# Multimodal Neopixel Peripheral Project Summary

Austin Rowland

Submitted
April 26, 2022

## Introduction

The purpose of this document is to describe the functionality of the multimodal Neopixel peripheral design (VHDL) and how users can interact with the peripheral based on their specific needs (assembly). The Neopixel peripheral can achieve seven different modes of functionality, based on what the user wants to accomplish, by using mode designation signals. The final product exceeded all expectations by meeting the required specifications from the end user and adding various additional functionality.

## Design Decisions and Implementation

The team chose to focus on creating a host of additional functionalities so that users can take advantage of the API and customize their Neopixel. The approach to the project and hardware was to implement a multiple-state machine, with each state representing different modes. Each mode consists of either storing or idle states, which determine the pixel's address, data, and other information. During the design process, the more widely applicable features were prioritized over some of the more specific features. Modern LED lights often come with solid gradient and color-changing features, so these features were prioritized over specific applications, such as the cyclone game from the original design.

## Device Functionality

### IO Address Constants and Mode Designation

For effective use, a user must understand the I/O ports and address assignments for setting pixel addresses and pixel data, as well as ports that select different modes for functionalities. In assembly, constant assignments should be made to access these mode I/O addresses properly.

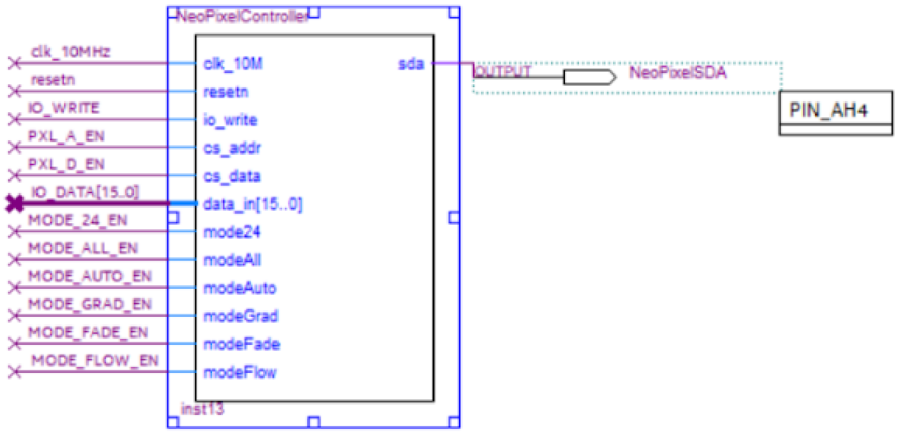

**Figure 1.** NeoPixelController Peripheral in Quartus Schematic showing all input and output ports.

| **I/O Address Assignment** | **I/O Address** |
|---|---|
| SWITCH_EN | 0x00 |
| LED_EN | 0x01 |
| TIMER_EN | 0x02 |
| HEX0_EN | 0x04 |
| HEX1_EN | 0x05 |
| PXL_A_EN | 0xB0 |
| PXL_D_EN | 0xB1 |
| MODE_24_EN | 0xB6 |
| MODE_ALL_EN | 0xB7 |
| MODE_AUTO_EN | 0xB8 |
| MODE_GRAD_EN | 0xB9 |
| MODE_FADE_EN | 0xBC |
| MODE_FLOW_EN | 0xBB |

**Figure 2.** Table with port listings in the IO Decoder with the respective I/O address.

## Neopixel Mode Functionality

An understanding of the different mode designations and their specific inputs (assembly) allow users to carry out different functionalities as described previously. A user can change the mode by sending an OUT of 1 to one of the several mode selections ports through chip select. The default mode sets a 16-bit color to an address using a single OUT to specify the address and a single OUT to specify the color. The 24-bit color mode takes three OUTs from the user and interprets them as RGB vectors, then sends that information to whichever pixel address is specified. The auto-increment feature is used by looping through multiple OUT statements to send color to the pixel data, and the address is incremented every time. The set all feature is used by incrementing through all addresses and sets the color to each pixel that was specified in the OUT statement. The gradient feature is used by using a single OUT statement and it sets all the pixels to a changing gradient. The fading feature is used by sending a single color to the peripheral; it will then set all the LEDs to that color and decrease and increase the brightness over time. The flow feature is used by setting the color vector and constantly incrementing/decrementing its RGB values to make a smooth, dynamic color-shifting gradient which updates with each clock cycle.

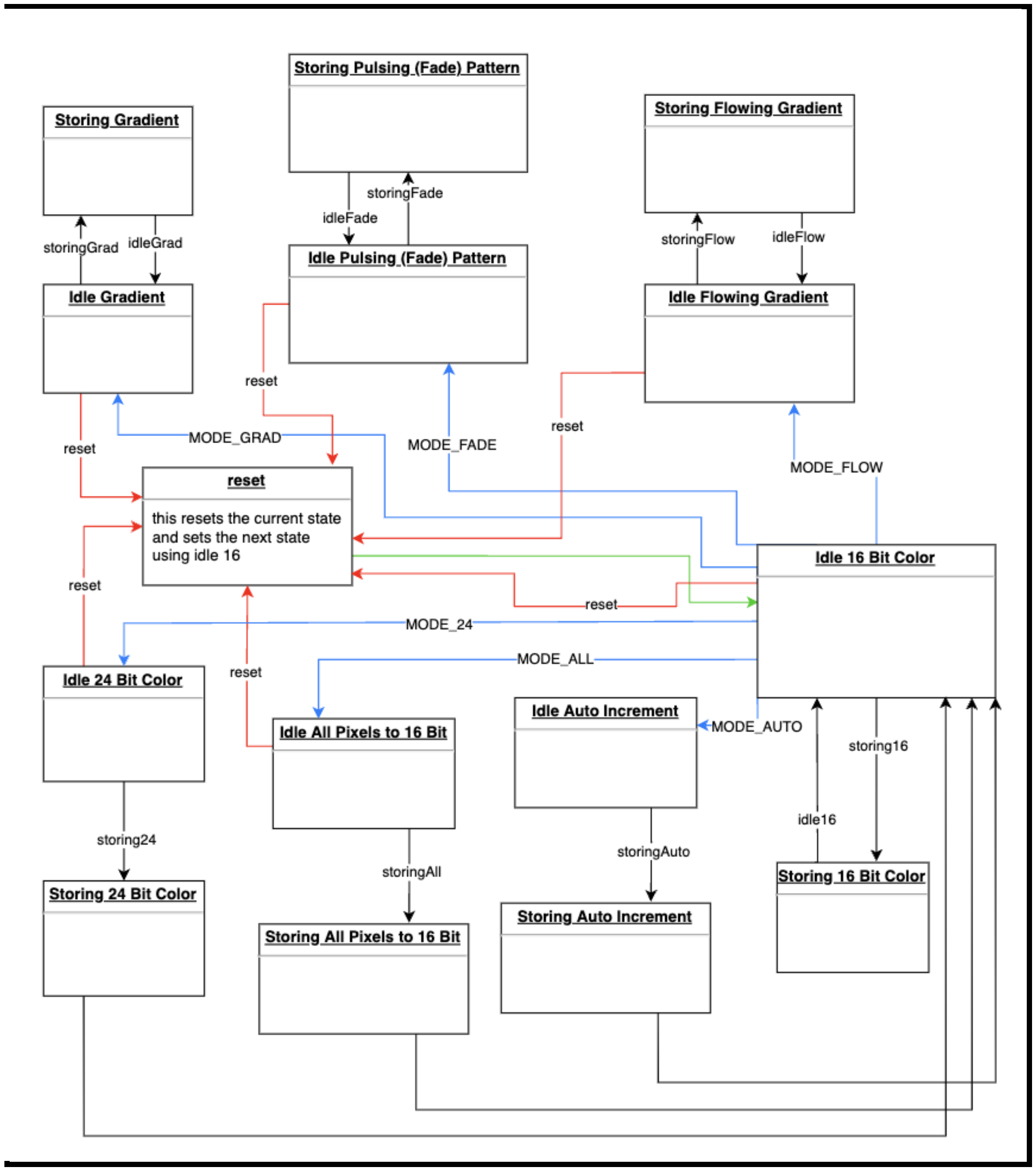

**Figure 3.** UML Diagram of the Neopixel Peripheral Modes

In the gradient mode, the RGB vector data is shifted between each address and RAM write buffer is updated accordingly to change the corresponding LED. For the flow mode, an RGB vector is specified in the peripheral and its RGB vector values and is either incremented or decremented based on the current color vector. The updated RGB values change all pixels' color values based on a clock cycle.

## Conclusion

The Neopixel peripheral provides a multi-use device that users can customize to fit their needs, and the implementation is all in VHDL so that the end user only needs to understand the mode designations and specific assembly inputs to use the peripheral. Recommendations to future users and engineers would be to understand the peripherals functionality and where it comes from. This will save any future teams a lot of time debugging and designing new features for the peripheral if that line in the sand is drawn early on.