NEC IR Transceiver



Content

[Introduction 3](#_Toc46100829)

[Transmitter 4](#_Toc46100830)

[Design 4](#_Toc46100831)

[Simulation 6](#_Toc46100832)

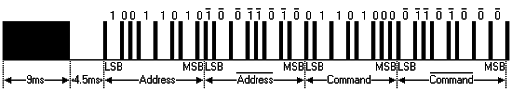
[Receiver 6](#_Toc46100833)

[Design 7](#_Toc46100834)

[Simulation 9](#_Toc46100835)

# Introduction

Infrared communication is frequently used for televisions and other appliances that require a remote controller. This projects contains a transceiver with an independent transmitter and receiver. Which are able to work separately and even without one of the two drivers. The protocol used is NEC from Sony. This protocol is widely implemented and makes this driver versatile.



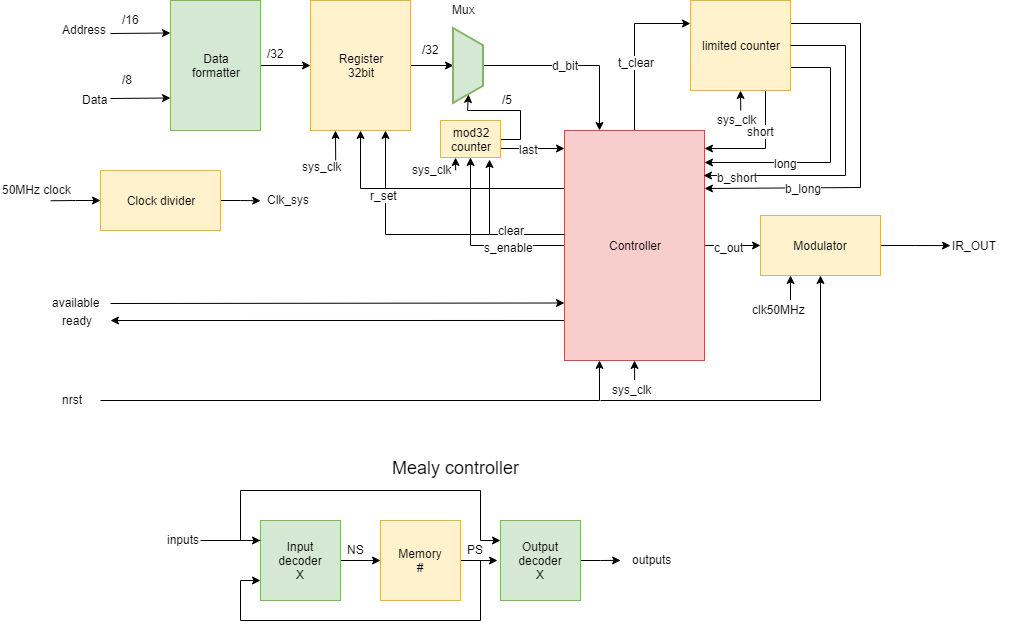
In the picture above there is one of the two NEC variants. This one uses an 8 bit address which is sent inverted as well. In this driver the extended NEC variant is used, which uses a 16 bit address bus for more devices to address. The advantage of the extended variant is that a normal 8 bit address can be represented by just inverting the lower byte.

Before the data is sent, a high burst (9ms) and a low signal (4.5ms) is transmitted. To indicate a beginning of a data packet. Then the address is sent (2x8 bit or 16 bit). Then the command is sent the same like the 8 bit address, with an inverted byte after the original command, to insure that the data is not corrupted. After that one end pulse is sent.

# Transmitter

## Design

The design is made with a Mealy controller and a specific data path. The Data formatter makes a packet to send directly, by concatenating the address with the data and the inverted data. This data is captured in a register after the controller received an available sign. Then the mod32 counter selects which bit is read from the register and that is modulated thru the modulator.



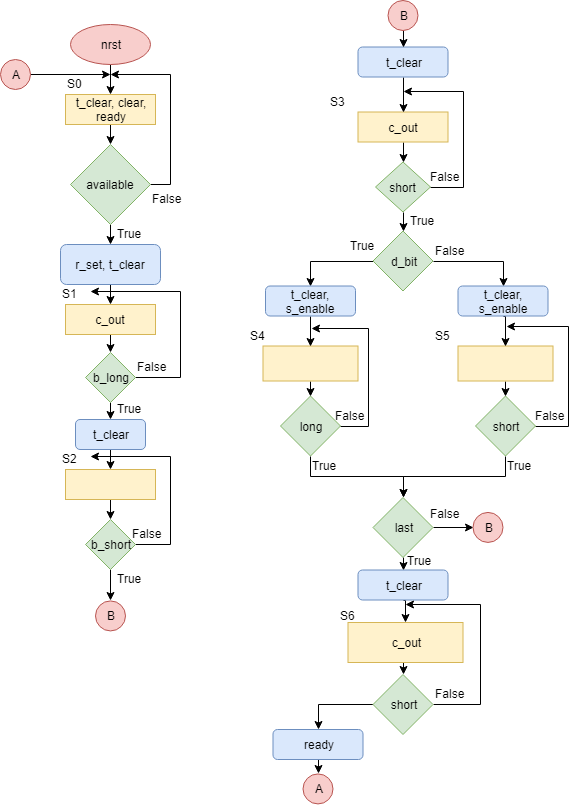
The characters t, s and r are used for:

T = Timer

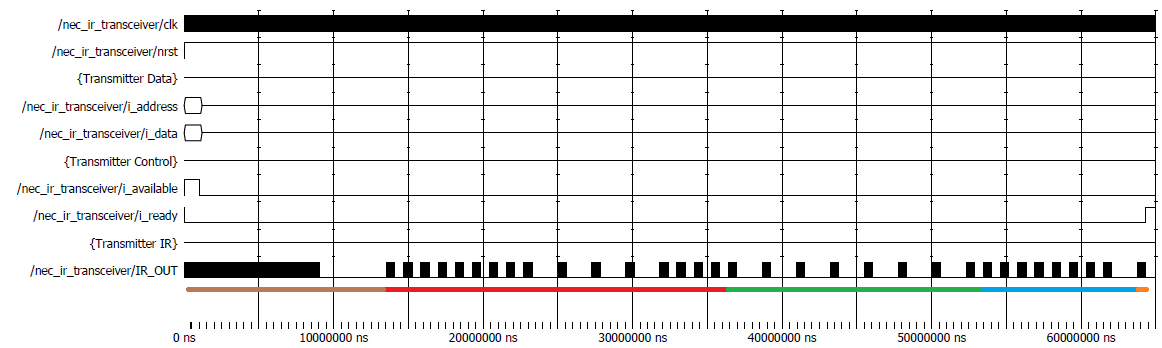
R = Register

S = Selection

According to this design, there is a matching ASM chart.



## Simulation



The brown part is the start burst pulse, the red part is the address (0x00F0), the green part is the command (0x01), the blue part is the inverse command (0xFE) and the orange part is the end bit.

# Receiver

The receiver needs a lot of safety to prevent infinite loops. So the design needs a watchdog timer and time-out safety in the controller. The period time of a clock pulse is the following.

Imagine using a 10 bit register that holds a value between 0 and 1023, then the maximum time after counting towards overflow is

This register is used for detecting the length of a pulse. We take the usual length of a pulse and add and subtract 10% of the time. This results in:

|  |  |  |
| --- | --- | --- |
| Original time | Max time | Min time |
| 9 ms | 9,9 ms | 8,1 ms |
| 4,5 ms | 4,95 ms | 4,05 ms |
| 1,69 ms | 1,859 ms | 1,521 ms |
| 563 us | 619 us | 507 ms |

If we convert these values to counter ticks this results in the following ticks:

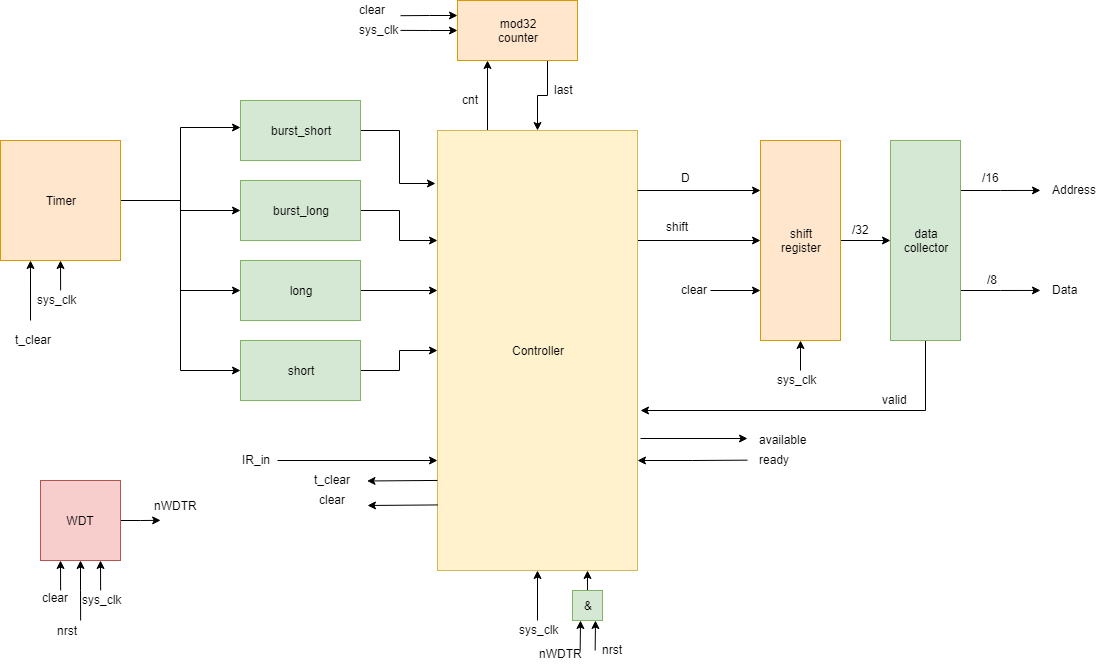
where is the time of the pulse, for example 9 ms.

|  |  |  |  |
| --- | --- | --- | --- |
| Original time | Original ticks | Max ticks | Min ticks |
| 9 ms | 878 | 966 | 790 |
| 4,5 ms | 439 | 483 | 395 |
| 1,69 ms | 165 | 182 | 148 |
| 563 us | 55 | 61 | 49 |

In the design we can take these measurements in the ranges of the timer to determine whether a pulse is valid to use or abort the whole message.

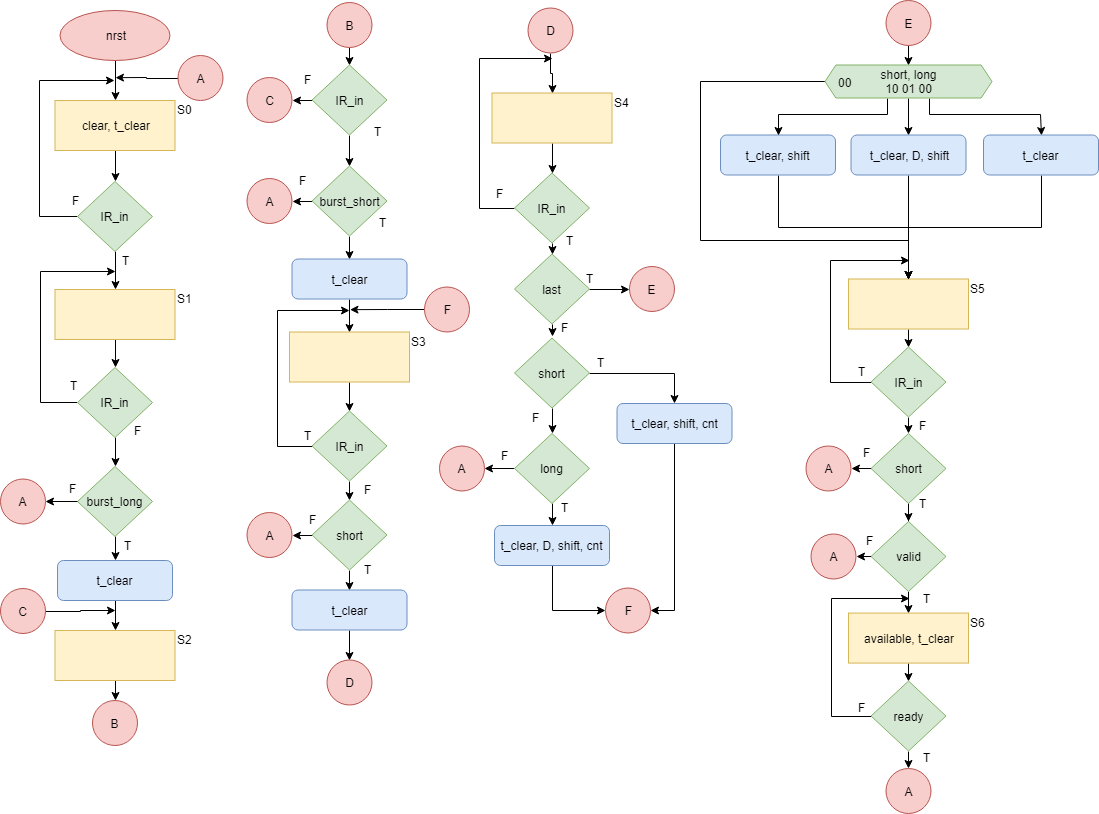
## Design

Based on the requirements above the following design is made.



The timer keeps track of the time where the green blocks indicate if the time is in between the boundaries of a specific signal. The mod32 counter is used for indicating the last bit is reached. The shift register shifts in the bits to form a total 32 bits value. After the shift register a data collector separates the address and data from the 32 bits value. If the data and not data are correct the valid flag is raised. If timing is not correct and it’s not detected by the controller itself, the watchdog timer resets the controller, to prevent from hanging infinitly.

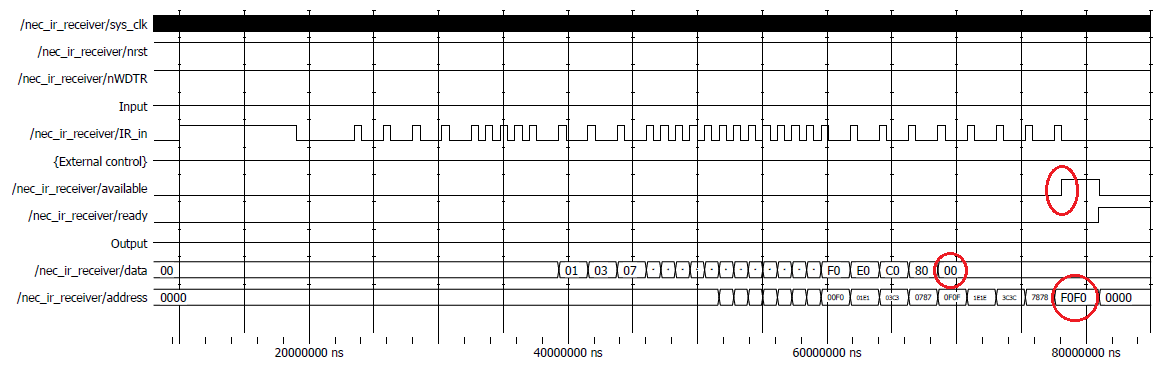
For the controller the following ASM is designed.



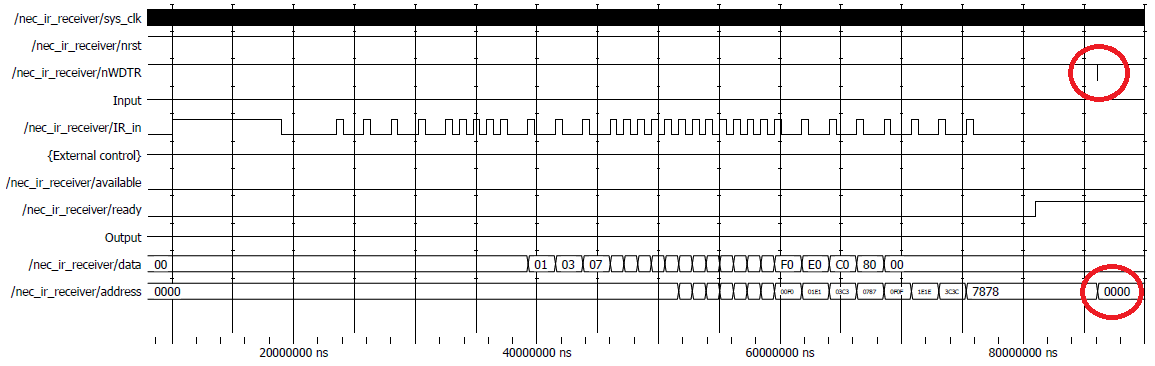
Every jump to label A is an exception based on timing. If a rising or falling edge occurs and the timing is not correct, it will jump to **S0** and restarts it’s procedure. If the exception happens during the message, it will fail continuously because the 9 ms signal is only at the beginning.

## Simulation

With a valid signal (Address is 0xF0F0 and the data is 0x00) The sequence succeeds with the right address and data and a data available flag high.



When The end bit is removed from the signal and thus there is no other edge to step into another state, the state machine is in an infinite loop, but as indicated in the red circles below, the watchdog timer resets the controller and after that the control clears all the registers and counters.



# Conclusion

The design behaves like the design requirements. The only thing that is not included is a repeat signal, in the transmitter nor the receiver. There is no option to accept a repeat signal in the receiver, because of a lack of bits, the watchdog timer shall reset the controller and the transmitter has no repeat option.

Because this device is build to send and receive direct commands, the repeat option is not included and not necessary.