Pixel Display Driver

Interface

Version 1.0

(11-21-2017)

Overview

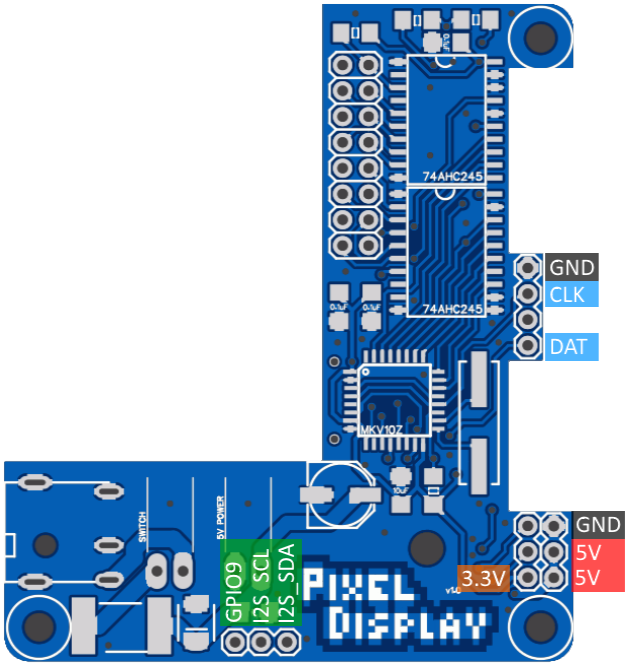
The Pixel Display Driver allows for the driving of a common 16x32 RGB LED panel through a simple SPI like interface (only CLOCK and DATA). This removes the burden of having to draw an image manually line by line, which can be very resource intensive. It also provides a real time level solution for boards that may not be able to provide the precise timing needed to run the display without flaws. The Pixel Display Driver was designed to have a Raspberry Pi plug directly into the top, but any board with SPI should also be able to drive the display as long as it also provides 3.3V.

Features

- 24bit colors
- Gamma correction (for more accurate colors)
- Fully buffered input data (to prevent tearing effect)
- Separate brightness control
- Frame rate up to 60 fps

Pinout

The driver board outputs 5V, but needs to be provided 3.3V. The bottom three pins in green are pass throughs of the inaccessible pins if a Raspberry Pi is connected. There are only two inputs which send data to the display.



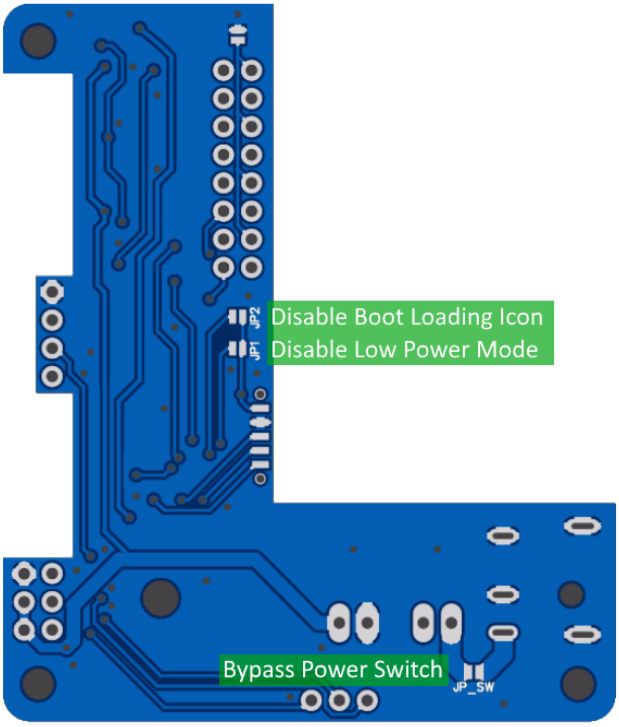
Jumper Configuration

On the bottom of the driver board, there are several jumpers that can be shorted to change functionality.

JP1 - Disables the low power mode which reduces brightness in order to keep the display power consumption under ~2A. (make sure to provide enough power)

JP2 - Disables the loading icon from appearing when the display is first powered on.

JP_SW - Bypasses the power switch so the driver is always on.



Data Transfer

Data is transferred to the display through a standard 2-wire SPI interface with a single data-in line and a clock line. All data is communicated by sending an entire data packet which includes the color data, metadata, and a sync byte. The end of a packet communication is signaled by a delay in the clock line.

Packet: [data (1536)] [meta (1)] [sync (1)]

data- [xxxx xxxx - xxxx xxxx]

Display data - *each pixel is 3 bytes (RGB) and fills each row from left to right and down through each row.*

metadata- [xxRS xxBB]

R - *(reset) set to 1 to reset to the loading icon*

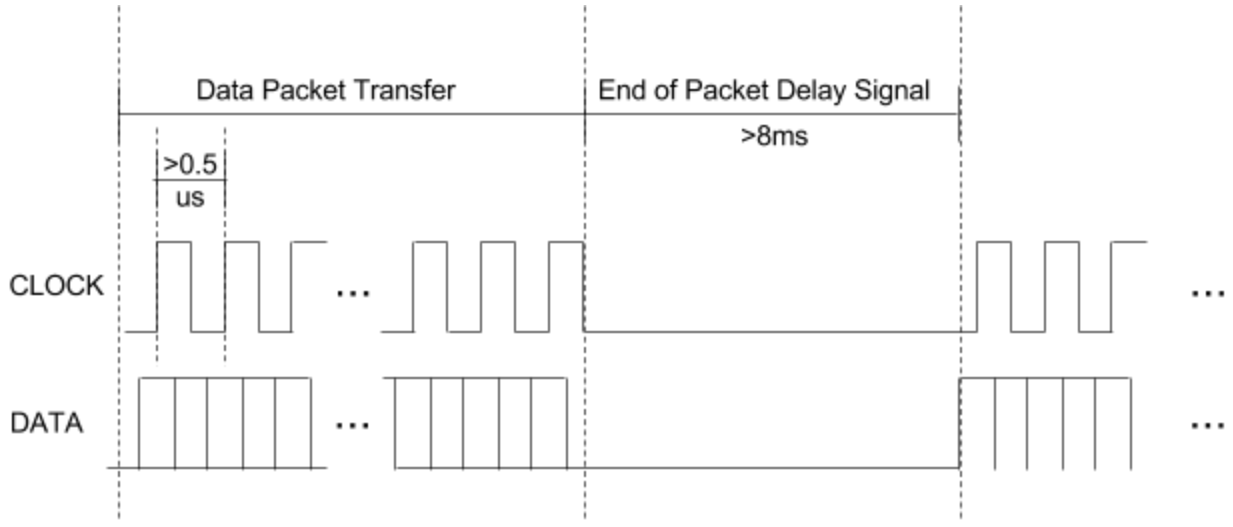
S - *(standby) set to 1 to initiate standby mode*

BB - *(brightness) set brightness value from 0-3*

sync- [0000 0001]

Sync byte to ensure correct data transfer

Diagram:



Timing Requirements

Parameter	Value	Description
Max Clock Frequency	1.8MHz	The maximum clock frequency that can be used when sending a packet
Min Packet End Delay	8ms	The minimum amount of time required to signal the end of a packet communication
Time to Timeout	~2s	If no packets are sent within this time, the display enters timeout mode
Max Frame Rate	60fps	The maximum frames that can be transferred to the display per second

Display Modes

The Pixel Display can be in different modes depending on meta data sent or current state of data transfer.

Reset - *Set on power up or in packet metadata*
(reset) -> [Loading Icon]

Standby - *Set in packet metadata and useful as power down routine*
(standby) -> [Loading Icon] -> (wait 2s) -> [Loading Icon] -> (clk line low) -> [Loading Icon] -> (wait 2s) -> [Blank]

Timeout - *Set when no data is transferred to the driver after ~2s*
(timeout) -> [Blank]

Data - *Set when data packet is successfully transferred with no metadata flags set*
(data) -> [Display Data]