```cpp
class SleepyDetector {
    static constexpr double TimerLengthMs = 20.;
    static constexpr double Eps = 1e-5;

    class Mono {
        struct Sample {
            Sample()
                : val(0.) { }

            void prepare() noexcept { val = 0.; }

            bool operator()(double x) noexcept {
                auto const xAbs = std::abs(x);
                auto const dist = val - xAbs;
                bool e = dist > Eps;
                val = xAbs;
                return e;
            }

            double val;
        };

    public:
        Mono()
            : sample()
            , timerIndex(0)
            , timerLength(0)
            , ringing(false) { }

        void prepare(int _timerLength) noexcept {
            sample.prepare();
            timerLength = _timerLength;
            timerIndex = 0;
            ringing = false;
        };

        void triggerNoteOn() noexcept {
            timerIndex = 0;
            ringing = true;
        }

        void operator()(double* smpls, int start, int end) noexcept {
            if (!ringing)
                return;
            for (auto s = start; s < end; ++s) {
                auto const y = smpls[s];
                if (sample(y)) {
                    timerIndex = 0;
                    return;
                }
            }
            timerIndex += end - start;
            if (timerIndex < timerLength)
                return;
            timerIndex = 0;
            ringing = false;
        }

        void operator()(double* smpls, int numSamples) noexcept {
            operator()(smpls, 0, numSamples);
        }

        Sample sample;
        int timerIndex, timerLength;
        bool ringing;
    };

public:
    SleepyDetector()
        : detectors()
        , noteOn(false) { }

    void prepare(double sampleRate) noexcept {
        auto const timerLength = static_cast<int>(math::msToSamples(TimerLengthMs, sampleRate));
        for (auto& d : detectors)
            d.prepare(timerLength);
        noteOn = false;
    };

    void panic(String& log) {
        log += "\nRinging: " + String(isRinging() ? "1" : "0");
        log += "\nNoteOn: " + String(noteOn ? "1" : "0");
    }

    void triggerNoteOn() noexcept {
        noteOn = true;
        for (auto& d : detectors)
            d.triggerNoteOn();
    }

    void triggerNoteOff() noexcept { noteOn = false; }

    void operator()(double** samples, int numChannels, int start, int end) noexcept {
        if (noteOn)
            return;
        for (auto ch = 0; ch < numChannels; ++ch) {
            auto const smpls = samples[ch];
            auto& detector = detectors[ch];
            detector(smpls, start, end);
        }
    }

    void operator()(double** samples, int numChannels, int numSamples) noexcept {
        operator()(samples, numChannels, 0, numSamples);
    }

    bool const isRinging() const noexcept {
        for (auto const& d : detectors)
            if (d.ringing)
                return true;
        return false;
    }

    bool const isSleepy() const noexcept { return !isRinging(); }

    bool const isNoteOn() const noexcept { return noteOn; }

private:
    std::array<Mono, 2> detectors {};
    bool noteOn;
};
```