Lab 3
CSE121
Xavier Thompson

ChatGPT Prompts

Prompt: write a simple c program to read accelerometer data from mreg1 on esp32c3 icm42670p chip

ChatGPT:

Reading accelerometer data from the ICM-42670-P chip on the ESP32-C3 involves initializing the I2C communication and reading data from the specific registers that store accelerometer data. The accelerometer data is typically stored in the XOUT_A, YOUT_A, and ZOUT_A registers. Here's a simple C program to read the accelerometer data from the XOUT_A register of the ICM-42670-P chip:

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_log.h"
#include "driver/i2c.h"

#define ICM42670_I2C_ADDR 0x68
#define I2C_MASTER_SCL_IO 22            // GPIO pin for the I2C clock
#define I2C_MASTER_SDA_IO 21            // GPIO pin for the I2C data
#define I2C_MASTER_FREQ_HZ 100000   // I2C master clock frequency

static const char *TAG = "ICM42670";

void i2c_init() {
    i2c_config_t conf;
    conf.mode = I2C_MODE_MASTER;
    conf.sda_io_num = I2C_MASTER_SDA_IO;
    conf.sda_pullup_en = GPIO_PULLUP_ENABLE;
    conf.scl_io_num = I2C_MASTER_SCL_IO;
    conf.scl_pullup_en = GPIO_PULLUP_ENABLE;
    conf.master.clk_speed = I2C_MASTER_FREQ_HZ;
    i2c_param_config(I2C_NUM_0, &conf);
    i2c_driver_install(I2C_NUM_0, I2C_MODE_MASTER, 0, 0, 0);
}

int16_t read_accelerometer_data() {
	uint8_t data[6];  // 2 bytes for each of X, Y, and Z accelerometer data

	// Read accelerometer data from the ICM-42670-P
	i2c_cmd_handle_t cmd = i2c_cmd_link_create();
	i2c_master_start(cmd);
	i2c_master_write_byte(cmd, (ICM42670_I2C_ADDR << 1) | I2C_MASTER_WRITE,
1);
	i2c_master_write_byte(cmd, 0x3B, 1);  // Address of XOUT_A register
	i2c_master_start(cmd);  // Repeated start
	i2c_master_write_byte(cmd, (ICM42670_I2C_ADDR << 1) | I2C_MASTER_READ, 1);
	i2c_master_read(cmd, data, 6, I2C_MASTER_LAST_NACK);
	i2c_master_stop(cmd);
	i2c_master_cmd_begin(I2C_NUM_0, cmd, 1000 / portTICK_RATE_MS);
	i2c_cmd_link_delete(cmd);

	// Combine the high and low bytes for each axis
	int16_t x_acceleration = (int16_t)((data[0] << 8) | data[1]);

	return x_acceleration;
}

void app_main() {
	i2c_init();

	while (1) {
	int16_t x_acceleration = read_accelerometer_data();
	ESP_LOGI(TAG, "X-axis Acceleration: %d", x_acceleration);
	vTaskDelay(1000 / portTICK_PERIOD_MS); // Read data every 1 second
	}
}
```

This code initializes I2C communication, reads the X-axis accelerometer data from the XOUT_A register, and prints the X-axis acceleration value. You can adapt this code to read Y and Z-axis accelerometer data by changing the register addresses (e.g., XOUT_A, YOUT_A, ZOUT_A).

I basically used this exact code but modified the read_accelerometer_data function call in app_main to read the values from the gyroscope addresses because I was having issues with the accelerometer data.

https://stackoverflow.com/a/71833539: I also used this code snippet to convert the Two's complement number that the gyroscope returns to Decimal and called the function in read_accelerometer_data().