**Infrared Remote Control**

This document is written for those who new to infrared transmission. Specially, for user of MSP430G2xx microcontroller.

**What is Infrared remote control (IRC)?**

Infrared control is way that you can control your everyday electronic devices such as: TV, air conditioner, projector, etc using infrared light. IRC has two LEDs, one emit infrared light while the other receive light from another source. The information sent or received by infrared LEDs is in HEX form and encoded by the length of high volt level (VCC or mark) or low volt level (GND or space).

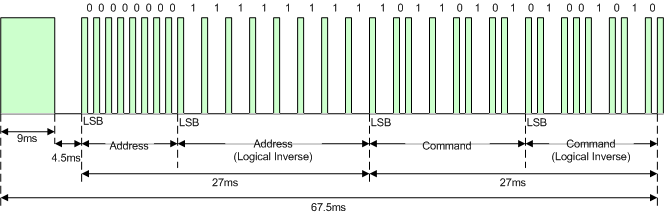
**A brief about the theory**

So you know what is sent between the two LEDs but the important question is How the information is sent?

The message bits are depended on the producer (LG, Panasonic, Samsung, etc) but it usually has three parts: Start bits, Data, End bits. When there is no message, it stays in IDLE state (VCC). For easier understanding, let divide into two parts (receive and transmission) and use NEC protocol to demonstrate how information is received.

NEC protocol

*Transmission process:*

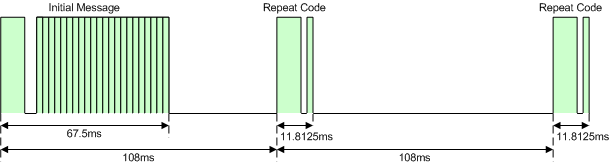
   
Figure 1. Example message frame using the NEC IR transmission protocol.

The two periods (9ms, 4.5ms) are the Start bits. The following periods are the information it contain: Address, Address Inverse, Command, Command Inverse. The End bit is the final pulse (VCC).

Now, have a closer look into the Data:

* Logical '0' – a 562.5µs pulse mark (VCC) followed by a 562.5µs space (GND)
* Logical '1' – a 562.5µs pulse mark (VCC) followed by a 1.6875ms space (GND)

Repeat code

   
Figure 2. Example repeat codes sent for a key held down on the transmitting remote controller.

If the key on the remote controller is kept depressed, a repeat code will be issued, typically around 40ms after the pulse burst that signified the end of the message. A repeat code will continue to be sent out at 108ms intervals, until the key is finally released. *However, in other protocols, the whole initial message may be repeated*.

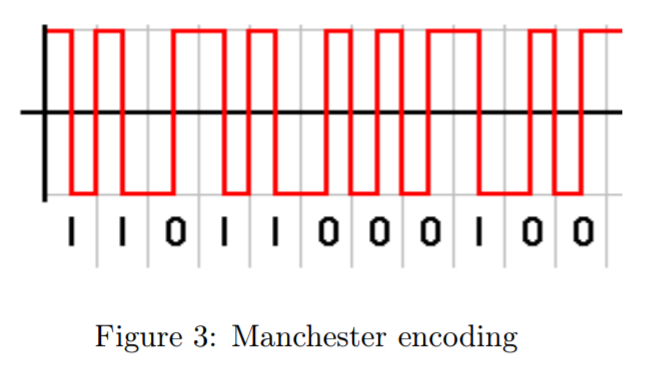
*Receiving process:*

For consistent we assume mark and space are high and low period of receiver.

The receiving process is quite similar to the transmission with: Start bits, Data, End bits and the duration of each mark and space are the same. But to avoid interference of infrared light from the environment, we transmit the space by a carrier wave with a frequency of 38KHz. Which meant no transmission for mark and 38KHz wave (with total length equal space length) for space (Receiver is in IDLE state, which mean mark and turn into space when receive carrier wave).

In summary, different producers will have different protocols but the frame remain the same with Start bits, Data, Stop bits. The differences are:

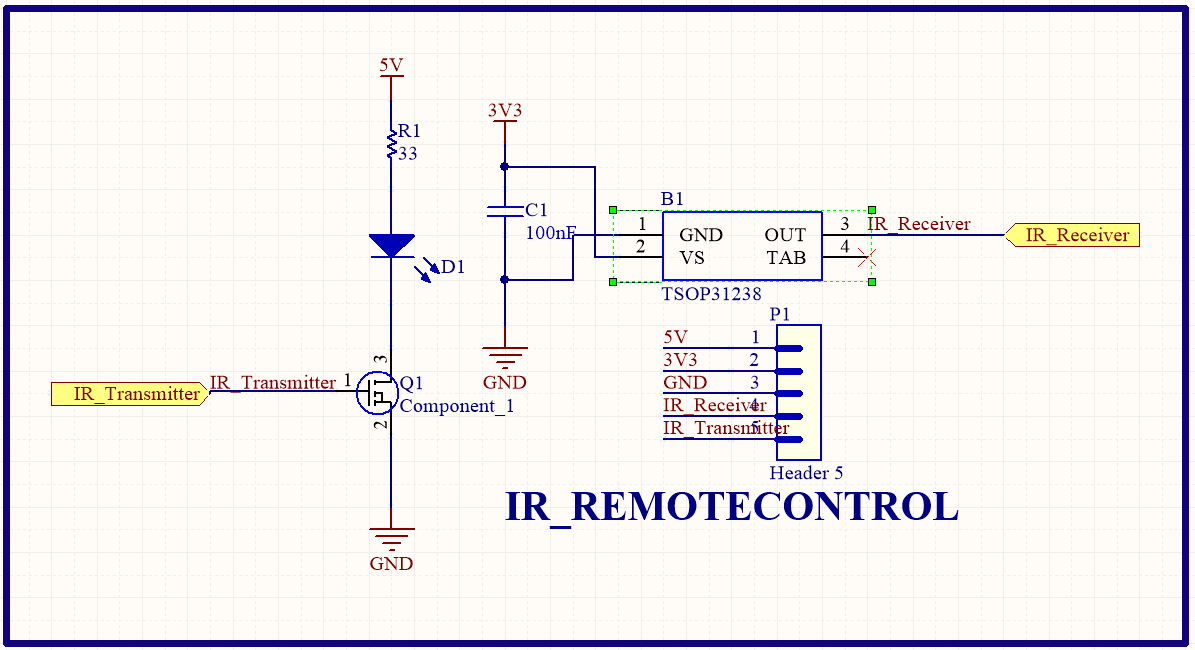
* How the Start bits and Stop bits are defined.
* The length of mark and space for logical “0” and logical “1”.
* For R5 and R6 protocol, Manchester encoding is used. That mean “1” starts with a mark and “0” starts with a space as shown on figure 3.



**Hardware**

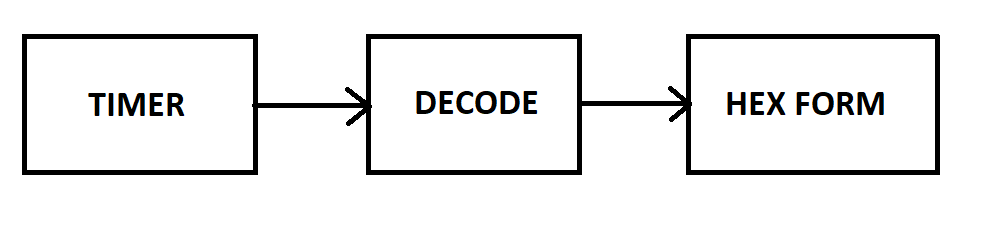
Hardware selection:

* Resistor: 33 ohm – 0.5W
* Capacitor: 100uF
* Emitting infrared LED
* Receiving infrared LED: TSOP31238
* MOSFET: LN2302LT1G



**Firmware**

Receive process:



* Timer is initialized and ready for receive message. Timer will get sample every 50us and count how many time (tick) of one status (mark or space). If the status changes, the code will put the number of tick (how long) into an array, reset the tick counter and count again. The process will continue until Stop bits are read. The number of tick is called raw code. Be noticed that after Stop bits, timer still continue taking sample but the information will not be record until Start bits are read again.
* The raw code then decoded into Hex form by using the producer protocol.

Transmission process:



* Hex data is encoded by producer protocol into mark and space.
* Timer’s Outmode function is initialized to generate PWM carrier wave at 38KHz to transmits the message.

For deeper understanding the code, you can try to debug or visit this website : http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html

**References**

[1] Ken Shirriff, “A Multi-Protocol Infrared Remote Library for the Arduino”, available: <http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html>

[2] Altium Tech Docs, “NEC Infrared Transmission Protocol”, available: <http://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol>

[3] EEVblog, “EEVblog #506 - IR Remote Control Arduino Protocol Tutorial”, available: <https://www.youtube.com/watch?v=BUvFGTxZBG8>

[4] huunho, “Nguyen ly va giai ma dieu khien tu xa hong ngoai”, available: http://codientu.org/threads/10714/