WS2812.C

add libraries for this file

Define values for later use

Define led gpio pin, default pin : 2

```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"
#include "boards/adafruit_qtpy_rp2040.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);
}
```

urgb_32 : organize the r,g,b values as they were introduced in module.

put pixel bits into FIFO, starting with the MSB, that is, to put 24 color value at top.

pattern_snakes :

for loop : i=0 ~ 149

t range to be defined in main

four color pattern show in pattern as x increases : g→r→b→off

```c
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);
    }
}
```

→ g
→ r
→ b
→ off

pattern_random :

A undefined random color pattern.

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

→ random color value

pattern_sparkle :

pixel sparkles (all color sparkle) when random value mod 16 ; pixel off otherwise.

```c
void pattern_sparkle(uint len, uint t) {
    if (t % 8)
        return;
    for (int i = 0; i < len; ++i)
        put_pixel(rand() % 16 ? 0 : 0xffffffff);
}
```

→ off / all on.

pattern_greys

pixel value : 0x 01 01 01 (G R B)

to 0x 64 64 64

↓ grey

```c
void pattern_greys(uint len, uint t) {
    int max = 100; // let's not draw too much current!
    t %= max;
    for (int i = 0; i < len; ++i) {
        put_pixel(t * 0x10101);
        if (++t >= max) t = 0;
    }
}
```

Construct a pattern table that includes all patterns

Call each one by its name while needed.

initialize stdio.

print PIN used to control pixel

use pio.0

Sm : State machine being configured to run

offset: PIO program address space

WS2812_PIN : GPIO used to connect PIO

800000 : baud rate freq

IS_RGBW : True if using 4 color mode

Count:

```c
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"},
};

int main() {
    //set_sys_clock_48();
    stdio_init_all();
    printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);

    // todo get free sm
    PIO pio = pio0;
    int sm = 0;
    uint offset = pio_add_program(pio, &ws2812_program);

    ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);

    // Power pin true
    gpio_init(PICO_DEFAULT_WS2812_POWER_PIN);
    gpio_set_dir(PICO_DEFAULT_WS2812_POWER_PIN, GPIO_OUT);
    gpio_put(PICO_DEFAULT_WS2812_POWER_PIN, 1);
    // PIO pio_1 = pio1;
    // pio_gpio_init(pio_1, PICO_DEFAULT_WS2812_POWER_PIN);
    // gpio_put(PICO_DEFAULT_WS2812_POWER_PIN, 1);

    int t = 0;
    while (1) {
        int pat = rand() % count_of(pattern_table);
        int dir = (rand() >> 30) & 1 ? 1 : -1;
        puts(pattern_table[pat].name);
        puts(dir == 1 ? "(forward)" : "(backward)");
        for (int i = 0; i < 1000; ++i) {
            pattern_table[pat].pat(NUM_PIXELS, t);
            sleep_ms(10);
            t += dir;
        }
    }
}
```

Count:

initialize power pin for WS2812

set direction as out

put 1, true to enable PIN.

Run for a pattern selected in table.

switch / hold pattern every (10ms)

WS2812.pio.h only included on

Defined from WS2812.pio

Constants used to calculate delay cycles

Assembly :

① out : take out bits from OSR, write them somewhere else.

Side 0 : Drive low the pin for side-set

[2] : [T3 - 1] = 3 - 1 = [2]

X : One of two scratch reg ↓ used in SM to hold and compare temporary data destination.

```c
// -------------------------------------------------- //
// 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,
};
```

① : jmp : unconditional jump to switch to a different piece of code

!X 3 : if X=0, go to 3 (WS2812_sum) Continue otherwise

side 1 [1] : set high on pin [T1 - 1] delay

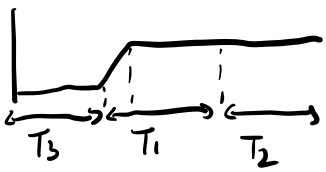
Continue this piece of code.

② unconditionally jump back to bit loop start (0)

Side 1 : Continue as high

[4] = [T2 - 1]

Now GPIO ⓪→①→② looks:



③ nop : no operation

side 0 [4] : side low [T2-1]

Now GPIO ⓪→①→③ looks:


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

0 → 3 : four instructions total starting from 0

④ .wrap : Final line.

State-machine configuration.

set wrap target & sideset (0~5)

WS2812 initialization:

set pio & GPIO connected

pin direction out

Cycle delay: T1 + T2 + T3

```
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 pin, float 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;
    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
```

data

set offset

sideset

FIFO used to write

Calculate clock.

Enable statemachine (0~4). ready to be used.