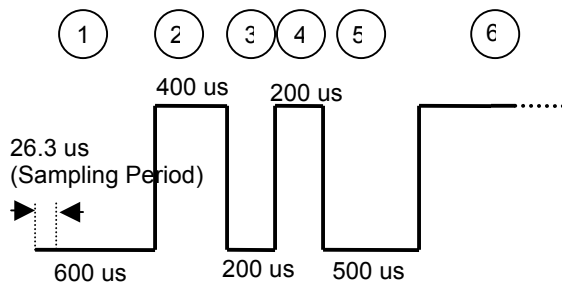


Figure 3: Incoming Signal

- |   |   |  |
|---|---|--|
| ① | → | Counter = 600 us / 26.3 = 16h                                    |
| ② | → | Counter = 400 us / 26.3 = 0Fh                                    |
| ③ | → | Counter = 200 us / 26.3 = 07h                                    |
| ④ | → | Counter = 200 us / 26.3 = 07h                                    |
| ⑤ | → | Counter = 500 us / 26.3 = 13h                                    |
| ⑥ | → | Counter = FFh (pulse width > 6.7 ms incoming signal has stopped) |

Figure 4: Counter Values

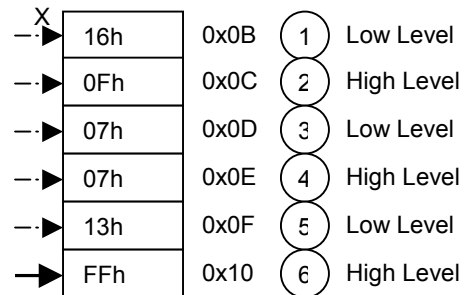


Figure 5: Address Writes

As shown in Figure 5, the first data of the incoming signal (first low level) is written to RAM address 0x0B(1). Then, the high level, then the low level... is written to the next RAM address until a value of FFh for the counter has occurred. This occurs when the incoming signal is stopped.

### Transmitting Captured Signal (Second Phase)

As described in the introduction section, logic 1 can be sent by 38 kHz pulses.

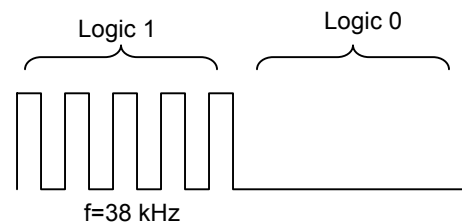


Figure 6: Logic 1, 38 kHz Pulses

38 kHz pulses can be achieved by simple code as follows (detailed description is in *main.asm*):

```

G_38KHZ:
mov     A,4h
mov reg [PRT2DR],A
mov     [COUNTER1],[COUNTER2] ;No mean
mov     [COUNTER1],[COUNTER2] ;No mean
mov     [COUNTER1],[COUNTER2] ;No mean
mov     A,0h
mov reg [PRT2DR],A
mov     [COUNTER1],[COUNTER2] ;No mean
ADD     [COUNTER1],8           ;No mean
dec     [TEMP2]
JNZ     G_38KHZ

```

The main objective of this stage is to create code that consumes 26.3 microseconds and repeat it until the value of the RAM address (the counter value stored in the previous stage) is zero. (We will decrement the value inside the RAM address until it is zero).

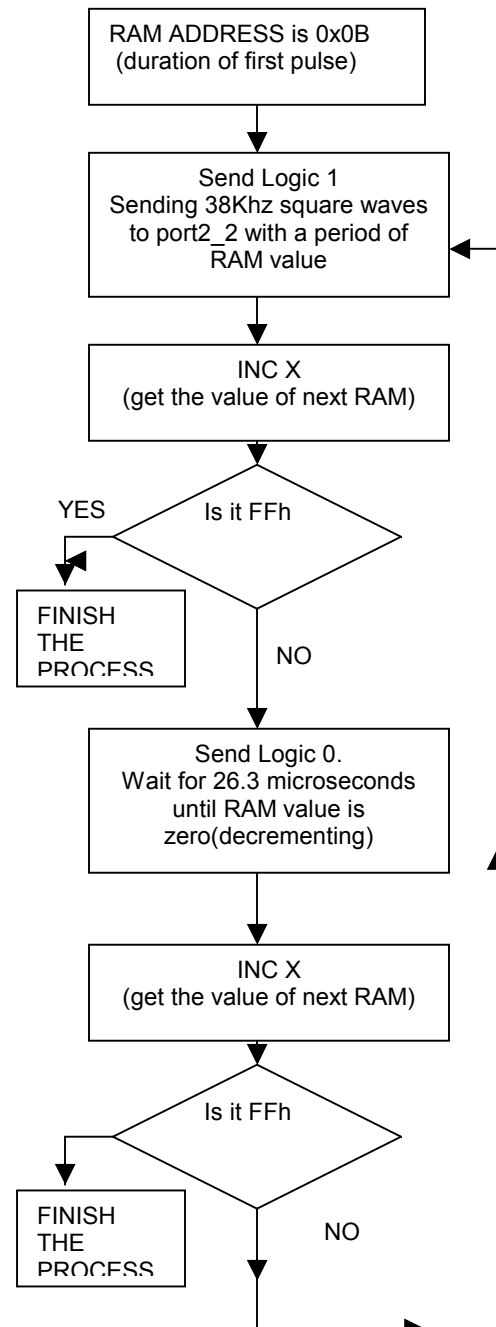
Because each G\_38KHZ loop lasts 26.3 microseconds, we do not need to add extra time consuming code when sending logic 1s. This is why we sample each signal with 26.3 microseconds. ( $1/38\text{KHZ}=26.3$  microsecond!)

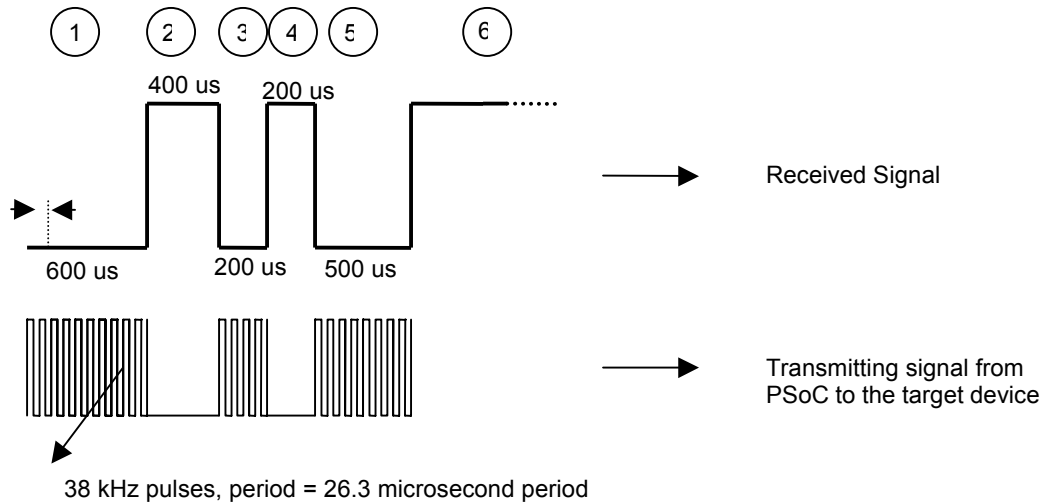
We will add time consuming code when sending logic 0's. (Actually we don't send anything; logic 0 is 0 volts. We make a delay of 26.3 microseconds.)

Note the following:

- Because the output of the IR receiver module is inverted, we will send logic 1 for duration of low-level values in the RAM and we will send logic 0 for duration of high-level values in the RAM. In brief, we send Logic 1 for low-level signals and we send Logic 0 for high-level signals.
- When you press the button connected to P1[0], the captured signal will be sent. (GPIO interrupt is used for this button.)

Figure 7: Process Flow





Note that the transmitting signal of the received signal is inverted because the output of the receiver module is inverted. Therefore the transmitting signal is the exact copy of the remote controller (teacher).

Figure 8: Transmitting Received Signal to Target Device

### About the Author

Name: Mehmet Zeki SONMEZ

Title: Yeditepe University: Electrical & Electronics Engineering Student & Student Assistant (last year).

Background: M.Zeki Sonmez is interested in design of both analog and digital (including microcontrollers) circuits. In the near future his area of interest will be RF Identification.

Contact: [zeki@sonmezticaret.com](mailto:zeki@sonmezticaret.com) or [mzsonmez@hotmail.com](mailto:mzsonmez@hotmail.com)  
Kanarya Bloklari.7.Blok D: 8  
Kayisdagi/ISTANBUL, TURKEY  
+90 532 413 70 50

Web Site: <http://electronicsclub.cjb.net>

## Schematic

### Configuration:

Configure System Clock to 3 MHz.

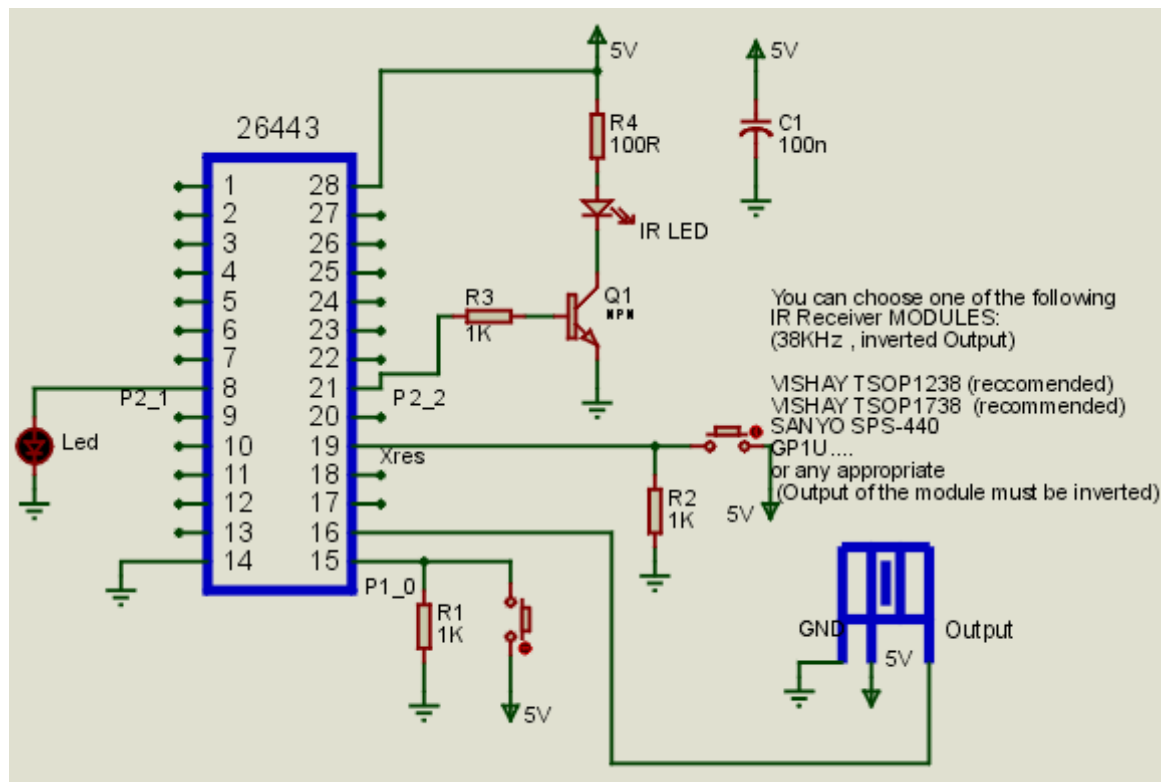
Configure Port1\_2 as input.

Configure Port1\_0 as input and **configure interrupt type as RISING EDGE in pinout view.**

Configure Port2\_1 as output.

Configure Port2\_2 as output.

- **1 x IR Receiver Module (38 kHz):** One of the following can be used. ( $f_o=38$  kHz)
  - Vishay / Telefunken TSOP1238 or TSOPxx38 (Tested, OK) (**Recommended**)
  - Sanyo SPS-440 (Tested, OK)
  - Sharp IS1U60 / IS1U60L (Not Tested, Probably OK)
  - GP1U58X (Not Tested, Probably OK)
  - Any Other Module with  $f_o=38$  kHz and inverted output



Cypress MicroSystems, Inc.  
22027 17th Avenue S.E. Suite 201  
Bothell, WA 98021  
Phone: 877.751.6100  
Fax: 425.939.0999

<http://www.cypressmicro.com/> / [http://www.cypress.com/aboutus/sales\\_locations.cfm](http://www.cypress.com/aboutus/sales_locations.cfm) / [support@cypressmicro.com](mailto:support@cypressmicro.com)

Copyright © 2002 Cypress MicroSystems, Inc. All rights reserved.

PSoc™ (Programmable System on Chip) is a trademark of Cypress MicroSystems, Inc.

All other trademarks or registered trademarks referenced herein are property of the respective corporations.

The information contained herein is subject to change without notice.