Raging Bits Addressable LEDs UART driver v1.0

Top level specs

Up to 900 bytes buffer.
Auto UART speed detection.
500Kbaud UART with 3.3v signal protection.
Independency of LED type. (Compatible with most of addressable LEDs brands.)
5V paththrough up to 5A.
Minimum of 20Hz (900bytes & 500kbaud)

Device interface

Input Power - GND and 5V Supply.
UART PORT. Organized as GND, RX, TX and device RESET access.
RESET is active LOW.

Power

Power consumption at 5V (without LEDs liten up) is aproximatly 15mA.

Commands

The system is designed to have the commands syncronized by length of input data.
This way there will always be a syncronization reply by the driver to any input.

Initialisation

The driver will take 2 steps before being ready for work.
First the device needs to detect the work speed being used by the user.
Second, it will need to be set with the size of data correspondent to the LED array to be controlled.
Lastely, and ready for work, will take arrays of data (of the length set by the last step) and at the end of each, the data is sent to the LED array.
Every time the driver is reset/power cycled, the initialisation needs to be performed.

UART speed auto detection

The UART speed detection is done upon receiving the character 'X' (0x58).
The user shall send the character 'X' continuously at the chosen speed until the device starts replying with the same 'X' back.
The available (in baud) speeds are:

4800	9600	14400	16200
28800	38400	57600	76800
115200	250000	500000	

Ex.:
Driver input:
> 0x58 0x58 0x58 0x58 0x58 0x58 ... 0x58 0x58 0x58 0x58 0x58 0x58 0x58
Driver output:
< 0x58 0x58 0x58 0x58 0x58 0x58 0x58
| Point at which the driver detected the user speed.

Data frame length

The data frame length is set right after the device speed detection is successful and before starting to relay the data o the LEDs.

The length can have a value from 1 to 900 and will alwais be sent as a 16bit value, MSB to LSB, headed by the value 'L' (0x4C) as indicative that the data length is being setby the user.

This process consists then in the respective format.

'L' <MSB length> <LSB length>

If the inserted value is valid, the driver will replay with a 'Y' (0x59) to indicate the the data length has been successfully set.

If the inserted value is invalid, the driver will reply with a 'N' (0x4E) to indicate that the value given by the user is invalid.

Until the driver is set with a valid length, it will be looped in the current step.

Ex:
Example case, the user in will have 24 LEDs to be driven.
Of the 24, the first 20 are RGB with a set of 3 bytes each and the last 4 are RGBW with a set of 4 bytes each, resulting in 76 bytes of total data. There for the 16bit representation of 76 decimal is 0x004C hexadecimal. For the case, there has been a mistake on the input, such that the first length value byte has suffered an interference such the value sent was 0xFA4C instead of 0x004C. After the first attempt the value is then set correctly.

Driver input (with wrong value)
> 0x4C 0xFA 0x4C
Driver output
< 0x4E

Driver input (with correct value)
> 0x4C 0x00 0x4C
Driver output
< 0x59

Data frame driving

As a driver, the device will simply convert frames of data, with the set length, into the addressable LEDs format, in a continuous loop.

Loop

As soon as the driver is ready for new data, the character 'R' (0x52) is sent.

Upon receiving 'R', the user shall send a lump of data with the configured size.

After the last byte is sent, the driver will acknowledge the data input with the character 'Y' (0x59).

At this point the driver sends the lump of data to the LEDs.

Any data sent by the user until the device is again ready, will be ignored.

Frame errors

This scheme allows a constant coodination between user and driver.
If frame bytes missing case needs to be contemplated, then the user will not have the chance to correct the frame but will have no delay to insert the next one.
At this point, after sending the full frame the user shall keep sending padding data such then missing bytes get filled with padding until the totallity of the data sent is received by the driver.
After a 'Y' character is received, the user shall stop sending any more padding.
This is a rare case, but not an impossible one.

Ex.:
Example case of a 2 LED string with 3bytes of data per LED. For this example, for the first frame one of the bytes sent is missing at the driver input and will have a fill with padding.
The second frame has a correct reception.

Driver output
> 0x52
Driver input
> 0x24 0x10 0x10 0x24 0x10 0x10 0x00 0x00 0x00
Driver output
> 0x59
| Point at which the driver received a full 6 bytes length and all other data is ignored. 1 byte of padding used.
Driver output
> 0x52
Driver input
> 0x24 0x10 0x10 0x24 0x10 0x10 0x00 0x00
Driver output
> 0x59
| Point at which the driver received a full 6 bytes length and all other data is ignored. No padding used.
...

Demo:

This demo has been done with 3 LEDs,
1x SK6812 – GRBWW (Green, Red, Blue, Warm White)
2x WS2812 – GRB (Green, Red, Blue)

The demo consists on:

1) Starting the driver
2) Setting the data length to 10, 1 x 4 bytes for GRBWW + 2 x 3 bytes for GRB.
3) Send the chunks of data for:

a) LED1 = WW LED2 = B LED3 = R
b) LED1 = B LED2 = G LED3 = R
c) LED1 = W LED2 = W LED3 = W
d) LED1 = W+WW LED2 = W LED3 = W
e) LED1 = WW LED2 = OFF LED3 = W

W is white by setting RGB set with same values.
WW is Warm White single die only. Only 1x SK6812 has warm white led die.
B is blue.
R is red.
G is green.

```
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58
58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58
58 58 58 58 58 58 58 58

4C 00 0A 59 52

00 00 00 0A 00 00 0A 00 0A 00 59 52

00 00 0A 00 0A 00 00 00 0A 00 59 52

0A 0A 0A 00 0A 0A 0A 0A 0A 0A 59 52

0A 0A 0A 0A 0A 0A 0A 0A 0A 0A 59 52

00 00 00 0A 00 00 00 0A 0A 0A 59 52
```

Illustration 1: Demo commands

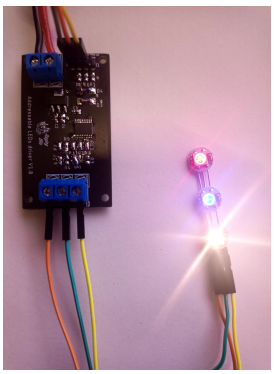

Illustration 2: a) LED1 = WW LED2 – R LED3 – R

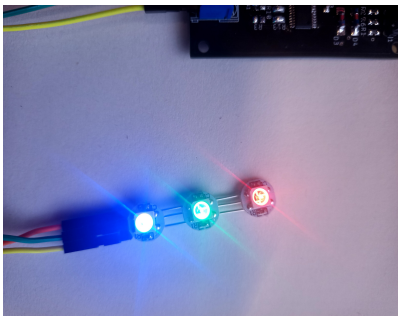

Illustration 3: b) LED1 = B LED2 – G LED3 – R

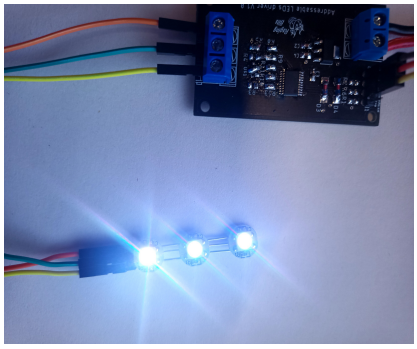

Illustration 4: c) LED1 = W LED2 – W LED3 – W

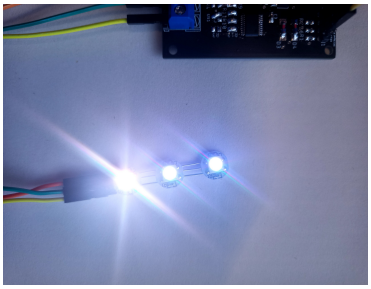

Illustration 5: d) LED1 = W+WW LED2 = W LED3 – W

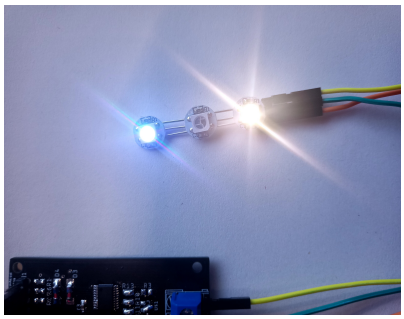

Illustration 6: e) LED1 = WW LED2 – OFF LED3 – W

Notes

The UART interface is done with 3.3v signals.
The TX pin will only be able to output 3.3v. The RX pin is capable of 5v input signals.

The refresh rate depends on a combination of 3 factors:
Frame length.
Uart speed.
User reaction time.

As a reference:
~20Hz refresh rate with Frame Size of 900bytes, 500Kbaud and meaningless user reaction time.
~20Hz refresh rate with Frame Size of 450bytes, 250Kbaud and meaningless user reaction time.
Etc.

Most common and tested LEDs

WS2812
WS2811
SK6812
Etc