```
// EasyREDVIO_ThingPlus.h
// jbrake@hmc.edu
// 02/06/2020

// This library provides an Arduino-style interface to control I/O
// devices on a RISC-V FE310 SoC on a SparkFun RED-V board.

#include <stdint.h>

////////////////////////////////////////////////////////////////////////////////
// Constant Definitions
////////////////////////////////////////////////////////////////////////////////

#define GPIO0_BASE  (0x10012000U)   // GPIO memory-mapped base address

#define GPIO0 ((GPIO*) GPIO0_BASE)  // Set up pointer to struct of type GPIO aligned at the base
GPIO0 memory-mapped address

#define LOW 0
#define HIGH 1

#define INPUT 0
#define OUTPUT 1
#define GPIO_IOF0 2

////////////////////////////////////////////////////////////////////////////////
// Bitfield Structs
////////////////////////////////////////////////////////////////////////////////

typedef struct
{
    volatile uint32_t   input_val;      // (GPIO offset 0x00) Pin value
    volatile uint32_t   input_en;       // (GPIO offset 0x04) Pin input enable*
    volatile uint32_t   output_en;      // (GPIO offset 0x08) Pin output enable*
    volatile uint32_t   output_val;     // (GPIO offset 0x0C) Output value
    volatile uint32_t   pue;            // (GPIO offset 0x10) Internal pull-up enable*
    volatile uint32_t   ds;             // (GPIO offset 0x14) Pin drive strength
    volatile uint32_t   rise_ie;        // (GPIO offset 0x18) Rise interrupt enable
    volatile uint32_t   rise_ip;        // (GPIO offset 0x1C) Rise interrupt pending
    volatile uint32_t   fall_ie;        // (GPIO offset 0x20) Fall interrupt enable
    volatile uint32_t   fall_ip;        // (GPIO offset 0x24) Fall interrupt pending
    volatile uint32_t   high_ie;        // (GPIO offset 0x28) High interrupt enable
    volatile uint32_t   high_ip;        // (GPIO offset 0x2C) High interrupt pending
    volatile uint32_t   low_ie;         // (GPIO offset 0x30) Low interrupt enable
    volatile uint32_t   low_ip;         // (GPIO offset 0x34) Low interrupt pending
    volatile uint32_t   iof_en;         // (GPIO offset 0x38) HW-Driven functions enable
    volatile uint32_t   iof_sel;        // (GPIO offset 0x3C) HW-Driven functions selection
    volatile uint32_t   out_xor;        // (GPIO offset 0x40) Output XOR (invert)
    // Registers marked with * are asynchronously reset to 0. All others are synchronously reset to
0.
} GPIO;

// Delay constants
#define COUNTS_PER_MS 898

/////////////////////////////////////////////////////////////////////////////
// GPIO Functions
/////////////////////////////////////////////////////////////////////////////

void pinMode(int pin, int function)
{
    switch(function) {
        case INPUT:
            GPIO0->input_en     |= (1 << pin);   // Sets a pin as an input
            break;
        case OUTPUT:
            GPIO0->output_en    |= (1 << pin);   // Set pin as an output
            GPIO0->iof_en       &= ~(1 << pin);
            break;
        case GPIO_IOF0:
            GPIO0->iof_sel      &= ~(1 << pin);
            GPIO0->iof_en       |= (1 << pin);
    }
}

void digitalWrite(int pin, int val)
{
    if (val) GPIO0->output_val |= (1 << pin);
    else     GPIO0->output_val &= ~(1 << pin);
}

int digitalRead(int pin)
{
    return (GPIO0->input_val >> pin) & 0x1;
}

////////////////////////////////////////////////////////////////////
// Delay Functions
////////////////////////////////////////////////////////////////////

void delayLoop(int ms) {
        // declare counter volatile so it isn't optimized away
        // counts_per_ms empirically determined such that delayLoop(100) waits 100 ms
        volatile int i = COUNTS_PER_MS * ms;

        while (i--); // count down time
}
```