```
struct BiquadCoeffecients  {  float b0, b1, b2, a1, a2; };
std::atomic<BiquadCoeffecients> coeffs;

BiquadCoeffecients calculateLowPassCoeffecients (float freq);

void audioThread (const float* src, float* dst, size_t n)
{
    static float lv1, lv2;
    auto local_coeffs = coeffs.load();

    for (size_t i = 0; i < n; ++i)
    {
        auto input = src[i];
        auto output = (input * local_coeffs.b0) + lv1;
        dst[i] = output;

        lv1 = (input * local_coeffs.b1) - (output* local_coeffs.a1) + lv2;
        lv2 = (input * local_coeffs.b2) - (output* local_coeffs.a2);
    }
}

void updateFrequencyParameter (float newValue)
{
    coeffs = calculateLowPassCoeffecients (newValue);
}
```

```
struct BiquadCoeffecients  {  float b0, b1, b2, a1, a2; };
std::atomic<BiquadCoeffecients> coeffs;
static_assert (std::atomic<BiquadCoefficients>::is_always_lock_free);

BiquadCoeffecients calculateLowPassCoeffecients (float freq);

void audioThread (const float* src, float* dst, size_t n)
{
    static float lv1, lv2;
    auto local_coeffs = coeffs.load();

    for (size_t i = 0; i < n; ++i)
    {
        auto input = src[i];
        auto output = (input * local_coeffs.b0) + lv1;
        dst[i] = output;

        lv1 = (input * local_coeffs.b1) - (output* local_coeffs.a1) + lv2;
        lv2 = (input * local_coeffs.b2) - (output* local_coeffs.a2);
    }
}

void updateFrequencyParameter (float newValue)
{
    coeffs = calculateLowPassCoeffecients (newValue);
}
```