```
// -------------------------------------------------- //
// 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
```

} define constants

```
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
};
```

} instructions to ws2812 that or determine delay cycles

Structure defined for ws2812 program

```
#if !PICO_NO_HARDWARE
static const struct pio_program ws2812_program = {
    .instructions = ws2812_program_instructions,
    .length = 4,
    .origin = -1,
};
```

– set definitions for pio state machine to work with ws281

```
static inline pio_sm_config ws2812_program_get_default_config(uint offset) {
 (10) pio_sm_config c = pio_get_default_sm_config();
 (11) sm_config_set_wrap(&c, offset + ws2812_wrap_target, offset + ws2812_wrap);
 (12) sm_config_set_sideset(&c, 1, false, false);
 (13) return c;
}
```

program initializes ws2812 &

```
#include "hardware/clocks.h"
static inline void ws2812_program_init(PIO pio, uint sm, uint offset, uint pin,
float freq, bool rgbw) {
 (7) pio_gpio_init(pio, pin);                                    configure GPIO
 (8) pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true);      set 1 "pin" to out.
 (9) pio_sm_config c = ws2812_program_get_default_config(offset);   set sm configuration
(14) sm_config_set_sideset_pins(&c, pin);  set side-set to ride to pins
(15) sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24);  set OSR refill params.
(16) sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);  manipulate FIFOS to get more throughput in DMA
(17) int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3;  # of cycles to output 1 bit
(18)  float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);  ~~set~~ slow down SM to get right bit rate
(19) sm_config_set_clkdiv(&c, div);  set clock divider
(20)  pio_sm_init(pio, sm, offset, &c);  load configuration into SM
(21) pio_sm_set_enabled(pio, sm, true);  make it go
}

#endif

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

Same program for multiple pins

```
#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 +
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
```