```
        + ((float)dev->calib.par_p2 * var1)) / 524288.0f);
    var1 = ((1.0f + (var1 / 32768.0f)) * ((float)dev->calib.par_p1));
    calc_pres = (1048576.0f - ((float)pres_adc));

    /* Avoid exception caused by division by zero */
    if ((int)var1 != 0) {
        calc_pres = (((calc_pres - (var2 / 4096.0f)) * 6250.0f) / var1);
        var1 = (((float)dev->calib.par_p9) * calc_pres * calc_pres) / 2147483648.0f;
        var2 = calc_pres * (((float)dev->calib.par_p8) / 32768.0f);
        var3 = ((calc_pres / 256.0f) * (calc_pres / 256.0f) * (calc_pres / 256.0f)
            * (dev->calib.par_p10 / 131072.0f));
        calc_pres = (calc_pres + (var1 + var2 + var3 + ((float)dev->calib.par_p7 * 128.0f)) / 16.0f);
    } else {
        calc_pres = 0;
    }

    return calc_pres;
}

/*!
 * @brief This internal API is used to calculate the
 * humidity value in float format
 */
static float calc_humidity(uint16_t hum_adc, const struct bme680_dev *dev)
{
    float calc_hum = 0;
    float var1 = 0;
    float var2 = 0;
    float var3 = 0;
    float var4 = 0;
    float temp_comp;

    /* compensated temperature data*/
    temp_comp  = ((dev->calib.t_fine) / 5120.0f);

    var1 = (float)((float)hum_adc) - (((float)dev->calib.par_h1 * 16.0f) + (((float)dev->calib.par_h3 / 2.0f)
        * temp_comp));
```