CALIFORNIA STATE POLYTECHNIC UNIVERSITY, POMONA
COLLEGE OF ENGINEERING

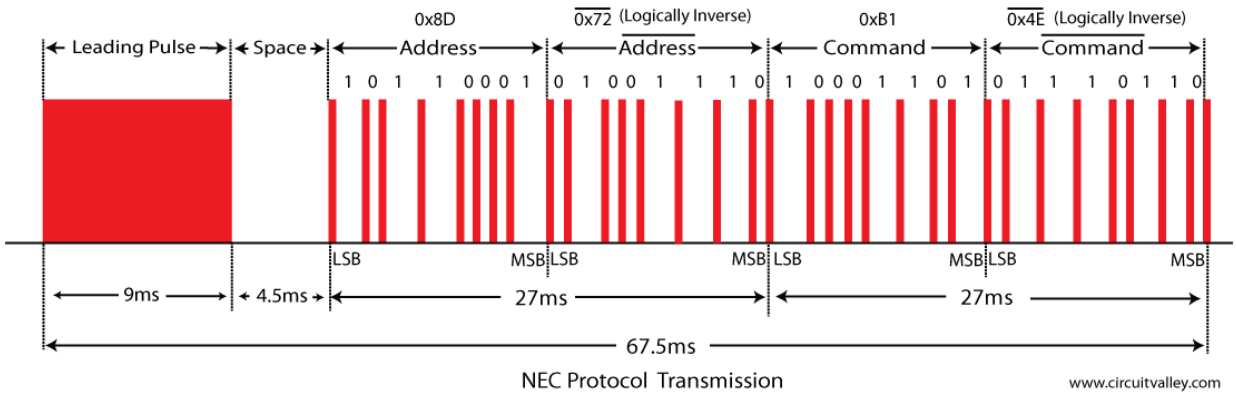
ECE 3301L Fall 2021 Microcontroller Lab Felix Pinai
Session 3

LAB 9: Interface to Infra Red Remote Control

The purpose of this lab is to introduce the student to the technique to interface to an Infra Red Remote Control.

Lab:

Below is a basic stream of pulse when a key on a remote control is pressed:



This is based on the NEC protocol transmission. Here are the basic steps on that protocol:

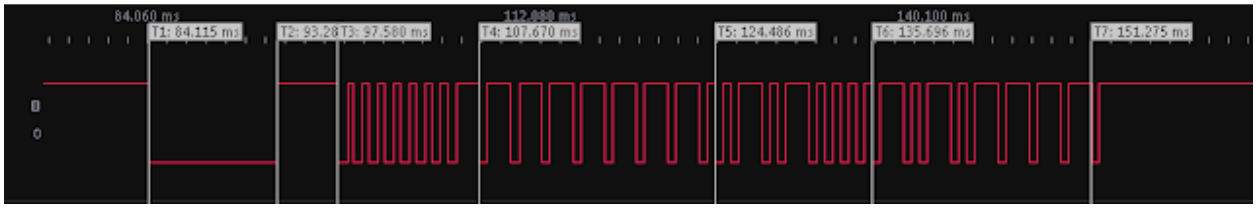
- A leading burst of pulses that lasts up 9 msec.
- A space period that lasts 4.5 msec.
- An 8-bit address field for the receiving device
- An 8-bit address inverse of the address
- An 8-bit command field
- An 8-bit inverse of the command field

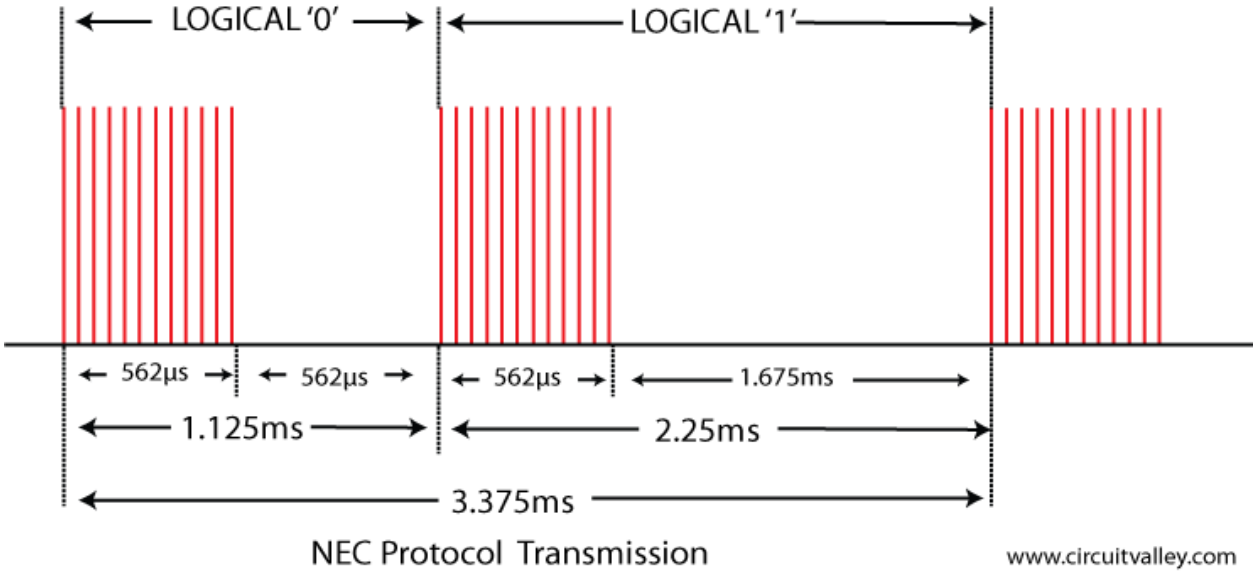
A total of 32 bits should be received after the 'Space 4.5 msec' field has been detected. To differentiate for each bit between the logic '0' or the logic '1', the following detection should be applied:

- A logic '0' should be detected when a 562-usec pulse burst is followed by a space period of 562 usec.
- A logic '1' should be detected when a 562-usec pulse burst is followed by a space of 1.675 msec

When the Infrared signal is received by an IR receiver, the output of that receiver should provide:

- The output stays at logic ‘1’ when no pulse is received
- The output goes to ‘0’ when the leading pulse is detected. The output should stay at logic ‘0’ for 9 msec.
- When the ‘Space’ is asserted, the output changes to logic ‘1’ for the period of 4.5 msec.
- Next a sequence of 32 bits should be transmitted. When the output changes from ‘1’ to ‘0’ and then back to ‘1’ with the same width of 562 usec, then a data logic ‘0’ is asserted. Otherwise, if the output goes from ‘1’ to ‘0’ for a period of 562 usec and then switches from ‘0’ to ‘1’ for a period of 1.675 msec, then a data logic ‘1’ is asserted.





PART 1)

To implement this protocol, we will need to use the Timer for the purpose to measure the elapsed time between two events and the input pin that has the edge-triggered interrupt feature (the INTx pins). Let us assume that we are going to use the INT0 pin to connect to the output of the IR receiver. That output is normally sitting on the logic '1' when no key is pressed on the remote control. A variable 'Nec_state' should be used and its initial value should be set to '0'. That is the first state of the IR decoding sequence. The program should use the pin INT0 to detect the different transitions of the IR signals and update the value of 'Nec_state' to keep track of the progress of the sequence. At the initial point, INT0 should be programmed to interrupt when the IR output transitions from high to low.

When an INT0 interrupt occurs, the first task is to read the content of the timer and save into a variable. The next step is to determine to update the state of 'Nec_state'

INT0_ISR:

- If Nec_State not in state 0 then read the 16-bit from TMR1H and TMR1L to store into variable Time_Elapsed. Clear both TMR1H and TMR1L
- Else, switch on Nec_State:

  Case 0:

  - Clear TMR1H and TMR1L
  - Program Timer 1 mode with count = 1usec using System clock running at 8Mhz
  - Enable Timer 1
  - Force bit count (bit_count) to 0
  - Set Nec_code = 0
  - Set Nec_State to state 1
  - Change Edge interrupt of INT0 to Low to High
  - return

  Case 1:

  - Check if Time_Elapsed value read is between 8500 usec and 9500 usec
  - If yes, force Nec_state to state 2 else do force_nec_state0()
  - Change Edge interrupt of INT0 to High to Low
  - return

  Case 2:

  - Check if Time_Elapsed value read is between 4000 usec and 5000 usec
  - If yes, force Nec_state to state 3 else do force_nec_state0()

- Change Edge interrupt of INT0 to Low to High
- return

Case 3:

- Check if Time_Elapsed value read is between 400 usec and 700 usec
- If yes, force Nec_state to state 4 else do force_nec_state0()
- Change Edge interrupt of INT0 to High to Low
- return

Case 4:

- Check if Time_Elapsed value read is between 400 usec and 1800 usec
- If yes,
    - shift left nec_code by 1 bit
    - Check if Time_Elapsed is greater than 1000 usec
    - If yes, add 1 to Nec_code else do nothing
    - Increment variable bit_count by 1
    - Check if bit_count > 31
    - If yes:
        - set nec_ok flag to 1
        - set INT0IE = 0
        - set Nec_State to 0
    - If not, do nothing
    - Set Nec_State to state 3
- If not, do force_nec_state0()
- Change Edge interrupt of INT0 to Low to High
- return

See the attached sample '.c' file to get started.

In the main routine, the variable 'nec_ok' will constantly checked. When it is set to 1, then a button on the remote has been detected. The TeraTerm console should show on the screen the long value of the Nec code (0x00FFccdd) where cc is the actual code and dd is the 1's complement of that code. The variable Nec_code1 stores that cc code.

PART 2)

Next, press all the buttons and collect all the codes and then place them consecutively into an array from the order of the buttons on the remote control. See example of array1 in the sample code.

The next step is to create another array called 'txt1' with the text values of each button. The text should be 3 characters long terminated with a '\0' to end the string. See the sample code.

A third array called ‘color’ should reflect the color of each button.

In the main code, after each button is pressed, use a for loop to look up for the corresponding definition of that button to determine the order of that button on the array in order to display that button on the LCD screen.

The last task is to output the color of the button onto one of the three RGB LEDs D1, D2, and D3 based on the column that button is located on the remote control. **The first 7 buttons are assigned to D1, the buttons from ‘+‘ to ‘2’ are associated to D2 and the last bottom seven bottom buttons are assigned to D3**. Don’t use if or case statement to select the color for the output. In addition, since the color black cannot be displayed, replaced it by the color WHITE.