```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"
```

→ Including necessary header files

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

→ defining global constants

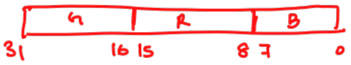
```c
#ifdef PICO_DEFAULT_WS2812_PIN
#define WS2812_PIN PICO_DEFAULT_WS2812_PIN
#else
// default to pin 2 if the board doesn't have a default WS2812 pin defined
#define WS2812_PIN 2
#endif

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

r g b

```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);
}
```

↳ Takes 8 bit R, G & B values.
returns 32 bit GRB
where



```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 b/w red, green & blue or no light depending on value of x, every iteration

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

→ calls put_pixel with random colour for "len" iterations. Returns null if t isn't divisible by 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);
}
```

→ calls put_pixel with black or white for "len" iterations. Returns null if t isn't divisible by 8.

black white

```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;
```

→ grey colour

→ limits t to a value b/w 0 & 100, & calls put_pixel with grey colour, & resets t to 0 if it exceeds max, "len" times

```
        }
    }

    typedef void (*pattern)(uint len, uint t);
    const struct {
        pattern pat;
        const char *name;
30  } pattern_table[] = {          → tuple with pattern names
            {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;          → initializing state machine.
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;   → Finds (MSB & 1) & decides if direction is
                                                                                1 or -1.
26          puts(pattern_table[pat].name);   → prints pattern name
27          puts(dir == 1 ? "(forward)" : "(backward)");   → prints forward if dir == 1
                                                                  backward if dir == -1
28          for (int i = 0; i < 1000; ++i) {
29              pattern_table[pat].pat(NUM_PIXELS, t);
                sleep_ms(10);    ↓
                t += dir;        assuming pattern goes to pattern_random()
            }                    due to value of pat
        }
    }

```

```
// -------------------------------------------------- //
// 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 times $T_1$, $T_2$, $T_3$ & other global variables

```
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
};
```

array with list of instructions for board

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

defining structure of a "program" for the WS2812

```
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;
}
```

Gets default state machine config & passes address, alongwith offsets to sm_config_set_wrap(..)

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

Function to initialize the PIO module with appropriate configs & clock speeds etc.

Also sets the calculated SM clock speed after division.

....

```
#endif

// ---------------- //
// ws2812_parallel //
// ---------------- //

#define ws2812_parallel_wrap_target 0
#define ws2812_parallel_wrap 3
```

All this portion is same as for non-parallel case above

```
#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
```

This is a similar implementation as above snippet, but is concerned with initialization when SM uses multiple pins for IO.

Rest of the function has identical implementation as part ⓘ