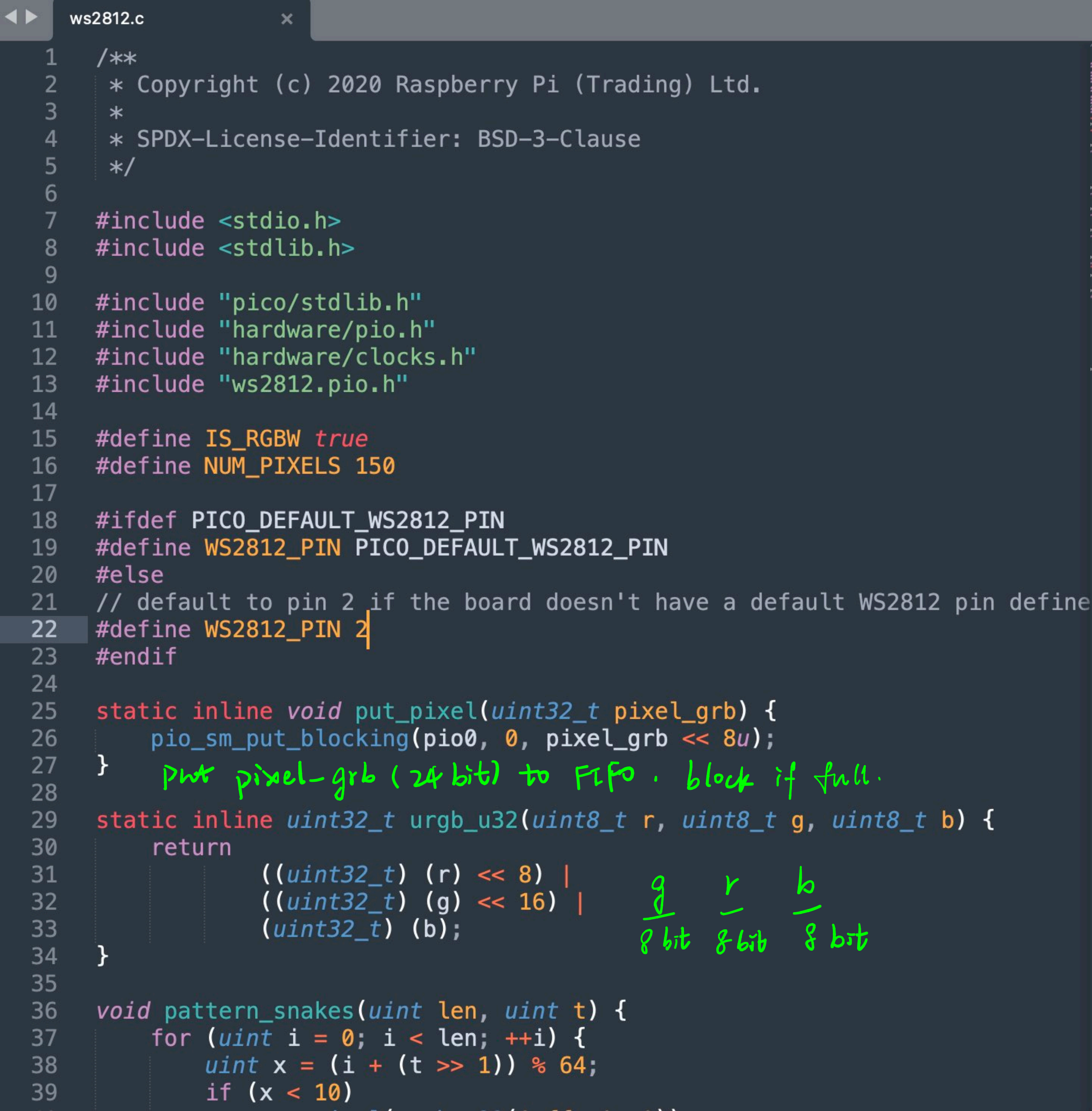


```c
/**
 * Copyright (c) 2020 Raspberry Pi (Trading) Ltd.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include <stdio.h>
#include <stdlib.h>

#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "ws2812.pio.h"

#define IS_RGBW true
#define NUM_PIXELS 150

#ifdef PICO_DEFAULT_WS2812_PIN
#define WS2812_PIN PICO_DEFAULT_WS2812_PIN
#else
// default to pin 2 if the board doesn't have a default WS2812 pin define
#define WS2812_PIN 2
#endif

static inline void put_pixel(uint32_t pixel_grb) {
    pio_sm_put_blocking(pio0, 0, pixel_grb << 8u);
}
```

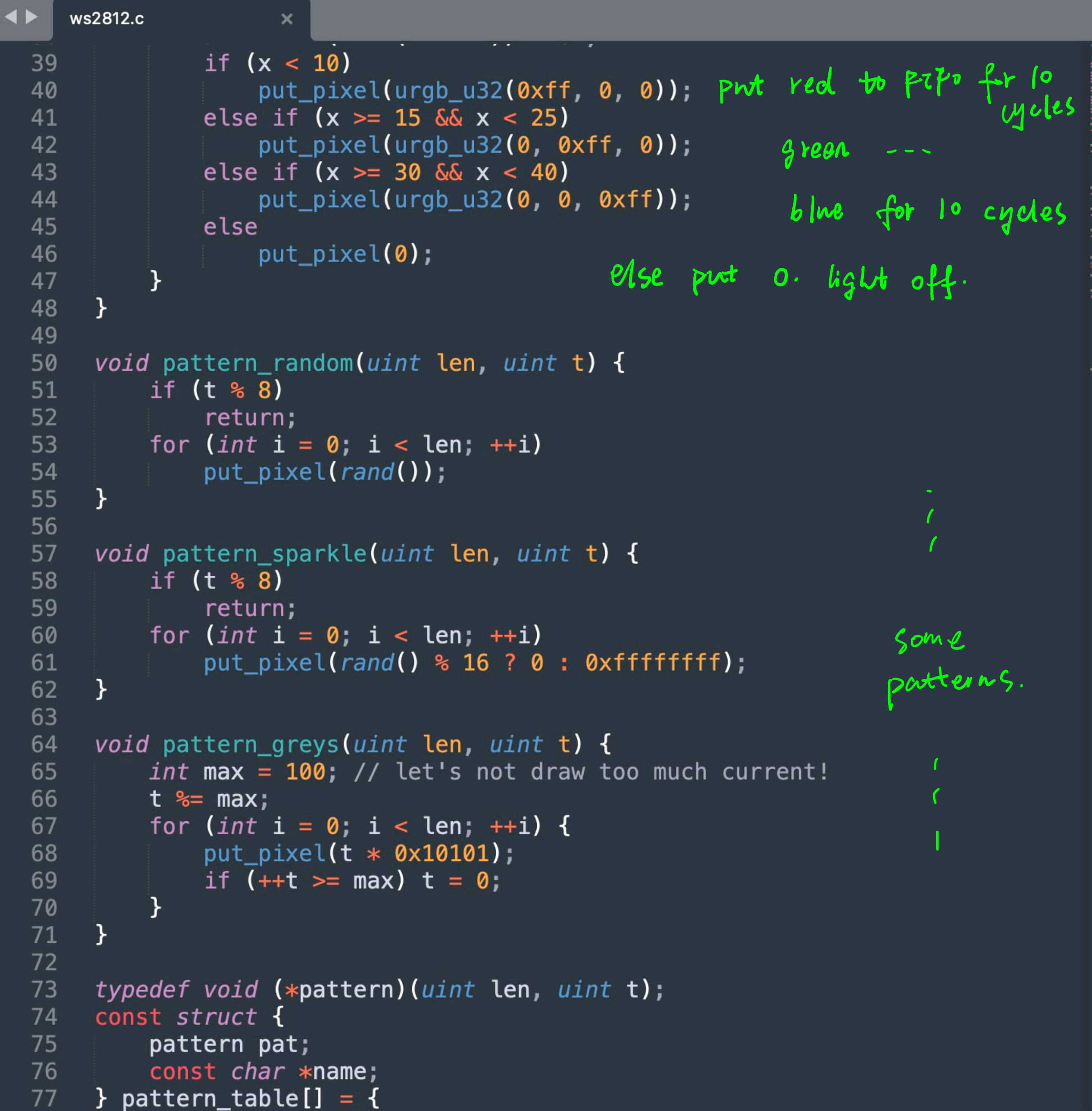
put pixel-grb (24 bit) to FIFO. block if full.

```c
static inline uint32_t urgb_u32(uint8_t r, uint8_t g, uint8_t b) {
    return
            ((uint32_t) (r) << 8) |
            ((uint32_t) (g) << 16) |
            (uint32_t) (b);
}
```

$\underline{g}$ $\underline{r}$ $\underline{b}$

8 bit 8 bit 8 bit

```c
void pattern_snakes(uint len, uint t) {
    for (uint i = 0; i < len; ++i) {
        uint x = (i + (t >> 1)) % 64;
        if (x < 10)
```



```c
        if (x < 10)
            put_pixel(urgb_u32(0xff, 0, 0));
        else if (x >= 15 && x < 25)
            put_pixel(urgb_u32(0, 0xff, 0));
        else if (x >= 30 && x < 40)
            put_pixel(urgb_u32(0, 0, 0xff));
        else
            put_pixel(0);
    }
}

void pattern_random(uint len, uint t) {
    if (t % 8)
        return;
    for (int i = 0; i < len; ++i)
        put_pixel(rand());
}

void pattern_sparkle(uint len, uint t) {
    if (t % 8)
        return;
    for (int i = 0; i < len; ++i)
        put_pixel(rand() % 16 ? 0 : 0xffffffff);
}

void pattern_greys(uint len, uint t) {
    int max = 100; // let's not draw too much current!
    t %= max;
    for (int i = 0; i < len; ++i) {
        put_pixel(t * 0x10101);
        if (++t >= max) t = 0;
    }
}

typedef void (*pattern)(uint len, uint t);
const struct {
    pattern pat;
    const char *name;
} pattern_table[] = {
```

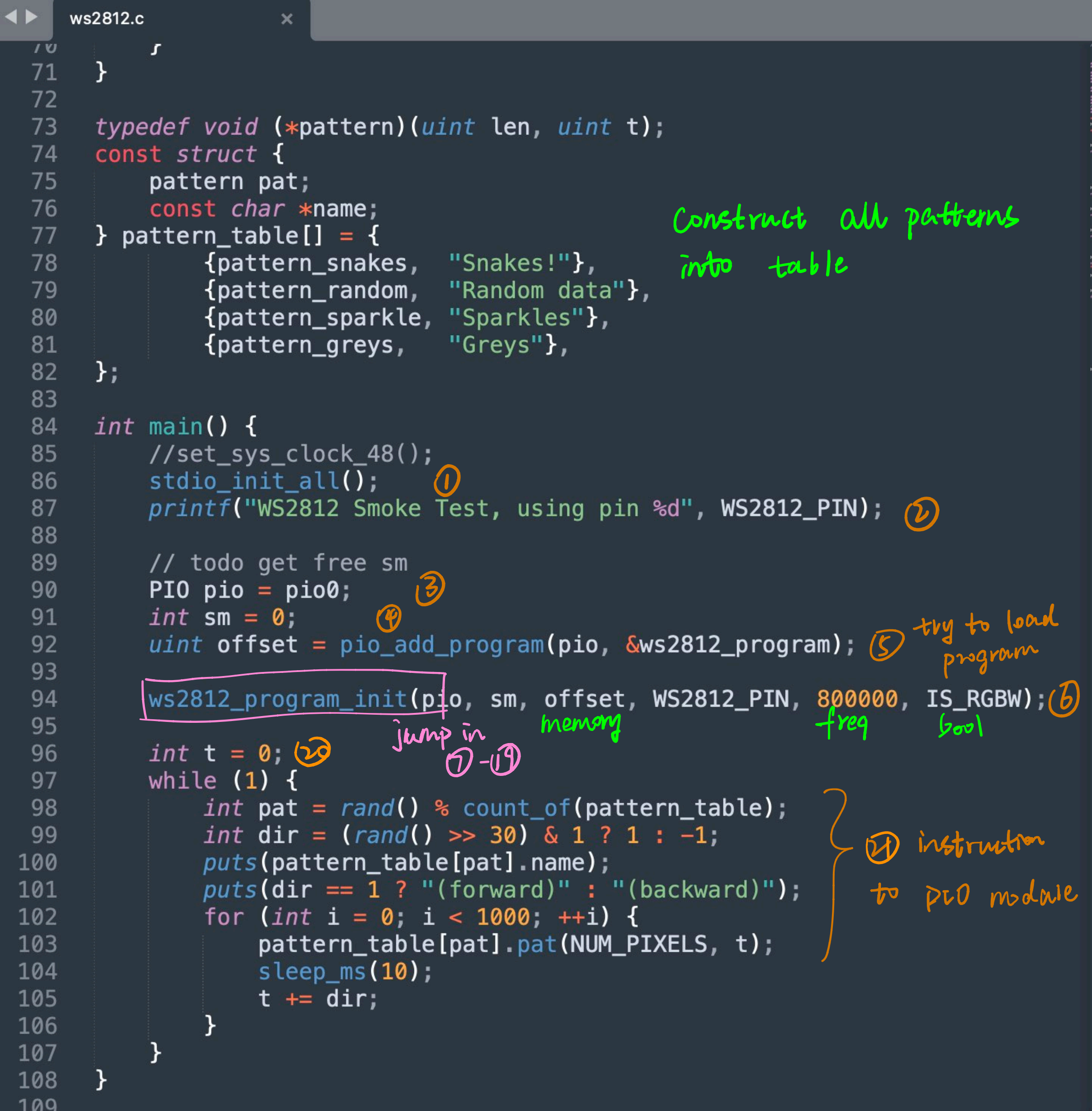
put red to FIFO for 10 cycles

green ---

blue for 10 cycles

else put 0. light off.

some patterns.



```c
        }
    }

    typedef void (*pattern)(uint len, uint t);
    const struct {
        pattern pat;
        const char *name;
    } pattern_table[] = {
            {pattern_snakes,  "Snakes!"},
            {pattern_random,  "Random data"},
            {pattern_sparkle, "Sparkles"},
            {pattern_greys,   "Greys"},
    };

    int main() {
        //set_sys_clock_48();
        stdio_init_all();
        printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);

        // todo get free sm
        PIO pio = pio0;
        int sm = 0;
        uint offset = pio_add_program(pio, &ws2812_program);

        ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);

        int t = 0;
        while (1) {
            int pat = rand() % count_of(pattern_table);
            int dir = (rand() >> 30) & 1 ? 1 : -1;
            puts(pattern_table[pat].name);
            puts(dir == 1 ? "(forward)" : "(backward)");
            for (int i = 0; i < 1000; ++i) {
                pattern_table[pat].pat(NUM_PIXELS, t);
                sleep_ms(10);
                t += dir;
            }
        }
    }
```

Construct all patterns into table

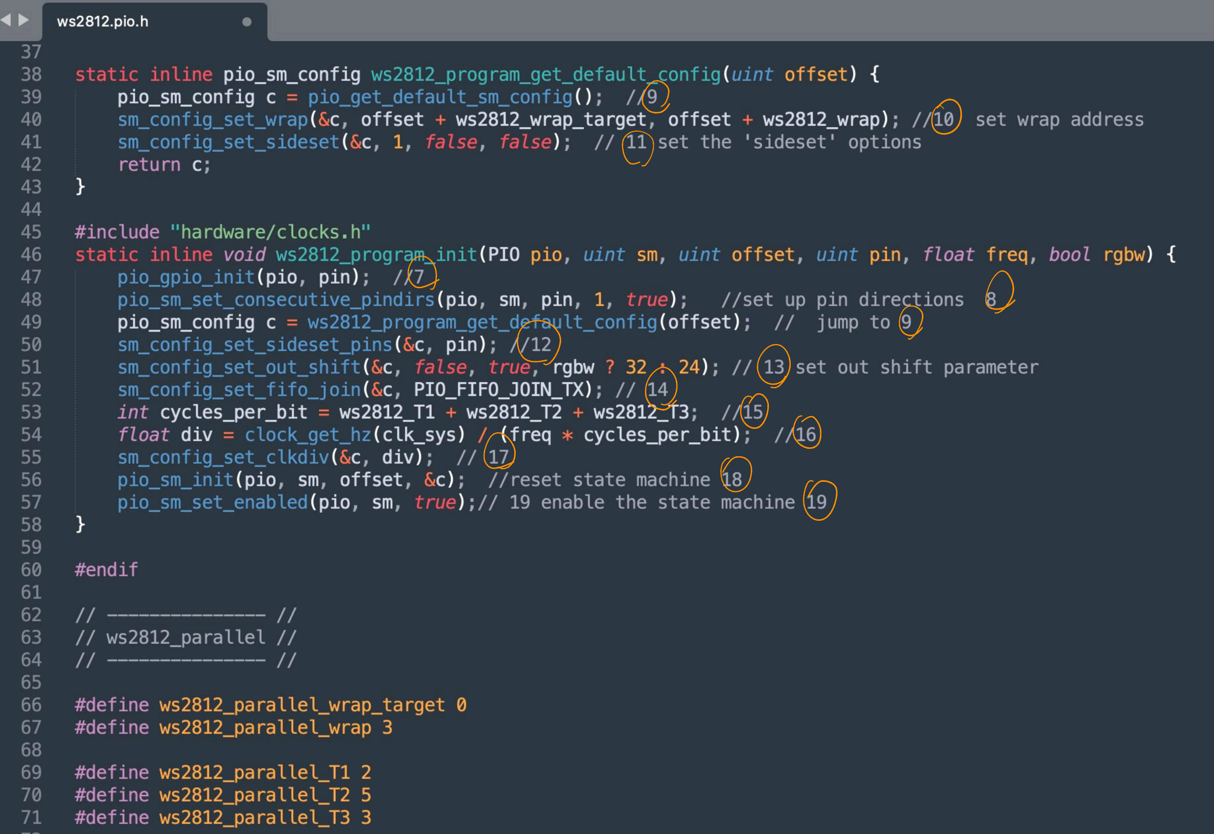
① ② ③ ④ ⑤ try to load program ⑥

memory freq bool

jump in ⑦-⑲

⑳

㉑ instruction to PIO module



```c

static inline pio_sm_config ws2812_program_get_default_config(uint offset) {
    pio_sm_config c = pio_get_default_sm_config();  //9
    sm_config_set_wrap(&c, offset + ws2812_wrap_target, offset + ws2812_wrap); //10  set wrap address
    sm_config_set_sideset(&c, 1, false, false);  // 11 set the 'sideset' options
    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
    pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true);   //set up pin directions  8
    pio_sm_config c = ws2812_program_get_default_config(offset);  //  jump to 9
    sm_config_set_sideset_pins(&c, pin); //12
    sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24); // 13 set out shift parameter
    sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX); // 14
    int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3;  //15
    float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);  //16
    sm_config_set_clkdiv(&c, div);  // 17
    pio_sm_init(pio, sm, offset, &c);  //reset state machine 18
    pio_sm_set_enabled(pio, sm, true);// 19 enable the state machine 19
}

#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
```