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

#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) {
    pio_sm_config c = pio_get_default_sm_config(); (9) default state machine configuration
    sm_config_set_wrap(&c, offset + ws2812_wrap_target, offset + ws2812_wrap); (10) set up wrap
    sm_config_set_sideset(&c, 1, false, false); (11) set up sideset option
    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); (7) set the gpio mode of the given pin as pio
    pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true); (8) set the pin's direction as output
    pio_sm_config c = ws2812_program_get_default_config(offset); (12) obtain the configuration of state machine
    sm_config_set_sideset_pins(&c, pin); (13) set sideset to write to pins starting at "pin"
    sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24);(14) set up out shifting parameters { shift osr to the left, autopull enabled, 32/24 threshold.
    sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX); (15) set FIFO_JOIN_STATE as Transmission CTX)
    int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3; (16) total cycles of execution to output a bit.
    float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
    sm_config_set_clkdiv(&c, div); } (17) slow down the state machine to match the freq
    pio_sm_init(pio, sm, offset, &c); (18) load the configuration into the state machine and go to the program starting address
    pio_sm_set_enabled(pio, sm, true); (19) make the state machine enabled.
}

#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
```

```
73  static const uint16_t ws2812_parallel_program_instructions[] = {
74          //     .wrap_target
75      0x6020, //  0: out    x, 32
76      0xa10b, //  1: mov    pins, !null           [1]
77      0xa401, //  2: mov    pins, x               [4]
78      0xa103, //  3: mov    pins, null            [1]
79          //     .wrap
80  };
81
82  #if !PICO_NO_HARDWARE
83  static const struct pio_program ws2812_parallel_program = {
84      .instructions = ws2812_parallel_program_instructions,
85      .length = 4,
86      .origin = -1,
87  };
88
89  static inline pio_sm_config ws2812_parallel_program_get_default_config(uint offset) {
90      pio_sm_config c = pio_get_default_sm_config();
91      sm_config_set_wrap(&c, offset + ws2812_parallel_wrap_target, offset + ws2812_parallel_wrap);
92      return c;
93  }
94
95  #include "hardware/clocks.h"
96  static inline void ws2812_parallel_program_init(PIO pio, uint sm, uint offset, uint pin_base, uint pin_count, float freq) {
97      for(uint i=pin_base; i<pin_base+pin_count; i++) {
98          pio_gpio_init(pio, i);
99      }
100     pio_sm_set_consecutive_pindirs(pio, sm, pin_base, pin_count, true);
101     pio_sm_config c = ws2812_parallel_program_get_default_config(offset);
102     sm_config_set_out_shift(&c, true, true, 32);
103     sm_config_set_out_pins(&c, pin_base, pin_count);
104     sm_config_set_set_pins(&c, pin_base, pin_count);
105     sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
106     int cycles_per_bit = ws2812_parallel_T1 + ws2812_parallel_T2 + ws2812_parallel_T3;
107     float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
108     sm_config_set_clkdiv(&c, div);
109     pio_sm_init(pio, sm, offset, &c);
110     pio_sm_set_enabled(pio, sm, true);
111 }
112
113 #endif
114
115
```