```c
#include "ad9959.h"

// ============================================================================
// calculate tuning words
// ============================================================================
double get_asf(double amp, uint8_t* buf) {
    uint32_t asf = round(amp * 1024);

    // validation
    if (asf > 1023) asf = 1023;
    if (asf < 1) asf = 1;

    buf[0] = 0x00;
    buf[1] = ((0x300 & asf) >> 8) | 0x10;
    buf[2] = 0xff & asf;

    return asf / 1023.0;
}
double get_ftw(ad9959_config* c, double freq, uint8_t* buf) {
    double sys_clk = c->ref_clk * c->pll_mult;
    uint32_t ftw = round(freq * 4294967296.l / sys_clk);

    // maybe add some validation here

    // flip order of bits (little endian -> big endian)
    uint8_t* bytes = (uint8_t*)&ftw;
    for (int i = 0; i < 4; i++) {
        buf[i] = bytes[3 - i];
    }

    // return the frequency that was ablt to be set
    return ftw * sys_clk / 4294967296.l;
}
double get_pow(double phase, uint8_t* buf) {
    uint32_t pow = round(phase / 360.0 * 16384.0);

    // make sure pow is within range?
    pow = pow % 16383;

    buf[0] = (0xff00 & pow) >> 8;
    buf[1] = 0xff & pow;

    return pow / 16383.0 * 360.0;
}

// ============================================================================
// Sending Tuning Words
// ============================================================================
void send_channel(uint8_t reg, uint8_t channel, uint8_t* buf, size_t len) {
    uint8_t csr[] = {0x00, 0x02 | (1u << (channel + 4))};
    spi_write_blocking(spi1, csr, 2);
    spi_write_blocking(spi1, &reg, 1);
    spi_write_blocking(spi1, buf, len);
}

void send(uint8_t reg, uint8_t* buf, size_t len) {
```

```
    spi_write_blocking(spi1, &reg, 1);
    spi_write_blocking(spi1, buf, len);
}

// ============================================================================
// Readback
// ============================================================================

void read_reg(uint8_t reg, size_t len, uint8_t* buf) {
    reg |= 0x80;
    spi_write_blocking(spi1, &reg, 1);
    spi_read_blocking(spi1, 0, buf, len);
}

void read_all() {
    spi_set_baudrate(spi1, 1 * MHZ);

    uint8_t resp[20];

    read_reg(0x00, 1, resp);
    printf(" CSR: %02x\n", resp[0]);

    read_reg(0x01, 3, resp);
    printf(" FR1: %02x %02x %02x\n", resp[0], resp[1], resp[2]);

    read_reg(0x02, 2, resp);
    printf(" FR2: %02x %02x\n", resp[0], resp[1]);

    for (int i = 0; i < 4; i++) {
        printf("CHANNEL %d:\n", i);

        uint8_t csr[] = {0x00, (1u << (i + 4)) | 0x02};

        spi_write_blocking(spi1, csr, 2);

        read_reg(0x03, 3, resp);
        printf(" CFR: %02x %02x %02x\n", resp[0], resp[1], resp[2]);

        read_reg(0x04, 4, resp);
        printf("CFTW: %02x %02x %02x %02x\n", resp[0], resp[1], resp[2], resp[3]);

        read_reg(0x05, 2, resp);
        printf("CPOW: %02x %02x\n", resp[0], resp[1]);

        read_reg(0x06, 3, resp);
        printf(" ACR: %02x %02x %02x\n", resp[0], resp[1], resp[2]);

        read_reg(0x07, 2, resp);
        printf("LSRR: %02x %02x\n", resp[0], resp[1]);

        read_reg(0x08, 4, resp);
        printf(" RDW: %02x %02x %02x %02x\n", resp[0], resp[1], resp[2], resp[3]);

        read_reg(0x09, 4, resp);
        printf(" FDW: %02x %02x %02x %02x\n", resp[0], resp[1], resp[2], resp[3]);
```

```
        read_reg(0x0a, 4, resp);
        printf(" CW1: %02x %02x %02x %02x\n", resp[0], resp[1], resp[2], resp[3]);
    }
    spi_set_baudrate(spi1, 100 * MHZ);
}

// ============================================================================
// Control
// ============================================================================
void set_pll_mult(ad9959_config* c, uint mult) {
    c->pll_mult = mult;

    uint8_t vco = 0;
    if (mult * c->ref_clk >= 255 * MHZ) {
        vco = 0x80;
    }

    uint8_t fr1[] = {0x01, vco | (mult << 2), 0x00, 0x00};
    spi_write_blocking(spi1, fr1, 4);

    for (int i = 0; i < 4; i++) {
        printf("%02x\n", fr1[i]);
    }
}

void set_ref_clk(ad9959_config* c, uint64_t freq) { c->ref_clk = freq; }

void single_step_mode() {
    uint8_t csr = 0xf2;
    send(0x00, &csr, 1);
    uint8_t cfr[3] = {0x00, 0x03, 0x04};
    send(0x03, cfr, 3);
}

void clear() {
    uint8_t clear[] = {0x00, 0xf2, 0x03, 0x00, 0x03, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00, 0x05,
                       0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x08, 0x00, 0x00,
                       0x00, 0x00, 0x09, 0x00, 0x00, 0x00, 0x00, 0x0a, 0x00, 0x00, 0x00, 0x00};

    spi_write_blocking(spi1, clear, sizeof clear);
}
```