```
    gas_sensor.power_mode = BME680_FORCED_MODE;

    //configure all the temperature, pressure, humidity and gas settings
    set_required_settings = BME680_OST_SEL | BME680_OSP_SEL | BME680_OSH_SEL | BME680_FILTER_SEL |
      BME680_GAS_SENSOR_SEL;
    rslt = bme680_set_sensor_settings(set_required_settings, &gas_sensor);
    rslt = bme680_set_sensor_mode(&gas_sensor);

    bme680_get_profile_dur(&meas_period, &gas_sensor);
    struct bme680_field_data data;      //create instance of bme680_field_data name 'data'

    while (1) {
        user_delay_ms(meas_period);         //delay for meas_period ms
        rslt = bme680_get_sensor_data(&data, &gas_sensor);      //read sensor measurements and store in data
        init_spi_lcd();                     //initialize spi for lcd before transactions

        Last_KeyPress = KeyPress;           //set last key press equal to current key press
        KeyPress = REG_PORT_IN0 & 0x04;     //mask pushbutton value onto current key press
        if (Last_KeyPress != KeyPress) {    //if last and current key value no equal
            if (KeyPress == 0x04) {         //and if key is not held down, increment counter
                count++;
            }
        }
        if (count % 2 == 0) {       //if count is even display temperature and pressure
            sprintf(dsp_buff_1, "T: %.2f degC", data.temperature / 100.0f);
            sprintf(dsp_buff_2, "P: %.2f hPa  ", data.pressure / 100.0f);
            update_lcd_dog();
        }
        else {                      //if count is odd display humidity and gas resistance
            sprintf(dsp_buff_1, "H: %.2f %%rH", data.humidity / 1000.0f);
            if (data.status & BME680_GASM_VALID_MSK) {
                sprintf(dsp_buff_2, "G: %ld ohms     ", data.gas_resistance);
            }
            update_lcd_dog();
        }
        //print values to TeraTerm
        printf("T: %.2f degC,  P: %.2f hPa,  H: %.2f %%rH", data.temperature / 100.0f, data.pressure / 100.0f,
```