

```
// ----- //
// 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
};
```

Define uint16\_t variable of  
ws2812 program  
instructions.

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

Define struct of ws2812 program

for default (introduction about the  
autogeneration of pioasm.

```
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;
}
```

Set wrap.

Set sideset.

```
#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);
```

Initialize GPIO for PIO usage (the pins for LED output signals)

from ws2812-  
program-init

⑨

from the main  
function in .c

⑥

⑦

```

8 pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true); set pin direction -
9 pio_sm_config c = ws2812_program_get_default_config(offset); get default configuration
10 sm_config_set_sideset_pins(&c, pin); set sideset to write to pins (in sm)
11 sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24);
12 sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX); manipulate the FIFOs joining in a sm config.
13 int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3; calculate cycles per bit.
14 float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit); get sm clock divider
15 sm_config_set_clkdiv(&c, div); set rate. in sm config.
16 pio_sm_init(pio, sm, offset, &c); initialize the state machine.
17 pio_sm_set_enabled(pio, sm, true); enable the state machine.
}

#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 +
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
```