```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 defined
#define WS2812_PIN 2
#endif

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

```

34

Send grb to output shift register (physically to Neopixel); waits until FIFO is free

```
29  static inline uint32_t urgb_u32(uint8_t r, uint8_t g, uint8_t b) {
30      return
31              ((uint32_t) (r) << 8) |
32              ((uint32_t) (g) << 16) |
33              (uint32_t) (b);
34  }
35
36  void pattern_snakes(uint len, uint t) {
37      for (uint i = 0; i < len; ++i) {
38          uint x = (i + (t >> 1)) % 64;
39          if (x < 10)
40              put_pixel(urgb_u32(0xff, 0, 0));
41          else if (x >= 15 && x < 25)
42              put_pixel(urgb_u32(0, 0xff, 0));
43          else if (x >= 30 && x < 40)
44              put_pixel(urgb_u32(0, 0, 0xff));
45          else
46              put_pixel(0);
47      }
48  }
49
50  void pattern_random(uint len, uint t) {
51      if (t % 8)
52          return;
53      for (int i = 0; i < len; ++i)
54          put_pixel(rand());
55  }
56
57  void pattern_sparkle(uint len, uint t) {
```

rgb to grb; bitwise

snake pattern with red, green and blue each for 10 cycles and no colour for remainder

random value to function to set the Neopixel colour but only 1/8 of the number of times

a case when t is divisible by 8

```
57  void pattern_sparkle(uint len, uint t) {
58      if (t % 8)
59          return;
60      for (int i = 0; i < len; ++i)
61          put_pixel(rand() % 16 ? 0 : 0xffffffff);
62  }
63
64  void pattern_greys(uint len, uint t) {
65      int max = 100; // let's not draw too much current!
66      t %= max;
67      for (int i = 0; i < len; ++i) {
68          put_pixel(t * 0x10101);
69          if (++t >= max) t = 0;
70      }
71  }
72
73  typedef void (*pattern)(uint len, uint t);
74  const struct {
75      pattern pat;
76      const char *name;
77  } pattern_table[] = {
78          {pattern_snakes,  "Snakes!"},
79          {pattern_random,  "Random data"},
80          {pattern_sparkle, "Sparkles"},
81          {pattern_greys,   "Greys"},
82  };
83
84  int main() {
85      //set_sys_clock_48();
```

variation of the previous function to display very bright colour intermittently

dims the light and then restarts and continues to do the same

function pointer for any of the defined light pattern functions

```
83
84  int main() {
85      //set_sys_clock_48();
86  1   stdio_init_all();
87  2   printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);
88
89      // todo get free sm
90  3   PIO pio = pio0;
91  4   int sm = 0;
92  5   uint offset = pio_add_program(pio, &ws2812_program);
93
94  6   ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);
95
96  22  int t = 0;
97  23  while (1) {
98  24      int pat = rand() % count_of(pattern_table);
99  25      int dir = (rand() >> 30) & 1 ? 1 : -1;
100 26      puts(pattern_table[pat].name);
101 27      puts(dir == 1 ? "(forward)" : "(backward)");
102 28      for (int i = 0; i < 1000; ++i) {
103 29          pattern_table[pat].pat(NUM_PIXELS, t);
104             sleep_ms(10);
105             t += dir;
106         }
107     }
108 }
```

initialisation

display pin info

SM and pio selection; and offset for the pio to add this program

To transmit the program, configure SM

Patterns are randomly flashed

```
// -------------------------------------------------- //
// 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
```

We model the time for these pio instructions

```
29  };
30
31  #if !PICO_NO_HARDWARE
32  static const struct pio_program ws2812_program = {
33      .instructions = ws2812_program_instructions,
34      .length = 4,
35      .origin = -1,
36  };
37
38  static inline pio_sm_config ws2812_program_get_default_config(uint offset) {
39      pio_sm_config c = pio_get_default_sm_config();
40      sm_config_set_wrap(&c, offset + ws2812_wrap_target, offset + ws2812_wrap);
41      sm_config_set_sideset(&c, 1, false, false);
42      return c;
43  }
44
45  #include "hardware/clocks.h"
46  static inline void ws2812_program_init(PIO pio, uint sm, uint offset, uint pin, float freq, bool rgbw) {
47      pio_gpio_init(pio, pin);
48      pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true);
49      pio_sm_config c = ws2812_program_get_default_config(offset);
50      sm_config_set_sideset_pins(&c, pin);
51      sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24);
52      sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
53      int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3;
54      float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
55      sm_config_set_clkdiv(&c, div);
56      pio_sm_init(pio, sm, offset, &c);
57      pio_sm_set_enabled(pio, sm, true);
```

10 The default settings and structure of the state machine

11 setting wrap address

12

13

7 initialising with GPIO pin assignment

8 number and direction of pin for SM

9

14

15 OSR register parameters

16 set FIFO

17

18 clock divider

19

20 initialise and enable state machine

21

```
57      pio_sm_set_enabled(pio, sm, true);
58  }
59
60  #endif
61
62  // --------------- //
63  // ws2812_parallel //
64  // --------------- //
65
66  #define ws2812_parallel_wrap_target 0
67  #define ws2812_parallel_wrap 3
68
69  #define ws2812_parallel_T1 2
70  #define ws2812_parallel_T2 5
71  #define ws2812_parallel_T3 3
72
73  static const uint16_t ws2812_parallel_program_instructions[] = {
74              //     .wrap_target
75      0x6020, //  0: out    x, 32
76      0xa10b, //  1: mov    pins, !null           [1]
77      0xa401, //  2: mov    pins, x               [4]
78      0xa103, //  3: mov    pins, null            [1]
79              //     .wrap
80  };
81
82  #if !PICO_NO_HARDWARE
83  static const struct pio_program ws2812_parallel_program = {
84      .instructions = ws2812_parallel_program_instructions,
85      .length = 4,
```

```
85      .length = 4,
86      .origin = -1,
87  };
88
89  static inline pio_sm_config ws2812_parallel_program_get_default_config(uint offset) {
90      pio_sm_config c = pio_get_default_sm_config();
91      sm_config_set_wrap(&c, offset + ws2812_parallel_wrap_target, offset + ws2812_parallel_wrap);
92      return c;
93  }
94
95  #include "hardware/clocks.h"
96  static inline void ws2812_parallel_program_init(PIO pio, uint sm, uint offset, uint pin_base, uint pin_count, float freq) {
97      for(uint i=pin_base; i<pin_base+pin_count; i++) {
98          pio_gpio_init(pio, i);
99      }
100     pio_sm_set_consecutive_pindirs(pio, sm, pin_base, pin_count, true);
101     pio_sm_config c = ws2812_parallel_program_get_default_config(offset);
102     sm_config_set_out_shift(&c, true, true, 32);
103     sm_config_set_out_pins(&c, pin_base, pin_count);
104     sm_config_set_set_pins(&c, pin_base, pin_count);
105     sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
106     int cycles_per_bit = ws2812_parallel_T1 + ws2812_parallel_T2 + ws2812_parallel_T3;
107     float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
108     sm_config_set_clkdiv(&c, div);
109     pio_sm_init(pio, sm, offset, &c);
110     pio_sm_set_enabled(pio, sm, true);
111 }
112
113 #endif
```