```
#pragma once

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

// ------ //
// ws2812 //
// ------ //

#define ws2812_wrap_target 0
#define ws2812_wrap 3

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

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) { 9
    pio_sm_config c = pio_get_default_sm_config(); 10
    sm_config_set_wrap(&c, offset + ws2812_wrap_target, offset + ws2812_wrap); 11
    sm_config_set_sideset(&c, 1, false, false); 12
    return c;
}

#include "hardware/clocks.h"
static inline void ws2812_program_init(PIO pio, uint sm, uint offset, uint pin, float freq, bool
rgbw) { 6
    pio_gpio_init(pio, pin); 7
```

批注 [顾1]: help the user to instantiate an instance of the LED driver program

批注 [顾2]: which of RP2040's two PIO instances we are dealing with

批注 [顾3]: which state machine on that PIO we want to configure to run the WS2812 program

批注 [顾4]: where the PIO program was loaded in PIO's 5-bit program address space

批注 [顾5]: which GPIO pin our WS2812 LED chain is connected to

批注 [顾6]: the frequency (or rather baud rate) we want to output data at

批注 [顾7]: true if we are using 4-colour LEDs (red, green, blue, white)

批注 [顾8]: configure a GPIO for use by PIO

```
    pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true); 8
    pio_sm_config c = ws2812_program_get_default_config(offset); 9
    sm_config_set_sideset_pins(&c, pin); 13
    sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24); 14
    sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX); 15
    int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3; 16
    float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit); 17
    sm_config_set_clkdiv(&c, div); 18
    pio_sm_init(pio, sm, offset, &c); 19
    pio_sm_set_enabled(pio, sm, true); 20
}

#endif

// --------------- //
// ws2812_parallel //
// --------------- //

#define ws2812_parallel_wrap_target 0
#define ws2812_parallel_wrap 3

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

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 +
```

批注 [顾9]: Sets the PIO pin direction of 1 pin starting at pin number pin to out

批注 [顾10]: Get the default configuration using the generated function for this program

批注 [顾11]: Sets the side-set to write to pins starting at pin pin

批注 [顾12]: False for shift_to_right .
True for autopull.

批注 [顾13]: This is the total number of execution cycles to output a single bit.

批注 [顾14]: Slow the state machine's execution down, based on the system clock speed and the number of execution cycles required per WS2812 data bit, so that we achieve the correct bit rate.

批注 [顾15]: Load the configuration into the state machine, and go to the start address (offset)

批注 [顾16]: make it go

```
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
```