```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/gpio.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);
}

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));
        else
            put_pixel(0);
    }
}

void pattern_random(uint len, uint t) {
    if (t % 8)
        return;
    for (int i = 0; i < len; ++i)
```

Include other

use the value defined in the board header (isn't one for QT Py)

Convert 3 8-bit numbers into a 24-bit color

Shift "red" left by 8 bits

0 0 0000000|00000000|00000000|00000000
31 empty 24|23 Green 16|15 red 8|7 blue 0

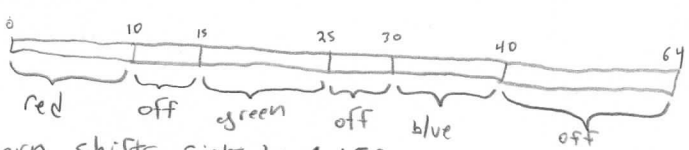
Number of pixels

"frame" number

every 2 "frames" (t>>1 divides t by 2), change the LED to the next state. Pattern is 64 LEDs long.

Based on x, set the color of the LED specified by i.

On frame 0, the pattern appears as below:



numbers

Pattern shifts right by 1 LED every 2 frames ("t")

only proceed + set the color every 8 frames

```
55          put_pixel(rand());
56      }
57
58      void pattern_sparkle(uint len, uint t) {
59          if (t % 8)
60              return;
61          for (int i = 0; i < len; ++i)
62              put_pixel(rand() % 16 ? 0 : 0xffffffff);
63      }
64
65      void pattern_greys(uint len, uint t) {
66          int max = 100; // let's not draw too much current!
67          t %= max;
68          for (int i = 0; i < len; ++i) {
69              put_pixel(t * 0x10101);
70              if (++t >= max) t = 0;
71          }
72      }
73
74      typedef void (*pattern)(uint len, uint t);
75      const struct {
76          pattern pat;
77          const char *name;
78      } pattern_table[] = {
79              {pattern_snakes,  "Snakes!"},
80              {pattern_random,  "Random data"},
81              {pattern_sparkle, "Sparkles"},
82              {pattern_greys,   "Greys"},
83      };
84
85      int main() {
86          //set_sys_clock_48();
87          stdio_init_all();
88          gpio_init(PICO_DEFAULT_WS2812_POWER_PIN);
89          gpio_set_dir(PICO_DEFAULT_WS2812_POWER_PIN, GPIO_OUT);
90          gpio_put(PICO_DEFAULT_WS2812_POWER_PIN, 1);
91          printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);
92
93          // todo get free sm
94          PIO pio = pio0;
95          int sm = 0;
96          uint offset = pio_add_program(pio, &ws2812_program);
97
98          ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);
99
100         int t = 0;
101         while (1) {
102             int pat = rand() % count_of(pattern_table);
103             int dir = (rand() >> 30) & 1 ? 1 : -1;
104             puts(pattern_table[pat].name);
105             puts(dir == 1 ? "(forward)" : "(backward)");
106             for (int i = 0; i < 1000; ++i) {
107                 pattern_table[pat].pat(NUM_PIXELS, t);
108                 sleep_ms(10);
```

send a random color. put_pixel will take care of eliminating the extra bits

only change every 8 frames

for a random selection of 1/16 of LEDs, turn OFF. Otherwise, white.

repeat pattern every 100 frames

increment t, then check value. reset to 0

define function pointer

define a struct containing the func. ptr and a name string

fill an array of these structs

init the UART interface

set the power pin for the LED as GPIO

set power pin as an output

write 1 to pin (set it high)

write text to the console

select which PIO to use

select which state machine

load the pio instructions into the pio

set up pio pins, clock, etc.

select a random pattern

advance forward or backward through frames

print pattern name to console

print direction to console

cycle through 1000 frames

call the function for the selected pattern.

wait

go to pattern_sparkle for program trace

```
109             t += dir;
110         }
111     }
112 }
```

increment or decrement the frame

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

#include "hardware/clocks.h"
static inline void ws2812_program_init(PIO pio, uint sm, uint offset, uint pi
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;
```

only process this file once

include header file

define constants

array of pio instructions

struct describing the pio program

sets up the state machine

This function sets up

link the GPIO to the pio

set the pin to an output

get the pio state machine state

set SM sideset pins, FIFO type, and clock divider

```
    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

// -------------- //
// 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 o
    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
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);
```

22 23 24 25

write the modified config to the SM

enable the state machine

not used by ws2812.C

```
    int cycles_per_bit = ws2812_parallel_T1 + ws2812_parallel_T2 + ws2812_par
    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

```