```
#include "Wire.h"
#define button 2
#define SLAVE_ADDR 0x0D
#define ADDR_PTR 0xB0

#define START_FREQ_R1 0x82
#define START_FREQ_R2 0x83
#define START_FREQ_R3 0x84

#define FREG_INCRE_R1 0x85
#define FREG_INCRE_R2 0x86
#define FREG_INCRE_R3 0x87

#define NUM_INCRE_R1 0x88
#define NUM_INCRE_R2 0x89

#define NUM_SCYCLES_R1 0x8A
#define NUM_SCYCLES_R2 0x8B

#define RE_DATA_R1 0x94
#define RE_DATA_R2 0x95

#define IMG_DATA_R1 0x96
#define IMG_DATA_R2 0x97

#define TEMP_R1 0x92
#define TEMP_R2 0x93

#define CTRL_REG 0x80
#define CTRL_REG2 0x81

#define STATUS_REG 0x8F


const float MCLK = 16.776*pow(10,6); // AD5933 Internal Clock Speed 16.776 MHz
const float start_freq = 1*pow(10,3); // Set start freq, < 100Khz
const float incre_freq = 1*pow(10,2); // Set freq increment
const int incre_num = 90; // Set number of increments; < 511

char state;

void setup() {
   Wire.begin();
   Serial.begin(115200);
   pinMode(button, INPUT);

   //nop - clear ctrl-reg
   writeData(CTRL_REG,0x0);

   //reset ctrl register
   writeData(CTRL_REG2,0x10);

   programReg();
}


void loop(){

   //Read state and enter FSM
```

```
    if(Serial.available()>0) {
      state = Serial.read();

      //FSM
      switch(state) {
        case 'A':  //Program Registers
          programReg();
          break;

        case 'B':  //Measure Temperature
          measureTemperature();
          break;

        case 'C':
          runSweep();
          delay(1000);
          break;

      /////Programming Device Registers/////


      }

      Serial.flush();
    }

}




void programReg(){

  // Set Range 1, PGA gain 1
  writeData(CTRL_REG,0x01);

  // Set settling cycles
  writeData(NUM_SCYCLES_R1, 0x07);
  writeData(NUM_SCYCLES_R2, 0xFF);

  // Start frequency of 1kHz
  writeData(START_FREQ_R1, getFrequency(start_freq,1));
  writeData(START_FREQ_R2, getFrequency(start_freq,2));
  writeData(START_FREQ_R3, getFrequency(start_freq,3));

  // Increment by 1 kHz
  writeData(FREG_INCRE_R1, getFrequency(incre_freq,1));
  writeData(FREG_INCRE_R2, getFrequency(incre_freq,2));
  writeData(FREG_INCRE_R3, getFrequency(incre_freq,3));

  // Points in frequency sweep (100), max 511
  writeData(NUM_INCRE_R1, (incre_num & 0x001F00)>>0x08 );
  writeData(NUM_INCRE_R2, (incre_num & 0x0000FF));

}


void runSweep() {
```

```
short re;
short img;
double freq;
double mag;
double phase;
double gain;
double impedance;
int i=0;

programReg();

// 1. Standby '10110000' Mask D8-10 of avoid tampering with gains
writeData(CTRL_REG,(readData(CTRL_REG) & 0x07) | 0xB0);

// 2. Initialize sweep
writeData(CTRL_REG,(readData(CTRL_REG) & 0x07) | 0x10);

// 3. Start sweep
writeData(CTRL_REG,(readData(CTRL_REG) & 0x07) | 0x20);


while((readData(STATUS_REG) & 0x07) < 4 ) {  // Check that status reg != 4, sweep not complete
   delay(100); // delay between measurements

   int flag = readData(STATUS_REG)& 2;


   if (flag==2) {

      byte R1 = readData(RE_DATA_R1);
      byte R2 = readData(RE_DATA_R2);
      re = (R1 << 8) | R2;

      R1  = readData(IMG_DATA_R1);
      R2  = readData(IMG_DATA_R2);
      img = (R1 << 8) | R2;

      freq = start_freq + i*incre_freq;
      mag = sqrt(pow(double(re),2)+pow(double(img),2));

      // phase = atan(double(img)/double(re));
      // phase = (180.0/3.1415926)*phase;  //convert phase angle to degrees

      // Phase Calibration
      // sys_phase = 118;
      // phase = phase - sys_phase;

      // gain = (1.0/197760)/9786.98;
//       impedance = 1/(gain*mag);

      Serial.print("Frequency: ");
      Serial.print(freq/1000);
      Serial.print(",kHz;");

      Serial.print(" Magnitude: ");
      Serial.print(mag);
      Serial.print(",kOhm;");

      Serial.print(" Resistance: ");
```

```
            Serial.print(re);
            Serial.print(",");

            Serial.print(" Reactance: ");
            Serial.print(img);
            Serial.println(",");

            // break;  //TODO: for single run, remove after debugging

            //Increment frequency
            if((readData(STATUS_REG) & 0x07) < 4 ){
               writeData(CTRL_REG,(readData(CTRL_REG) & 0x07) | 0x30);
               i++;
            }

         }
      }


      //Power down
//      writeData(CTRL_REG,0xA0);
      writeData(CTRL_REG,(readData(CTRL_REG) & 0x07) | 0xA0);
}



void writeData(int addr, int data) {

 Wire.beginTransmission(SLAVE_ADDR);
 Wire.write(addr);
 Wire.write(data);
 Wire.endTransmission();
 delay(1);
}


int readData(int addr){
   int data;

   Wire.beginTransmission(SLAVE_ADDR);
   Wire.write(ADDR_PTR);
   Wire.write(addr);
   Wire.endTransmission();

   delay(1);

   Wire.requestFrom(SLAVE_ADDR,1);

   if (Wire.available() >= 1){
      data = Wire.read();
   }
   else {
      data = -1;
   }

   delay(1);
   return data;
}
```

```
boolean measureTemperature() {

    // Measure temperature '10010000'
    writeData(CTRL_REG, 0x90);
    //TODO: necessary to write to second control register?

    delay(10); // wait for 10 ms



    //Check status reg for temp measurement available
    int flag = readData(STATUS_REG)& 1;

  if (flag == 1) {


    // Temperature is available
    int temperatureData = readData(TEMP_R1) << 8;
    temperatureData |= readData(TEMP_R2);
    temperatureData &= 0x3FFF; // remove first two bits

    if (temperatureData & 0x2000 == 1) { // negative temperature

      temperatureData -= 0x4000;
    }

    double val = double(temperatureData) / 32;
    temperatureData /= 32;

    Serial.print("Temperature: ");
    Serial.print(val);
    //Serial.write(176);  //degree sign
    Serial.println("C.");


    // Power Down '10100000'
    writeData(CTRL_REG,0xA0);


    return true;

  } else {
    return false;
  }
}


byte getFrequency(float freq, int n){
    long val = long((freq/(MCLK/4)) * pow(2,27));
    byte code;

      switch (n) {
        case 1:
          code = (val & 0xFF0000) >> 0x10;
          break;
```

```
        case 2:
          code = (val & 0x00FF00) >> 0x08;
          break;

        case 3:
          code = (val & 0x0000FF);
          break;

        default:
          code = 0;
      }

    return code;
}
```