```
// ----------------------------------------- //
// This file is autogenerated by pioasm; do not edit! //
// ----------------------------------------- //

#pragma once
```

If build is not targetting the RP2040 device then include the header file "hardware/pio.h"

```
#if !PICO_NO_HARDWARE
#include "hardware/pio.h"
#endif

// ------ //
// ws2812 //
// ------ //
```

Defining time 1, 2, 3 and other variables

```
#define ws2812_wrap_target 0
#define ws2812_wrap 3

#define ws2812_T1 2
#define ws2812_T2 5
#define ws2812_T3 3
```

Programming the instruction set

```
static const uint16_t ws2812_program_instructions[] = {
        //     .wrap_target
    0x6221, //  0: out    x, 1         side 0 [2]
    0x1123, //  1: jmp    !x, 3        side 1 [1]
    0x1400, //  2: jmp    0            side 1 [4]
    0xa442, //  3: nop                 side 0 [4]
        //     .wrap
};
```

```
#if !PICO_NO_HARDWARE

static const struct pio_program ws2812_program = {

    .instructions = ws2812_program_instructions,

    .length = 4,

    .origin = -1,

};


static inline pio_sm_config ws2812_program_get_default_config(uint offset) {

    pio_sm_config c = pio_get_default_sm_config();
    sm_config_set_wrap(&c, offset + ws2812_wrap_target, offset + ws2812_wrap);
    sm_config_set_sideset(&c, 1, false, false);
    return c;

}


#include "hardware/clocks.h"

static inline void ws2812_program_init(PIO pio, uint sm, uint offset, uint pin, float freq, bool rgbw) {

    pio_gpio_init(pio, pin);
    pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true);
    pio_sm_config c = ws2812_program_get_default_config(offset);
    sm_config_set_sideset_pins(&c, pin);
    sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24);
    sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
    int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3;
    float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
    sm_config_set_clkdiv(&c, div);
    pio_sm_init(pio, sm, offset, &c);
    pio_sm_set_enabled(pio, sm, true);
}
```

Here we will get default configuration in c of the state machine

setting wrap address in sm (memomory address to wrap, if instruction not update program counter then instructure memory address which set program counter to wrap target)

It sets the sideset option in the configuration

we are configuring the GPIO that is being used by the PIO

At PIO, set the pin direction to the O/p

using the generated function we will get default configuration

In the state machine configuration set pin

It tells if we have RGBW or the RGB led. True if we output & False if we shift to right

Here we will set up the FIFO join in configuration of sm.

1 bit no. of cycles to output

To achieve correct bit rate slow down the sm's execution time

Set divider of the sm clock

Jump at the offset of the start address and Load the configuration

Enable the state machine running that is set at

```
#endif

// ---------- //
// ws2812_parallel //
// ---------- //
```

Defining variables

```
#define ws2812_parallel_wrap_target 0
#define ws2812_parallel_wrap 3
```

Defining time variables T_1, T_2 and T_3

```
#define ws2812_parallel_T1 2
#define ws2812_parallel_T2 5
#define ws2812_parallel_T3 3
```

This is the program instruction set

```
static const uint16_t ws2812_parallel_program_instructions[] = {
        //     .wrap_target
    0x6020, //  0: out    x, 32
    0xa10b, //  1: mov    pins, !null       [1]
    0xa401, //  2: mov    pins, x           [4]
    0xa103, //  3: mov    pins, null        [1]
        //     .wrap
};

#if !PICO_NO_HARDWARE
static const struct pio_program ws2812_parallel_program = {
    .instructions = ws2812_parallel_program_instructions,
    .length = 4,
    .origin = -1,
};
```

```
static inline pio_sm_config ws2812_parallel_program_get_default_config(uint offset) {

    pio_sm_config c = pio_get_default_sm_config();
    sm_config_set_wrap(&c, offset + ws2812_parallel_wrap_target, offset + ws2812_parallel_wrap);
    return c;

}



#include "hardware/clocks.h"

static inline void ws2812_parallel_program_init(PIO pio, uint sm, uint offset, uint pin_base, uint
pin_count, float freq) {
    for(uint i=pin_base; i<pin_base+pin_count; i++) {

        pio_gpio_init(pio, i);

    }
    pio_sm_set_consecutive_pindirs(pio, sm, pin_base, pin_count, true);
    pio_sm_config c = ws2812_parallel_program_get_default_config(offset);
    sm_config_set_out_shift(&c, true, true, 32);
    sm_config_set_out_pins(&c, pin_base, pin_count);
    sm_config_set_set_pins(&c, pin_base, pin_count);
    sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
    int cycles_per_bit = ws2812_parallel_T1 + ws2812_parallel_T2 + ws2812_parallel_T3;
    float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
    sm_config_set_clkdiv(&c, div);
    pio_sm_init(pio, sm, offset, &c);
    pio_sm_set_enabled(pio, sm, true);

}


#endif
```