NEC IR Transceiver



Content

[Introduction 3](#_Toc45223709)

[Transmitter 4](#_Toc45223710)

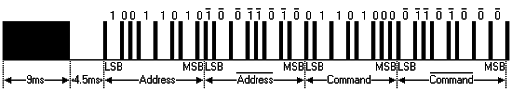
[Design 4](#_Toc45223711)

[Simulation 6](#_Toc45223712)

[Receiver 6](#_Toc45223713)

# 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.



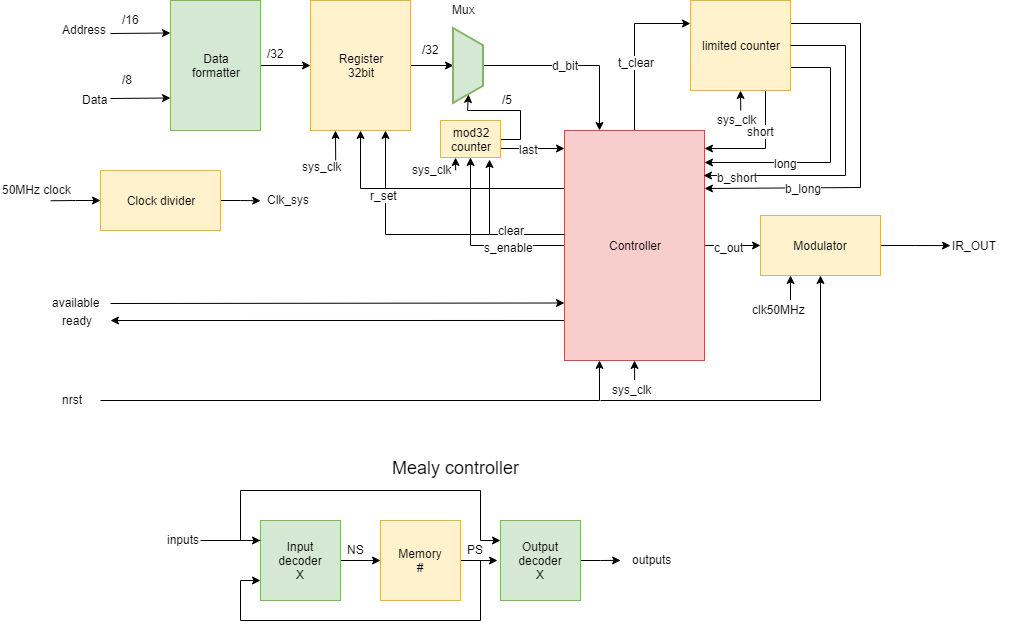
In the picture above there is one of the two NEC variants. This one uses an 8 bit address which is sent inverted as well. In this driver the extended NEC variant is used, which uses a 16 bit address bus for more devices to address. The advantage of the extended variant is that a normal 8 bit address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



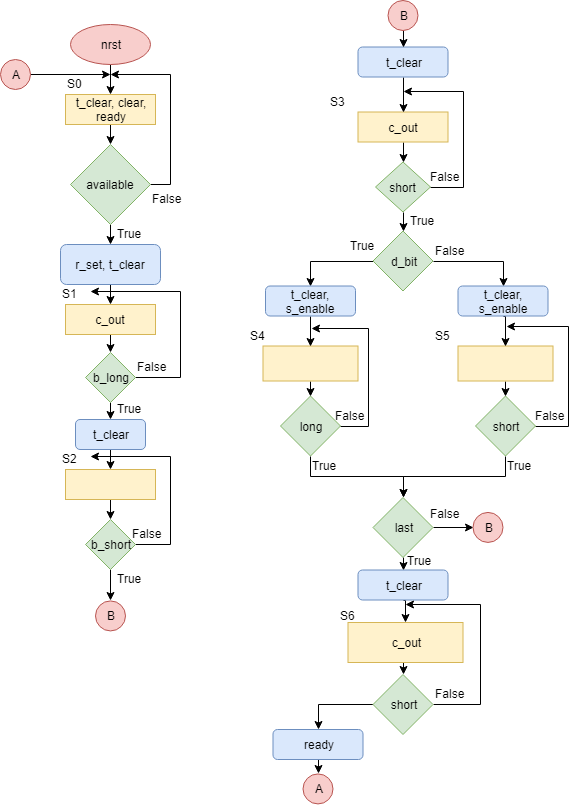
The characters t, s and r are used for:

T = Timer

R = Register

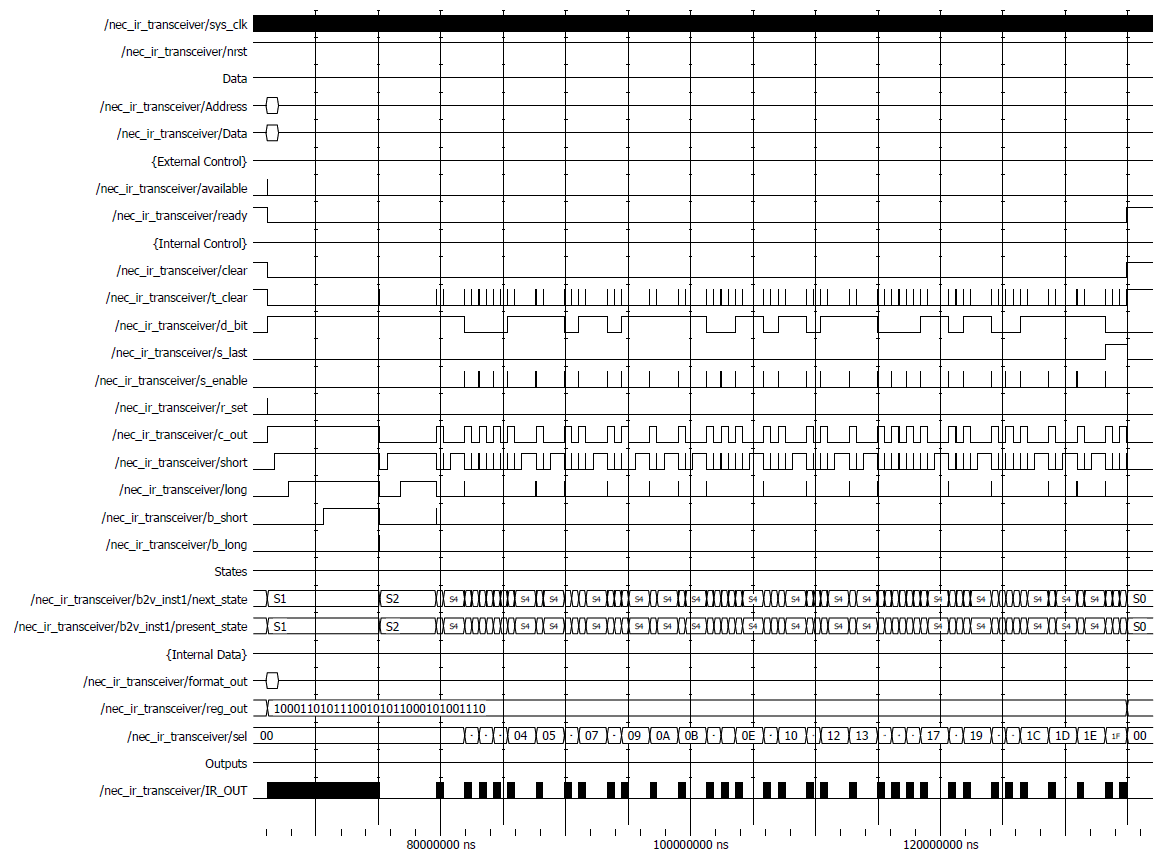
S = Selection

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



## Simulation

TEMPORARY SIMULATION



# 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.