==========
WS2803Due v0.1 Beta README
WS2803Due:

Written by:
James Martin
February 24, 2013
vwgogreen@gmail.com
==========

Contents:

1. Introduction

WS2803Due is an open source library intended for use with the Arduino Due hardware platform and WS2803 LED driver chips. The intent of this library is to provide fast and efficient switching of RGB LED pixels and can be used with standard single filament LED's as well. The library utilizes the SPI port to transmit data to the LED drivers, and no soft SPI option is available. The user is able to define the clock divider that specifies the speed at which the data is transmitted. Play around with the clock divider setting to find the maximum speed that the chips can handle with your set-up. The data sheet says that the chips can handle clock frequencies up to 25MHz. This is all dependent on the physical set-up and connections of your LED drivers. In general, the further apart and from the SPI port the slower the transmission rates need to be. Another feature is the ability to tell the library to handle the latching of data, or allow the user to handle the latching of the data outside of the library. Handling the latching outside of the library allows the program to perform other calculations instead of having a delay where nothing is being accomplished (for more info on external latching please see the links section).

Version 0.1 covers:

- 3 Constructors
- Getter and setter for user defined SPI clock divider
- Getter and setter for user defined number of WS2803 chips
- Getter and setter for user defined number of RGB pixels (e.g. SMD5050 RGB)
- Getter and setter for user defined number of LED's
- Getter and setter for user defined external timing
- Set data values by pixel with RGB data or by individual LED and brightness
- Simple show method transmits all the data at once

2. License

3. System requirements

Arduino Due or comparable hardware that uses the SAM3x architecture.
Arduino IDE 1.5.1r2 or later, or comparable IDE is required.

+------------------+
| 4. Installation |
+------------------+

Place the “WS2803Due” in the Arduino IDE “libraries folder, and then add it to the list of available libraries in the IDE. Then select it from Sketch > Import Library. If you need assistance please follow the guide posted by Arduino.
http://www.arduino.cc/en/Hacking/Libraries

+------------------------+
| 5. Implementation |
+------------------------+

Step 1.
In order to use this library you must first install the library so that your IDE knows where it is located, and then include it at the top of your sketch by using:

#include <WS2803Due.h>

That is all that is needed to access the constructors and methods in the library. Start by creating an object of type WS2803Due. The constructor to use depends on your set-up.

Step 2.
If the required information is not known at setup time use the default constructor:
WS2803Due <Object Name> = WS2803();

If all available ports on the WS2803(s) are full use:
WS2803Due <Object Name> = WS2803Due(uint16_t <Number of WS2803 Chips>, uint8_t <Clock Divider>, bool <External Timing>)

If not all available ports on the WS2803(s) are being used then use:
WS2803Due <Object Name> = WS2803Due(uint16_t <Number of WS2803 Chips>, uint16_t <Number of RGB Pixels>, uint8_t <Clock Divider>, bool <External Timing>)

Step 3.
Once the object is created and all required environment variables are set call 'begin' which will allocate the necessary memory for storing the color data, and create and start the SPI with the environment variables that were declared:
<Object Name>.begin()

Step 4.
Send data to chips by using one of two methods. The LED's and pixels are 1 indexed because they are physical objects. For example, the first LED in the row is 1 and not 0 similarly the first pixel in the row is 1 and not 0. An RGB pixel has one red, one green, and one blue LED filament inside. So essentially there are 3 LED's in each pixel.

The first method is used to assign brightness of each LED filament individually.
<Object Name>.setLEDColor(uint16_t <Led Number>, uint8_t <Brightness>)

The second method is used to assign color values to common RGB LEDs.
<Object Name>.setPixelColor(uint16_t pixelNum, uint8_t red, uint8_t green, uint8_t blue)

Step 5.
Call the show method which will transmit the data to the LED driver chips. If external latching is being used then the library will not latch the data. If external latching is not being used, then the latching will be dealt with outside of the library.

<Object Name>.show()

+----------------------------+
| 6. Method Definitions |
+----------------------------+

1. WS2803Due(uint16_t <numChips>, uint8_t <clock>, bool <latching>)
Constructor to be used when every port on the WS2803 chips are being used.

Where:
<numChips> is the number of WS2803 chips that are being used
<clock> is the value from 1-255 of the clock divider for SPI
<latching> is a boolean value that defines whether the library will handle the latching of data, or the user will handle latching of data. TRUE for external latching, and FALSE for internal latching.

2. WS2803Due(uint16_t <numChips>, uint16_t <numPixels>, uint8_t <clock>, bool <latching>)
Constructor to be used when not every port on the WS2803 chips are being used.

Where:
<numChips> is the number of WS2803 chips that are being used
<numPixels> is the number of RBG LED's being used, if not using RGB LED's. If not using RGB LED's divide the total number of LED's by 3 for this value.
<clock> is the value from 1-255 of the clock divider for SPI
<latching> is a boolean value that defines whether the library will handle the latching of data, or the user will handle latching of data. TRUE for external latching, and FALSE for internal latching.

3. begin(void)
To be used to intialize the SPI variables and instance, and allocate the proper amount of memory for storing the LED data to be used for displaying. This method has to be called after instantiation, and prior to storing and displaying.
Return: void

4. setClockDiv(uint8_t <clock>)
Sets the clock divider for the SPI. Start higher and work your way down to get the fastest results. See the links section for more information regarding clock divisions and SPI in general.
Return: void

Where:
<clock> is a value from 1 - 255

5. setExtLatching (bool <latching>)
Sets whether external latching will be used, or if the library will handle the latching of the data. In short, if the library handles the latching of the data, the user is not able to do anything while the data is being latched. If the user wants utilize something similar to protothreads or similar libraries to maximize processing time (See link regarding timing). The WS2803 chips require a low clock signal of at least 400 microseconds (e.g. delayMicroseconds(400);) in order to latch the data.
Return: void

Where:
<latching> is TRUE for external or FALSE for internal.

6. setNumChips(uint16_t <numChips>)

Sets the number of WS2803 chips that are being used. This is most commonly used if the default constructor was instantiated and the number of chips were not known at that time.
Return: void

Where:
<numChips> is the total number of WS2803 chips being used.

7. setNumLEDs(uint16_t <numLEDs>
Sets the number of total LEDs being used. This should be a multiple of 3 because this library is intended for RGB LED's.
Return: void

8. setNumPixels(uint16_t <numPixels>)
Sets the number RGB LED pixels being used. If single LED's are being used, take the total number of LED's and divide that by 3 in order to set the number being used.
Return: void

Where:
<numPixels> is the number of RGB LED's being used.

9. getClockDiv(void)
Gets the current clock divider being used for the SPI transmission.
Return: uint8_t

10. getExtLatching (void)
Gets the state of the external timing.
Return: boolean

11. getNumChips(void)
Gets the number of WS2803 chips being used.
Return: uint16_t

12. getNumLEDs(void)
Gets the number of total LED's being used.
Return: uint16_t

13. getNumPixels(void)
Gets the total number of RGB pixels being used.
Return: uint16_t

+-----------+
| 7. Links |
+-----------+

Arduino Due:
http://arduino.cc/en/Main/ArduinoBoardDue

SPI:
http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
http://arduino.cc/en/Reference/SPI

WS2803 Website
http://www.world-semi.com/en/Driver/Lighting_LED_driver_chip/WS2803/

WS2803 Data Sheet
http://www.world-semi.com/uploads/soft/120502/1-120502110159.rar

Introduction to timing and protothreading
http://playground.arduino.cc//Main/TutorialList#Timing

Protothreading library for Arduino
http://code.google.com/p/arduinode/downloads/detail?name=pt.zip&can=2&q=