```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"
```

Includes necessary header files

```c
#define IS_RGBW true
#define NUM_PIXELS 150
```

Defines global constants

```c
#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 defined
#define WS2812_PIN 2
#endif
```

Comment defines block pretty well

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

← calls pio_sm_put_blocking with left shifted grb value.

→ At this point the LED should begin to glow.

```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);
}
```

↳ This function takes 8 bit R,G&B values and returns 24 bit GRB values.

G	R	B
31 16	15 8	7 0

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

→ shifts between green, red & blue light or no light to create patterns. This depends on the value of x during each iteration.

↳ R

↳ G

↳ B

```c
void pattern_random(uint len, uint t) {
31.     if (t % 8) (we assume it fails)
            return;
32.     for (int i = 0; i < len; ++i)
33.         put_pixel(rand());
}
```

→ Calls put-pixel with random colours, generated with rand(). The function doesn't execute if t is a multiple of 8.

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

← put-pixel is used to switch between white light & black light over time.

```c
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;
```

→ calls put-pixel & sets it to varying shades of the color grey.

→ limits t between 0 & 100.

```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();
1   stdio_init_all();
2   printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);

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

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

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

Holds tuples with pattern names & patterns that the program switches between

initialize state machi

Uses a bitmask to find the MSB & inturn give direction =1 if forward & -1 if backw

Uses put to simultaneously execute a patterm function and print the pattern name.

As the value is randomly generated, we will assume in the first case it will go to random pattern.

```c
// -------------------------------------------------- //
// 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
```

Defines global variables to be used

```c
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
};
```

This struct holds the instrucions in the assembly level for the ws 2812

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

Defines structure of a ws2812 program.

→ This function gets the default configuration of the ws2812. will be used for initialization

```c
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;
}

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

→ This function initializes the ws 2812 module with clock speed & basic configs.

```c
7   pio_gpio_init(pio, pin);
8   pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true);
9   pio_sm_config c = ws2812_program_get_default_config(offset);
14  sm_config_set_sideset_pins(&c, pin);
15  sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24);
```

Initializes side-set, set & FIFOs of the PIO module.

```c
16  sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
17  int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3;
18  float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
```

set the clock to be used by the PIO module

```c
19  sm_config_set_clkdiv(&c, div);
20  pio_sm_init(pio, sm, offset, &c);
21  pio_sm_set_enabled(pio, sm, true);
}

#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
```

In general this program
is essentially the same as
the above one, but it can
be used to be configured
with more than one pin
for it's functionality.