FRICTION CHEAT SHEET

```
    // FREQUENCY AND DETUNE FUNCTIONS
    void setFrequency(float frequency);   // 0.0 - whatever, not sure what happens when negative :)
    void setFrequency1(float frequency1);
    void setFrequency2(float frequency2);
    void setFrequency3(float frequency3);

    void setSemitone1(int8_t semi);   // -24 to 24
    void setSemitone2(int8_t semi);
    void setSemitone3(int8_t semi);

    void setDetune(float detune);     // 0.0 to whatever.
    void setDetune1(float detune);    // 1.0 is one octave.
    void setDetune2(float detune);    // best results with detune between 0.00 and 0.02
    void setDetune3(float detune);

    void setOsc1LFO(bool lfo);    // true or false
    void setOsc2LFO(bool lfo);
    void setOsc3LFO(bool lfo);

    void setFM1(uint8_t fm); // 0 - 127
    void setFM2(uint8_t fm);
    void setFM3(uint8_t fm);

    void setFMOctaves(uint8_t octs);  // 1 - 128
    void setFM1Octaves(uint8_t octs);
    void setFM2Octaves(uint8_t octs);
    void setFM3Octaves(uint8_t octs);

    void setFM1Source(uint8_t source);    // 0 - 3 where
    void setFM2Source(uint8_t source);    // 0 is linear, 1 is Osc1
    void setFM3Source(uint8_t source);    // 2 is Osc2 and 3 is Osc3

    void setFM1Shape(uint8_t shape);  // 0 - 2 where
    void setFM2Shape(uint8_t shape);  // 0 is linear full signal
    void setFM3Shape(uint8_t shape);  // 1 is envelope1, 2 is envelope2

    void fmToZeroHertz(bool);     // true or false, if set to true the FM sounds more harmonic

    void setPortamento(int32_t port); // 0 - 127

    void set12bit(bool)      // true or false



    // WAVEFORM FUNCTIONS
    void setWaveform(uint16_t waveForm);    // JUST FOR 8bit WAVEFORMS
    void setWaveform1(uint16_t waveForm);   // 0 - 15
    void setWaveform2(uint16_t waveForm);   //
    void setWaveform3(uint16_t waveForm);   //

    // SHORTNAMES FOR WAVEFORMS
    SINE 0
    SQUARE 1
    PULSE 2
    TRIANGLE 3
    SAW 4
    FUZZ 5
    DIGI1 6
    DIGI2 7
    DIGI3 8
    DIGI4 9
    NOISE 10
    DIGI6 11
    TAN1 12
    TAN2 13
    TAN3 14
    TAN4 15




    // GAIN FUNCTIONS
    void setGain(float value); // 0.0 - 1.0
    void setGain1(float value); // 0.0 - 1.0
    void setGain2(float value); // 0.0 - 1.0
    void setGain3(float value); // 0.0 - 1.0

    float getGain();       // 0.0 - 1.0
    float getGain1();       // 0.0 - 1.0
    float getGain2();       // 0.0 - 1.0
    float getGain3();       // 0.0 - 1.0



    // NOTE FUNCTIONS
    void noteOn(uint8_t note, uint8_t vel); // 0 - 127
    void noteOn(uint8_t note); // 0 - 127

    void noteOff(uint8_t note); // 0 - 127
    void noteOff();

    float getNoteFrequency(uint8_t note); // 0 - 127
```

```
    // ENVELOPE FUNCTIONS
    void enableEnvelope1();
    void disableEnvelope1();

    void setEnv1Attack(uint8_t att); // 0 - 127
    void setEnv1Decay(uint8_t dec); // 0 - 127
    void setEnv1Sustain(uint8_t sus); // 0 - 127
    void setEnv1Release(uint8_t rel); // 0 - 127

    void enableEnvelope2();
    void disableEnvelope2();

    void setEnv2Attack(uint8_t att); // 0 - 127
    void setEnv2Decay(uint8_t dec); // 0 - 127
    void setEnv2Sustain(uint8_t sus); // 0 - 127
    void setEnv2Release(uint8_t rel); // 0 - 127



//synth parameters as MIDI controller numbers
    IS_12_BIT 3
    CUTOFF 4        // not currently used
    ZERO_HZ_FM 5
    FM_OCTAVES 6
    AMP_ENV 7
    PORTAMENTO 8

    FREQUENCY1 10
    SEMITONE1 11
    DETUNE1 12
    GAIN1 13
    WAVEFORM1 14
    FM1 15
    FM1_OCTAVES 16
    FM1_SOURCE 17
    FM1_SHAPE 18
    LFO1 19

    FREQUENCY2 20
    SEMITONE2 21
    DETUNE2 22
    GAIN2 23
    WAVEFORM2 24
    FM2 25
    FM2_OCTAVES 26
    FM2_SOURCE 27
    FM2_SHAPE 28
    LFO2 29

    FREQUENCY3 30
    SEMITONE3 31
    DETUNE3 32
    GAIN3 33
    WAVEFORM3 34
    FM3 35
    FM3_OCTAVES 36
    FM3_SOURCE 37
    FM3_SHAPE 38
    LFO3 39

    ENV1_ENABLE 113
    ENV1_ATTACK 114
    ENV1_DECAY 115
    ENV1_SUSTAIN 116
    ENV1_RELEASE 117

    ENV2_ENABLE 123
    ENV2_ATTACK 124
    ENV2_DECAY 125
    ENV2_SUSTAIN 126
    ENV2_RELEASE 127
```