

```

1  /**
2   * Copyright (c) 2020 Raspberry Pi (Trading) Ltd.
3   *
4   * SPDX-License-Identifier: BSD-3-Clause
5   */
6
7  #include <stdio.h>
8  #include <stdlib.h>
9
10 #include "pico/stdlib.h"
11 #include "hardware/pio.h"
12 #include "hardware/clocks.h"
13 #include "ws2812.pio.h"
14
15 #define IS_RGBW true
16 #define NUM_PIXELS 150
17
18 #ifdef PICO_DEFAULT_WS2812_PIN
19 #define WS2812_PIN PICO_DEFAULT_WS2812_PIN
20 #else
21 // default to pin 2 if the board doesn't have a default WS2812 pin defined
22 #define WS2812_PIN 2
23 #endif
24
25 static inline void put_pixel(uint32_t pixel_grb) {
26     pio_sm_put_blocking(pio0, 0, pixel_grb << 8u);
27 }
28
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 }

```

①⑨ turn on LED light

①⑧ change r,g,b to GRB sequence

```

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 void pattern_random(uint len, uint t) {
50     if (t % 8)
51         return;
52     for (int i = 0; i < len; ++i)
53         put_pixel(rand());
54 }

55 void pattern_sparkle(uint len, uint t) {
56     if (t % 8)
57         return;
58     for (int i = 0; i < len; ++i)
59         put_pixel(rand() % 16 ? 0 : 0xffffffff);
60 }

61 void pattern_greys(uint len, uint t) {
62     int max = 100; // let's not draw too much current!
63     t %= max;
64     for (int i = 0; i < len; ++i) {
65         put_pixel(t * 0x10101);
66         if (++t >= max) t = 0;
67     }
68 }
69
70
71

```

①⑦ led color change  
from red → green  
↓  
blue

①⑦ led color change  
randomly

①⑦ led randomly turn on  
white light

①⑦ led turn on  
grey light all  
the time

```

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();
86     stdio_init_all(); ①
87     printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);
88     // print out WS2812 Pin's number
89     // todo get free sm
90     PIO pio = pio0; ③ choose pio0 instance to use
91     int sm = 0;
92     uint offset = pio_add_program(pio, &ws2812_program);
93     // load assembled program into PIO's instruction memory
94     ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);
95     // ⑤ Jump
96     int t = 0;
97     while (1) {
98         int pat = rand() % count_of(pattern_table);
99         int dir = (rand() >> 30) & 1 ? 1 : -1;
100         puts(pattern_table[pat].name);
101         puts(dir == 1 ? "(forward)" : "(backward)");
102         for (int i = 0; i < 1000; ++i) {
103             pattern_table[pat].pat(NUM_PIXELS, t);
104             sleep_ms(10);
105             t += dir;
106         }
107     }
108 }
109

```

② randomly choose a pattern and direction to print out.

④ display the pattern on led

```

5  #pragma once
6
7  #if !PICO_NO_HARDWARE
8  #include "hardware/pio.h"
9  #endif
10
11 // ----- //
12 // ws2812 //
13 // ----- //
14
15 #define ws2812_wrap_target 0
16 #define ws2812_wrap 3
17
18 #define ws2812_T1 2
19 #define ws2812_T2 5
20 #define ws2812_T3 3
21
22 static const uint16_t ws2812_program_instructions[] = {
23     |      |      //      .wrap_target
24     0x6221, // 0: out    x, 1          side 0 [2]
25     0x1123, // 1: jmp    !x, 3          side 1 [1]
26     0x1400, // 2: jmp    0          side 1 [4]
27     0xa442, // 3: nop                    side 0 [4]
28     |      |      //      .wrap
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 }

```

```

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); ① configure a GPIO for use by PIO
48     pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true); ⑦ set the pin direction to output at the PIO
49     pio_sm_config c = ws2812_program_get_default_config(offset); ⑧ get the default configuration using
50     sm_config_set_sideset_pins(&c, pin); ← the generated function for this
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); ⑬ load configuration
57     pio_sm_set_enabled(pio, sm, true); ⑭ enable the state machine
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,
86     .origin = -1,
87 };

```

⑦ set the pin direction to output at the PIO

⑧ get the default configuration using the generated function for this program

⑨ sets the side-set to write at pin

⑩ false for shift-to-right true for autopull 32 or 24 for the number of bits for the autopull threshold

⑪ get the total number of execution cycle to output a single bit

⑫ achieve correct bit rate

⑬ load configuration into state machine

⑭ enable the state machine

```

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

```