/\*\*

 \* Copyright (c) 2020 Raspberry Pi (Trading) Ltd.

 \*

 \* SPDX-License-Identifier: BSD-3-Clause

 \*/

#include <stdio.h> // Header files included in the code

#include <stdlib.h>

#include <boards/adafruit\_qtpy\_rp2040.h>

#include "pico/stdlib.h"

#include "hardware/pio.h"

#include "hardware/clocks.h"

#include "ws2812.pio.h"

#define IS\_RGBW true

#define NUM\_PIXELS 150 // 150 denotes the number of pixels

#ifdef PICO\_DEFAULT\_WS2812\_PIN

#define WS2812\_PIN PICO\_DEFAULT\_WS2812\_PIN

#else

#define WS2812\_PIN 12 // WS2812 LED PIN number of 12 from datasheet

#endif

// default to pin 2 if the board doesn't have a default WS2812 pin defined

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)); //LED color changing logic

        else if (x >= 15 && x < 25) // depending on value of x

            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)

        put\_pixel(rand());

}

void pattern\_sparkle(uint len, uint t) {

    if (t % 8)

        return;

    for (int i = 0; i < len; ++i)

        put\_pixel(rand() % 16 ? 0 : 0xffffffff);

}

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;

    }

}

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();

    int gpio = PICO\_DEFAULT\_WS2812\_POWER\_PIN; //Pin 12 is def as power pin

    gpio\_init(gpio);

    gpio\_set\_dir(gpio, GPIO\_OUT); //GPIO direction is set to output

    gpio\_put(gpio,1); //GPIO is set as high

    stdio\_init\_all();

    printf("WS2812 Smoke Test, using pin %d", WS2812\_PIN);

    // todo get free sm

    PIO = pio0; // Uses PIO0

    int sm = 0;

    uint offset = pio\_add\_program(pio, &ws2812\_program);

    ws2812\_program\_init(pio, sm, offset, WS2812\_PIN, 800000, IS\_RGBW);

    int t = 0;

    while (1) {

         int pat = rand() % count\_of(pattern\_table); //random number div by

         int dir = (rand() >> 30) & 1 ? 1 : -1; //length of pattern table

         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;

        }

     }

    while (true) {

        printf("Hello, world!\n"); //Toggles between printing Hello

        put\_pixel(urgb\_u32(0xff, 0, 0)); //world and blinking the LED

        sleep\_ms(1000);

        put\_pixel(0);

        sleep\_ms(1000);

    }

    return 0;

}

// -------------------------------------------------- //

// 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, uint sm, uint offset, uint pin, float freq, bool rgbw) {

    pio\_gpio\_init(pio, pin); //WS2812 PIN is set to GPIO

    pio\_sm\_set\_consecutive\_pindirs(pio, sm, pin, 1, true); //direction as op

    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

// --------------- //

// 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 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, 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

s