

```

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

```

```

#include <stdio.h>
#include <stdlib.h>

```

Including
header files
needed to run
the code

```

#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "ws2812.pio.h"

```

```

#define IS_RGBW true
#define NUM_PIXELS 150

```

defining
constants

```

#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

```

checking if
default pin is
assigned in one
of the headers

```

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

```

defining function put_pixel, using it
static and inline

```

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);
}

```

static
inline
return

32 bit unsigned guaranteed
move 8 bits left 8 bits
move 8 bits left 16
32 bit unsigned guaranteed

reorganizing
bits basically
encoding as 24 bits/bit #

29

```

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

```

Changing
the LED
Value based
on x
to make
pattern

Creating a Pattern for led

Setting to certain color
Setting to certain color
Setting to certain color

```

        put_pixel(0); Setting led to 0
    }
}

```

defining function

```

void pattern_random(uint len, uint t) {
    if (t % 8) if t/8 = 0 return
        return;
    for (int i = 0; i < len; ++i) defining till I is certain value
        put_pixel(rand()); Setled random value
}

```

defining function

```

void pattern_sparkle(uint len, uint t) {
    if (t % 8)
        return;
    for (int i = 0; i < len; ++i) Same as above
        put_pixel(rand() % 16 ? 0 : 0xffffffff); Pick random value between 0, 16
}

```

if rand / 16 = 0 then

defining

```

void pattern_greys(uint len, uint t) {
    int max = 100; // let's not draw too much current! Setting max value of 100%
    t %= max;
    for (int i = 0; i < len; ++i) defining till len length
        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"},
};

```

Building a vector with each pattern

4 patterns defined previously

```

int main() {
    //set_sys_clock_48(); Commented out
    stdio_init_all(); ①
    printf("WS2812 Smoke Test, using pin %d", WS2812_PIN); ② Print which pin its using

    // todo get free sm
    PIO pio = pio0; ③ Setting the PIO to pio0
}

```

④ first PIO hardware instance

initialize all stdio + VFS linked

`int sm = 0;` ^{declaring an integer}
`uint offset = pio_add_program(pio, &ws2812_program);` ^{Program name}

`ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);` ^{trying to load program}

`int t = 0;` ^{new int}

`while (1) {`

`int pat = rand() % count_of(pattern_table);` ^{randomly pick which pattern}

`int dir = (rand() >> 30) & 1 ? 1 : -1;` ^{pick random number}

`puts(pattern_table[pat].name);` ^{pick pattern in array}

`puts(dir == 1 ? "(forward)" : "(backward)");` ^{if dir = 1 then forward}

`for (int i = 0; i < 1000; ++i) {` ^{go for 1000 iterations}

`pattern_table[pat].pat(NUM_PIXELS, t);`

`sleep_ms(10);` ^{sleep momentarily 150}

`t += dir;` ^{add dir to t}

`}`

`}`

`}`

Initialize Program

while loop to randomize for LED

pulling from function other printout

```
// ----- //
// This file is autogenerated by pioasm; do not edit! //
// ----- //
```

```
#pragma once
```

```
#if !PICO_NO_HARDWARE
#include "hardware/pio.h"
#endif
```

Checking if hardware is ready
defined then defining it if not

```
// ----- //
// ws2812 //
// ----- //
```

```
#define ws2812_wrap_target 0
#define ws2812_wrap 3
```

defining
variables

```
#define ws2812_T1 2
#define ws2812_T2 5
#define ws2812_T3 3
```

```
static const uint16_t ws2812_program_instructions[] = {
```

locating

	//	.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	

Completed out

```
};
```

```
#if !PICO_NO_HARDWARE
```

```
static const struct pio_program ws2812_program = {
    .instructions = ws2812_program_instructions,
    .length = 4,
    .origin = -1,
};
```

if hardware not defined
then using program
instructions from above

```
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;
```

Gets the wrap and sideset
options in the state machine
that it pulled from default config

```
#include "hardware/clocks.h"
```

Include clocks

no output

Init Program based on previous fixed values

```
static inline void ws2812_program_init(PIO pio, uint sm, uint offset, uint pin, float freq, bool rgbw) {
```

init pin pio_gpio_init(pio, pin);

pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true); *Setting consecutive pin directions*

pio_sm_config c = ws2812_program_get_default_config(offset); *Setting C to default config*

sm_config_set_sideset_pins(&c, pin); *Setting sideset pins*

sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24); *Setting up outshifting*

sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX); *Setting up Fifo joining in state machine*

int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3;

float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit); *Calculating value so you can set the clock div*

sm_config_set_clkdiv(&c, div); *Set state machine clock divider*

pio_sm_init(pio, sm, offset, &c); *Resets and configures to constant state*

pio_sm_set_enabled(pio, sm, true); *Enables state machine*

```
}
```

```
#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
```

*Same as above
but for parallel example
which we didn't do*

```
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
offset) {
    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,
uint 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);
    int cycles_per_bit = ws2812_parallel_T1 + ws2812_parallel_T2 +
ws2812_parallel_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

```