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

#include <stdio.h>
#include <stdlib.h>
#include <boards/adafruit_qtpy_rp2040.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
defined
#define WS2812_PIN 11
#endif

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

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);
}

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));
```

INCLUSIONS OF LIBRARY HEADERS

DEFINE NO. OF PIXELS

NEOPIXEL INPUT PIN

35

OUTPUT DATA TO NEOPIXEL USING PIO

CONVERT TO R, G, B VALUES FROM 32 BIT DATA

```
        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[] = {
        {pattern_snakes,  "Snakes!"},
        {pattern_random,  "Random data"},
        {pattern_sparkle, "Sparkles"},
        {pattern_greys,   "Greys"},
};

int main() {
    //set_sys_clock_48();
    int gpio = PICO_DEFAULT_WS2812_POWER_PIN;
    gpio_init(gpio);
    gpio_set_dir(gpio, GPIO_OUT);
    gpio_set_outover(gpio, GPIO_OVERRIDE_HIGH);
    stdio_init_all();
```

33 IF T MODULUS 8=0, RETURN

34 CALL PUT PIXEL FUNCTION IF RAN MOD16=0 = 0xffffffff ELSE =0

THE PATTERN TABLE STORES PARAMETERS FOR VARIOUS PATTERNS

1 POWER GPIO FOR WS2812

2 INITIALIZE GPIO

3 SET GPIO DIRECTION

4 PULL GPIO HIGH

5 INITIALIZES ALL GPIO'S

```
6   printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);

    // todo get free sm
7   PIO pio = pio0;
8   int sm = 0;
9   uint offset = pio_add_program(pio, &ws2812_program);
10  ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);

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

DECLARE PIO INSTANCE, & OBJECT

SET STATE MACHINE VARIABLE

↳ LOAD INTO PIO MEMORY

↳ INIT WS812 AS PIO

↗ MAP A RANDOM NO., SCALE TO 0 to 3

RIGHT SHIFT RAND NO. BY 30 & AND WITH 1 RETURN 1 IF TRUE OR -1

↓ CALL THE CORRESPONDING PATTERN

DETERMINE DIRECTION IE FORWARD OR BACKWARD

INCREMENT OR DECREMENT t WRT DIR

PATTER RANDOM = SPARKLE (ASSUME)

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

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

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

HARDWARE LIBRARY FOR GPIO

FREQUENCY VARIABLES

DEFAULT SM CONFIGURATION

WRAP ADDRESS SETUP

CURRENT GPIO CONFIGURATION FOR STATEMACHINE

```
}

#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);
  pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true);
  pio_sm_config c = ws2812_program_get_default_config(offset);
  sm_config_set_sideset_pins(&c, pin);
  sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24);
  sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
  int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_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
```

SELECT GPIO FOR PIO INSTANCE

PIN DIRECTION SET

→ PARAMETERS OF OSR REG

SET UP FIFO FOR SM

SET NO. CYCLES

STATE MACHINE CLOCK DIVIDER

INITIALIZE STATE MACHINE

ENABLE STATE MACHINE