Some of code have already be amotated with "//" command added by Yuxuan

WS2812.c

C:\Users\24664\Desktop\ws2812.c 1

```
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 //RGB+W LED strip uses either a 4-in-1 LED chip that has a white chip as well as red,
16 //green and blue or an RGB chip alongside a white chip.
17 #define IS_RGBW true
18 #define NUM_PIXELS 150
19
20 #ifdef PICO_DEFAULT_WS2812_PIN
21 #define WS2812_PIN PICO_DEFAULT_WS2812_PIN
22 #else
23 // default to pin 2 if the board doesn't have a default WS2812 pin defined
24 #define WS2812_PIN 2
25 #endif
26
27 static inline void put_pixel(uint32_t pixel_grb) {
28     pio_sm_put_blocking(pio0, 0, pixel_grb << 8u);
29 }
30
31 static inline uint32_t urgb_u32(uint8_t r, uint8_t g, uint8_t b) {
32     return
33             ((uint32_t) (r) << 8) |
34             ((uint32_t) (g) << 16) |
35             (uint32_t) (b);
36 }
37
38 void pattern_snakes(uint len, uint t) {
39     for (uint i = 0; i < len; ++i) {
40         uint x = (i + (t >> 1)) % 64;
41         if (x < 10)
42             put_pixel(urgb_u32(0xff, 0, 0));
43         else if (x >= 15 && x < 25)
44             put_pixel(urgb_u32(0, 0xff, 0));
45         else if (x >= 30 && x < 40)
46             put_pixel(urgb_u32(0, 0, 0xff));
47         else
48             put_pixel(0);
49     }
50 }
51
52 void pattern_random(uint len, uint t) {
53     if (t % 8)
54         return;
55     for (int i = 0; i < len; ++i)
56         put_pixel(rand());
```

support libraries

output the shifted pixel values

function for setup pio 0

put the pixel_rgb into ws2812

Helper method: If FIFO has room, then push the data, otherwise wait

0x00000000ff

r 0000ff00

g 00ff0000

b 00000000ff

r,g,b value are 8 ~~bytes~~ bits

~~r,g,b,w value~~ put the r,g,b value and combine them into ~~rgbw~~ value

grbW

put red to FIFO for 10 cycles

Set snack color mode according to different value of t within 'RGB' range

green

blue

light off

set the color of the pattern randomly

(210 - 32)

```
57  }
58
59  void pattern_sparkle(uint len, uint t) {
60      if (t % 8)
61          return;
62      for (int i = 0; i < len; ++i)
63          put_pixel(rand() % 16 ? 0 : 0xffffffff);
64  }
65
66  void pattern_greys(uint len, uint t) {
67      int max = 100; // let's not draw too much current!
68      t %= max;
69      for (int i = 0; i < len; ++i) {
70          put_pixel(t * 0x10101);
71          if (++t >= max) t = 0;
72      }
73  }
74
75  typedef void (*pattern)(uint len, uint t);
76  const struct {
77      pattern pat;
78      const char *name;
79  } pattern_table[] = {
80          {pattern_snakes,   "Snakes!"},
81          {pattern_random,   "Random data"},
82          {pattern_sparkle,  "Sparkles"},
83          {pattern_greys,    "Greys"},
84  };
85
86  int main() {
87      //set_sys_clock_48();
88      stdio_init_all();
89      printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);
90
91      // todo get free sm
92      PIO pio = pio0;
93      int sm = 0;
94      uint offset = pio_add_program(pio, &ws2812_program);
95
96      ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);
97
98      int t = 0;
99      while (1) {
100         int pat = rand() % count_of(pattern_table);
101         int dir = (rand() >> 30) & 1 ? 1 : -1;
102         puts(pattern_table[pat].name);
103         puts(dir == 1 ? "(forward)" : "(backward)");
104         for (int i = 0; i < 1000; ++i) {
105             pattern_table[pat].pat(NUM_PIXELS, t);
106             sleep_ms(10);
107             t += dir;
108         }
109     }
110 }
111
```

① int rand (void): returns a pseudo-random number in the range of 0 to RAND_MAX

Implement the sparkle pattern for LED

pattern greys
GRB
0x01 01 01
to 0x646464
↓
grey

Implement the grey pattern for LED

with 'void' type return value → define a pointer named pattern with a pattern table to shift the pattern light

→ constructor definition

tuple with Pattern names

We can assign values to a specific item lens equal to NUM_PIXELS

① Initial all standard I/O

② corresponding to Pin#2

③ A new pio instance variable

④ A new free state machine variable

⑤ Add the 5 bit ws2812 program address to the pio instance

⑥ Initiate the WS2812 program by finding if there are free space for pio, sum, offset.

⑦ corresponding to while true as a continuous loop that work constantly

random choose a pat

For each time choose a pattern and output the color value to WS2812.

judge the 2 bits.

→ use the random number to set the 'dir' value

→ dir can only take two possible values

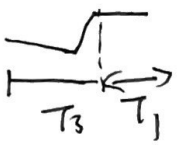
→ add the value of 1 or −1 into value t

choose random color.

and so the value of t changes randomly since dir is arbitary

```
1  // -------------------------------------------------- //
2  // This file is autogenerated by pioasm; do not edit! //
3  // -------------------------------------------------- //
4
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 }
44
45 #include "hardware/clocks.h"
46 static inline void ws2812_program_init(PIO pio, uint sm, uint offset, uint pin, float f
    req, 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);
```

GPIO (0,1)



← Include the necessary hardware/pio.h if there is no 'PICO_NO_HARDWARE' flag

array with list of instructions for board

(0) This Command to shift a 1 bit data into register x, set the side-set pin to low (0). and wait for 2 delay cycles before next instruction

(1) Jump to 3 when x==0, otherwise go(2) set side-pins(1) and wait for 1 cycles before next instruction

(2) Jump to 0 and set side-pins to pulse(1) and wait for 4 clock cycles for the next instructions

(3) Do nothing and set side-set pin to 0 and wait for 4 cycles for the next instruction

Define the instruction length and origin of WS2812_program_instructions

set wrap addres in a sm configure

→ OutPin (32), set Pin (0) and side set (disabled) etc...

→ pindirs

set up the state machine wrap and sideset based on offset

pointer to configure struct

bit-count

optional

configure'd with WS 2812 program default

Use the state machine to set the pin direction for the 1 pin

first pin

set the Sideset and Pin

set the FIFO join value

→ calculate clock divider for sm

calculate clock div and set state machine

```
56  pio_sm_init(pio, sm, offset, &c);
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,
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);
```

(20) Like the configuration to initiate the state machine and move PC pointer to offset

(21) Enable the state machine

The code to set up the parallel similar program

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

Sets the calculated SM clock speed after division