M

Generated by Doxygen 1.12.0

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 m_delay_buffer Struct Reference

`#include <m_eng_delay_buffer.h>`

Data Fields

- int valid
- int pos
- int index
- int n_buffers
- int buffer_array_size
- float ** buffers

3.1.1 Field Documentation

3.1.1.1 buffer_array_size

`int m_delay_buffer::buffer_array_size`

3.1.1.2 buffers

`float** m_delay_buffer::buffers`

3.1.1.3 index

`int m_delay_buffer::index`

3.1.1.4 n_buffers

`int m_delay_buffer::n_buffers`

3.1.1.5 pos

```
int m_delay_buffer::pos
```

3.1.1.6 valid

```
int m_delay_buffer::valid
```

The documentation for this struct was generated from the following file:

- m_eng_delay_buffer.h

3.2 m_eng_3_band_eq_str Struct Reference

```
#include <m_eng_equaliser.h>
```

Data Fields

- m_parameter low
- m_parameter mid
- m_parameter high
- float coefs [3]
- int control_mode
- m_lr_low_pass_filter_str filters [2]

3.2.1 Field Documentation

3.2.1.1 coefs

```
float m_eng_3_band_eq_str::coefs[3]
```

3.2.1.2 control_mode

```
int m_eng_3_band_eq_str::control_mode
```

3.2.1.3 filters

```
m_lr_low_pass_filter_str m_eng_3_band_eq_str::filters[2]
```

3.2.1.4 high

```
m_parameter m_eng_3_band_eq_str::high
```

3.2.1.5 low

```
m_parameter m_eng_3_band_eq_str::low
```

3.2.1.6 mid

```
m_parameter m_eng_3_band_eq_str::mid
```

The documentation for this struct was generated from the following file:

- m_eng_equaliser.h

3.3 m_eng_3_band_splitter_str Struct Reference

```
#include <m_eng_band_splitter.h>
```

Data Fields

- m_lr_low_pass_filter_str filters [3]

3.3.1 Field Documentation

3.3.1.1 filters

```
m_lr_low_pass_filter_str m_eng_3_band_splitter_str::filters[3]
```

The documentation for this struct was generated from the following file:

- m_eng_band_splitter.h

3.4 m_eng_adaptive_waveshaper_str Struct Reference

```
#include <m_eng_adaptive_waveshaper.h>
```

Data Fields

- m_parameter coefficient
- float(∗ shape)(float x)
- float local_amplitude

3.4.1 Field Documentation

3.4.1.1 coefficient

```
m_parameter m_eng_adaptive_waveshaper_str::coefficient
```

3.4.1.2 local_amplitude

```
float m_eng_adaptive_waveshaper_str::local_amplitude
```

3.4.1.3 shape

```
float(* m_eng_adaptive_waveshaper_str::shape) (float x)
```

The documentation for this struct was generated from the following file:

- m_eng_adaptive_waveshaper.h

3.5 m_eng_amplifier_str Struct Reference

```
#include <m_eng_buffer_mixer_amp.h>
```

Data Fields

- m_parameter gain
- float g
- m_setting mode
- int db

3.5.1 Field Documentation

3.5.1.1 db

```
int m_eng_amplifier_str::db
```

3.5.1.2 g

```
float m_eng_amplifier_str::g
```

3.5.1.3 gain

```
m_parameter m_eng_amplifier_str::gain
```

3.5.1.4 mode

```
m_setting m_eng_amplifier_str::mode
```

The documentation for this struct was generated from the following file:

- m_eng_buffer_mixer_amp.h

3.6 m_eng_band_pass_filter_str Struct Reference

```
#include <m_eng_pass_filter.h>
```

Data Fields

- m_parameter center
- m_parameter bandwidth
- float a0
- float a1
- float a2
- float a3
- float a4
- float x1
- float x2
- float y1
- float y2

3.6.1 Field Documentation

3.6.1.1 a0

```
float m_eng_band_pass_filter_str::a0
```

3.6.1.2 a1

```
float m_eng_band_pass_filter_str::a1
```

3.6.1.3 a2

```
float m_eng_band_pass_filter_str::a2
```

3.6.1.4 a3

```
float m_eng_band_pass_filter_str::a3
```

3.6.1.5 a4

```
float m_eng_band_pass_filter_str::a4
```

3.6.1.6 bandwidth

```
m_parameter m_eng_band_pass_filter_str::bandwidth
```

3.6.1.7 center

```
m_parameter m_eng_band_pass_filter_str::center
```

3.6.1.8 x1

```
float m_eng_band_pass_filter_str::x1
```

3.6.1.9 x2

```
float m_eng_band_pass_filter_str::x2
```

3.6.1.10 y1

```
float m_eng_band_pass_filter_str::y1
```

3.6.1.11 y2

```
float m_eng_band_pass_filter_str::y2
```

The documentation for this struct was generated from the following file:

- m_eng_pass_filter.h

3.7 m_eng_biquad_str Struct Reference

```
#include <m_eng_biquad.h>
```

Data Fields

- float a0
- float a1
- float a2
- float a3
- float a4
- float x1
- float x2
- float y1
- float y2
- m_setting type
- m_parameter cutoff
- m_parameter bandwidth
- m_parameter db_gain

3.7.1 Field Documentation

3.7.1.1 a0

```
float m_eng_biquad_str::a0
```

3.7.1.2 a1

```
float m_eng_biquad_str::a1
```

3.7.1.3 a2

```
float m_eng_biquad_str::a2
```

3.7.1.4 a3

```
float m_eng_biquad_str::a3
```

3.7.1.5 a4

```
float m_eng_biquad_str::a4
```

3.7.1.6 bandwidth

```
m_parameter m_eng_biquad_str::bandwidth
```

3.7.1.7 cutoff

```
m_parameter m_eng_biquad_str::cutoff
```

3.7.1.8 db_gain

```
m_parameter m_eng_biquad_str::db_gain
```

3.7.1.9 type

```
m_setting m_eng_biquad_str::type
```

3.7.1.10 x1

```
float m_eng_biquad_str::x1
```

3.7.1.11 x2

```
float m_eng_biquad_str::x2
```

3.7.1.12 y1

```
float m_eng_biquad_str::y1
```

3.7.1.13 y2

```
float m_eng_biquad_str::y2
```

The documentation for this struct was generated from the following file:

- m_eng_biquad.h

3.8 m_eng_compressor_str Struct Reference

```
#include <m_eng_compressor.h>
```

Data Fields

- m_parameter ratio
- m_parameter threshold
- m_parameter attack
- m_parameter release
- float alpha
- float rho
- float e_final

3.8.1 Field Documentation

3.8.1.1 alpha

```
float m_eng_compressor_str::alpha
```

3.8.1.2 attack

```
m_parameter m_eng_compressor_str::attack
```

3.8.1.3 e_final

```
float m_eng_compressor_str::e_final
```

3.8.1.4 ratio

```
m_parameter m_eng_compressor_str::ratio
```

3.8.1.5 release

```
m_parameter m_eng_compressor_str::release
```

3.8.1.6 rho

```
float m_eng_compressor_str::rho
```

3.8.1.7 threshold

```
m_parameter m_eng_compressor_str::threshold
```

The documentation for this struct was generated from the following file:

- m_eng_compressor.h

3.9 m_eng_context Struct Reference

```
#include <m_eng_context.h>
```

Data Fields

- uint16_t status_flags
- int n_profiles
- int profile_array_size
- m_eng_profile * profiles
- int active_profile
- int profile_switch_triggered
- int new_profile
- int profiles_switching
- int profile_switch_progress
- int profile_switch_samples
- int profile_switch_type
- float declick_buffer [DECLICK_BUFSIZE]
- int profile_maintainance_index
- float prev_block [AUDIO_BLOCK_SAMPLES]
- int runs
- m_eng_high_pass_filter_str output_hpf
- m_eng_low_pass_filter_str input_lpf
- m_transformer output_amp
- float dc_blocker_avg

3.9.1 Field Documentation

3.9.1.1 active_profile

```
int m_eng_context::active_profile
```

3.9.1.2 dc_blocker_avg

```
float m_eng_context::dc_blocker_avg
```

3.9.1.3 declick_buffer

```
float m_eng_context::declick_buffer[DECLICK_BUFSIZE]
```

3.9.1.4 input_lpf

```
m_eng_low_pass_filter_str m_eng_context::input_lpf
```

3.9.1.5 n_profiles

```
int m_eng_context::n_profiles
```

3.9.1.6 new_profile

```
int m_eng_context::new_profile
```

3.9.1.7 output_amp

```
m_transformer m_eng_context::output_amp
```

3.9.1.8 output_hpf

```
m_eng_high_pass_filter_str m_eng_context::output_hpf
```

3.9.1.9 prev_block

```
float m_eng_context::prev_block[AUDIO_BLOCK_SAMPLES]
```

3.9.1.10 profile_array_size

```
int m_eng_context::profile_array_size
```

3.9.1.11 profile_maintainance_index

```
int m_eng_context::profile_maintainance_index
```

3.9.1.12 profile_switch_progress

```
int m_eng_context::profile_switch_progress
```

3.9.1.13 profile_switch_samples

```
int m_eng_context::profile_switch_samples
```

3.9.1.14 profile_switch_triggered

```
int m_eng_context::profile_switch_triggered
```

3.9.1.15 profile_switch_type

```
int m_eng_context::profile_switch_type
```

3.9.1.16 profiles

```
m_eng_profile* m_eng_context::profiles
```

3.9.1.17 profiles_switching

```
int m_eng_context::profiles_switching
```

3.9.1.18 runs

```
int m_eng_context::runs
```

3.9.1.19 status_flags

```
uint16_t m_eng_context::status_flags
```

The documentation for this struct was generated from the following file:

- m_eng_context.h

3.10 m_eng_delay_str Struct Reference

```
#include <m_eng_delay.h>
```

Data Fields

- m_setting tempo
- m_setting note
- m_parameter delay_gain
- float delay_samples
- float g
- m_delay_buffer buf

3.10.1 Field Documentation

3.10.1.1 buf

```
m_delay_buffer m_eng_delay_str::buf
```

3.10.1.2 delay_gain

```
m_parameter m_eng_delay_str::delay_gain
```

3.10.1.3 delay_samples

```
float m_eng_delay_str::delay_samples
```

3.10.1.4 g

```
float m_eng_delay_str::g
```

3.10.1.5 note

```
m_setting m_eng_delay_str::note
```

3.10.1.6 tempo

```
m_setting m_eng_delay_str::tempo
```

The documentation for this struct was generated from the following file:

- m_eng_delay.h

3.11 m_eng_dirty_octave_str Struct Reference

```
#include <m_eng_dirty_octave.h>
```

Data Fields

- m_parameter fuzz
- float dc_average
- float lpf_alpha
- float last_out_sample

3.11.1 Field Documentation

3.11.1.1 dc_average

```
float m_eng_dirty_octave_str::dc_average
```

3.11.1.2 fuzz

```
m_parameter m_eng_dirty_octave_str::fuzz
```

3.11.1.3 last_out_sample

```
float m_eng_dirty_octave_str::last_out_sample
```

3.11.1.4 lpf_alpha

```
float m_eng_dirty_octave_str::lpf_alpha
```

The documentation for this struct was generated from the following file:

- m_eng_dirty_octave.h

3.12 m_eng_distortion_str Struct Reference

```
#include <m_eng_distortion.h>
```

Data Fields

- m_setting function
- m_lr_low_pass_filter_str low_pass
- m_eng_waveshaper_str dist
- m_parameter wet_mix
- m_parameter bass_mix
- m_parameter bass_cutoff

3.12.1 Field Documentation

3.12.1.1 bass_cutoff

```
m_parameter m_eng_distortion_str::bass_cutoff
```

3.12.1.2 bass_mix

```
m_parameter m_eng_distortion_str::bass_mix
```

3.12.1.3 dist

```
m_eng_waveshaper_str m_eng_distortion_str::dist
```

3.12.1.4 function

```
m_setting m_eng_distortion_str::function
```

3.12.1.5 low_pass

```
m_lr_low_pass_filter_str m_eng_distortion_str::low_pass
```

3.12.1.6 wet_mix

```
m_parameter m_eng_distortion_str::wet_mix
```

The documentation for this struct was generated from the following file:

- m_eng_distortion.h

3.13 m_eng_envelope_str Struct Reference

```
#include <m_eng_envelope.h>
```

Data Fields

- m_parameter min_center
- m_parameter max_center
- m_parameter width
- m_parameter speed
- m_parameter sensitivity
- m_parameter smoothness
- float alpha
- float e
- int chunk_size
- m_eng_band_pass_filter_str filter

3.13.1 Field Documentation

3.13.1.1 alpha

```
float m_eng_envelope_str::alpha
```

3.13.1.2 chunk_size

```
int m_eng_envelope_str::chunk_size
```

3.13.1.3 e

```
float m_eng_envelope_str::e
```

3.13.1.4 filter

```
m_eng_band_pass_filter_str m_eng_envelope_str::filter
```

3.13.1.5 max_center

m_parameter m_eng_envelope_str::max_center

3.13.1.6 min_center

m_parameter m_eng_envelope_str::min_center

3.13.1.7 sensitivity

m_parameter m_eng_envelope_str::sensitivity

3.13.1.8 smoothness

m_parameter m_eng_envelope_str::smoothness

3.13.1.9 speed

m_parameter m_eng_envelope_str::speed

3.13.1.10 width

m_parameter m_eng_envelope_str::width

The documentation for this struct was generated from the following file:

- m_eng_envelope.h

3.14 m_eng_flanger_str Struct Reference

#include <m_eng_flanger.h>

Data Fields

- m_parameter range
- m_parameter tempo
- m_parameter depth
- m_parameter mix
- m_setting note
- float r
- float s
- float d
- float wet_mix
- float dry_mix
- float period
- float t
- m_delay_buffer buf

3.14.1 Field Documentation

3.14.1.1 buf

```
m_delay_buffer m_eng_flanger_str::buf
```

3.14.1.2 d

```
float m_eng_flanger_str::d
```

3.14.1.3 depth

```
m_parameter m_eng_flanger_str::depth
```

3.14.1.4 dry_mix

```
float m_eng_flanger_str::dry_mix
```

3.14.1.5 mix

```
m_parameter m_eng_flanger_str::mix
```

3.14.1.6 note

```
m_setting m_eng_flanger_str::note
```

3.14.1.7 period

```
float m_eng_flanger_str::period
```

3.14.1.8 r

```
float m_eng_flanger_str::r
```

3.14.1.9 range

```
m_parameter m_eng_flanger_str::range
```

3.14.1.10 s

```
float m_eng_flanger_str::s
```

3.14.1.11 t

```
float m_eng_flanger_str::t
```

3.14.1.12 tempo

```
m_parameter m_eng_flanger_str::tempo
```

3.14.1.13 wet_mix

```
float m_eng_flanger_str::wet_mix
```

The documentation for this struct was generated from the following file:

- m_eng_flanger.h

3.15 m_eng_high_pass_filter_str Struct Reference

```
#include <m_eng_pass_filter.h>
```

Data Fields

- m_parameter cutoff_frequency
- float a0
- float a1
- float a2
- float a3
- float a4
- float x1
- float x2
- float y1
- float y2

3.15.1 Field Documentation

3.15.1.1 a0

```
float m_eng_high_pass_filter_str::a0
```

3.15.1.2 a1

```
float m_eng_high_pass_filter_str::a1
```

3.15.1.3 a2

```
float m_eng_high_pass_filter_str::a2
```

3.15.1.4 a3

```
float m_eng_high_pass_filter_str::a3
```

3.15.1.5 a4

```
float m_eng_high_pass_filter_str::a4
```

3.15.1.6 cutoff_frequency

```
m_parameter m_eng_high_pass_filter_str::cutoff_frequency
```

3.15.1.7 x1

```
float m_eng_high_pass_filter_str::x1
```

3.15.1.8 x2

```
float m_eng_high_pass_filter_str::x2
```

3.15.1.9 y1

```
float m_eng_high_pass_filter_str::y1
```

3.15.1.10 y2

```
float m_eng_high_pass_filter_str::y2
```

The documentation for this struct was generated from the following file:

- m_eng_pass_filter.h

3.16 m_eng_log_entry Struct Reference

```
#include <m_eng_logging.h>
```

Data Fields

- int type
- const char * file_name
- const char * line
- const char * data_type
- const char * function
- const char * message
- uint64_t cycle
- uint32_t data
- uint32_t trace_depth

3.16.1 Field Documentation

3.16.1.1 cycle

```
uint64_t m_eng_log_entry::cycle
```

3.16.1.2 data

```
uint32_t m_eng_log_entry::data
```

3.16.1.3 data_type

```
const char* m_eng_log_entry::data_type
```

3.16.1.4 file_name

```
const char* m_eng_log_entry::file_name
```

3.16.1.5 function

```
const char* m_eng_log_entry::function
```

3.16.1.6 line

```
const char* m_eng_log_entry::line
```

3.16.1.7 message

```
const char* m_eng_log_entry::message
```

3.16.1.8 trace_depth

```
uint32_t m_eng_log_entry::trace_depth
```

3.16.1.9 type

```
int m_eng_log_entry::type
```

The documentation for this struct was generated from the following file:

- m_eng_logging.h

3.17 m_eng_low_end_compressor_str Struct Reference

```
#include <m_eng_low_end_compressor.h>
```

Data Fields

- m_eng_compressor_str bass_comp
- m_eng_compressor_str mids_comp
- m_lr_low_pass_filter_str low_pass
- m_lr_low_pass_filter_str mid_pass

3.17.1 Field Documentation

3.17.1.1 bass_comp

```
m_eng_compressor_str m_eng_low_end_compressor_str::bass_comp
```

3.17.1.2 low_pass

```
m_lr_low_pass_filter_str m_eng_low_end_compressor_str::low_pass
```

3.17.1.3 mid_pass

```
m_lr_low_pass_filter_str m_eng_low_end_compressor_str::mid_pass
```

3.17.1.4 mids_comp

```
m_eng_compressor_str m_eng_low_end_compressor_str::mids_comp
```

The documentation for this struct was generated from the following file:

- m_eng_low_end_compressor.h

3.18 m_eng_low_pass_filter_str Struct Reference

`#include <m_eng_pass_filter.h>`

Data Fields

- m_parameter cutoff_frequency
- float a0
- float a1
- float a2
- float a3
- float a4
- float x1
- float x2
- float y1
- float y2

3.18.1 Field Documentation

3.18.1.1 a0

`float m_eng_low_pass_filter_str::a0`

3.18.1.2 a1

`float m_eng_low_pass_filter_str::a1`

3.18.1.3 a2

`float m_eng_low_pass_filter_str::a2`

3.18.1.4 a3

`float m_eng_low_pass_filter_str::a3`

3.18.1.5 a4

`float m_eng_low_pass_filter_str::a4`

3.18.1.6 cutoff_frequency

`m_parameter m_eng_low_pass_filter_str::cutoff_frequency`

3.18.1.7 x1

```
float m_eng_low_pass_filter_str::x1
```

3.18.1.8 x2

```
float m_eng_low_pass_filter_str::x2
```

3.18.1.9 y1

```
float m_eng_low_pass_filter_str::y1
```

3.18.1.10 y2

```
float m_eng_low_pass_filter_str::y2
```

The documentation for this struct was generated from the following file:

- m_eng_pass_filter.h

3.19 m_eng_mixer_str Struct Reference

```
#include <m_eng_buffer_mixer_amp.h>
```

Data Fields

- m_parameter ratio

3.19.1 Field Documentation

3.19.1.1 ratio

```
m_parameter m_eng_mixer_str::ratio
```

The documentation for this struct was generated from the following file:

- m_eng_buffer_mixer_amp.h

3.20 m_eng_n_band_splitter_str Struct Reference

```
#include <m_eng_band_splitter.h>
```

Data Fields

- m_lr_low_pass_filter_str * filters

3.20.1 Field Documentation

3.20.1.1 filters

```
m_lr_low_pass_filter_str* m_eng_n_band_splitter_str::filters
```

The documentation for this struct was generated from the following file:

- m_eng_band_splitter.h

3.21 m_eng_noise_suppressor_str Struct Reference

```
#include <m_eng_noise_suppressor.h>
```

Data Fields

- m_parameter threshold
- m_parameter ratio
- m_parameter max_reduction
- float e_final
- int r

3.21.1 Field Documentation

3.21.1.1 e_final

```
float m_eng_noise_suppressor_str::e_final
```

3.21.1.2 max_reduction

```
m_parameter m_eng_noise_suppressor_str::max_reduction
```

3.21.1.3 r

```
int m_eng_noise_suppressor_str::r
```

3.21.1.4 ratio

```
m_parameter m_eng_noise_suppressor_str::ratio
```

3.21.1.5 threshold

```
m_parameter m_eng_noise_suppressor_str::threshold
```

The documentation for this struct was generated from the following file:

- m_eng_noise_suppressor.h

3.22 m_eng_percussifier_str Struct Reference

```
#include <m_eng_percussifier.h>
```

Data Fields

- m_parameter tempo
- m_parameter note
- m_parameter trigger_threshold
- m_parameter arm_threshold
- m_parameter refractory_period
- m_parameter fade_in
- m_parameter fade_out
- int state
- int fade_in_samples
- int hold_samples
- int refractory_samples
- int timer
- float decay_rate
- float rms_short
- float rms_long
- float alpha_short
- float alpha_long
- float gain
- float fade_alpha
- int r

3.22.1 Field Documentation

3.22.1.1 alpha_long

```
float m_eng_percussifier_str::alpha_long
```

3.22.1.2 alpha_short

```
float m_eng_percussifier_str::alpha_short
```

3.22.1.3 arm_threshold

m_parameter m_eng_percussifier_str::arm_threshold

3.22.1.4 decay_rate

float m_eng_percussifier_str::decay_rate

3.22.1.5 fade_alpha

float m_eng_percussifier_str::fade_alpha

3.22.1.6 fade_in

m_parameter m_eng_percussifier_str::fade_in

3.22.1.7 fade_in_samples

int m_eng_percussifier_str::fade_in_samples

3.22.1.8 fade_out

m_parameter m_eng_percussifier_str::fade_out

3.22.1.9 gain

float m_eng_percussifier_str::gain

3.22.1.10 hold_samples

int m_eng_percussifier_str::hold_samples

3.22.1.11 note

m_parameter m_eng_percussifier_str::note

3.22.1.12 r

int m_eng_percussifier_str::r

3.22.1.13 refractory_period

```
m_parameter m_eng_percussifier_str::refractory_period
```

3.22.1.14 refractory_samples

```
int m_eng_percussifier_str::refractory_samples
```

3.22.1.15 rms_long

```
float m_eng_percussifier_str::rms_long
```

3.22.1.16 rms_short

```
float m_eng_percussifier_str::rms_short
```

3.22.1.17 state

```
int m_eng_percussifier_str::state
```

3.22.1.18 tempo

```
m_parameter m_eng_percussifier_str::tempo
```

3.22.1.19 timer

```
int m_eng_percussifier_str::timer
```

3.22.1.20 trigger_threshold

```
m_parameter m_eng_percussifier_str::trigger_threshold
```

The documentation for this struct was generated from the following file:

- m_eng_percussifier.h

3.23 m_eng_profile Struct Reference

```
#include <m_eng_profile.h>
```

Data Fields

- m_pipeline * front_pipeline
- m_pipeline * back_pipeline
- int pipelines_swapping
- int pipeline_swap_progress
- int pipeline_swap_samples
- int pipeline_swap_type
- int back_pipeline_warmed_up
- m_pipeline_mod_ll * jobs
- m_pipeline_mod_ll * blocked_jobs
- int active
- int transition_policy
- float * prev_block
- m_transformer output_amp
- int runs

3.23.1 Field Documentation

3.23.1.1 active

```
int m_eng_profile::active
```

3.23.1.2 back_pipeline

```
m_pipeline* m_eng_profile::back_pipeline
```

3.23.1.3 back_pipeline_warmed_up

```
int m_eng_profile::back_pipeline_warmed_up
```

3.23.1.4 blocked_jobs

```
m_pipeline_mod_ll* m_eng_profile::blocked_jobs
```

3.23.1.5 front_pipeline

```
m_pipeline* m_eng_profile::front_pipeline
```

3.23.1.6 jobs

```
m_pipeline_mod_ll* m_eng_profile::jobs
```

3.23.1.7 output_amp

```
m_transformer m_eng_profile::output_amp
```

3.23.1.8 pipeline_swap_progress

```
int m_eng_profile::pipeline_swap_progress
```

3.23.1.9 pipeline_swap_samples

```
int m_eng_profile::pipeline_swap_samples
```

3.23.1.10 pipeline_swap_type

```
int m_eng_profile::pipeline_swap_type
```

3.23.1.11 pipelines_swapping

```
int m_eng_profile::pipelines_swapping
```

3.23.1.12 prev_block

```
float* m_eng_profile::prev_block
```

3.23.1.13 runs

```
int m_eng_profile::runs
```

3.23.1.14 transition_policy

```
int m_eng_profile::transition_policy
```

The documentation for this struct was generated from the following file:

- m_eng_profile.h

3.24 m_eng_profiler_entry Struct Reference

```
#include <m_eng_logging.h>
```

Data Fields

- const char * function_name
- uint64_t open_cycle
- uint64_t calls
- uint64_t total_cycles
- uint64_t peak_cycles
- double ra_cycles

3.24.1 Field Documentation

3.24.1.1 calls

```
uint64_t m_eng_profiler_entry::calls
```

3.24.1.2 function_name

```
const char* m_eng_profiler_entry::function_name
```

3.24.1.3 open_cycle

```
uint64_t m_eng_profiler_entry::open_cycle
```

3.24.1.4 peak_cycles

```
uint64_t m_eng_profiler_entry::peak_cycles
```

3.24.1.5 ra_cycles

```
double m_eng_profiler_entry::ra_cycles
```

3.24.1.6 total_cycles

```
uint64_t m_eng_profiler_entry::total_cycles
```

The documentation for this struct was generated from the following file:

- m_eng_logging.h

3.25 m_eng_simple_distortion_str Struct Reference

```
#include <m_eng_simple_distortion.h>
```

Data Fields

- m_parameter pregain
- m_parameter postgain

3.25.1 Field Documentation

3.25.1.1 postgain

```
m_parameter m_eng_simple_distortion_str::postgain
```

3.25.1.2 pregain

```
m_parameter m_eng_simple_distortion_str::pregain
```

The documentation for this struct was generated from the following file:

- m_eng_simple_distortion.h

3.26 m_eng_warbler_str Struct Reference

```
#include <m_eng_warbler.h>
```

Data Fields

- m_parameter center
- m_parameter width
- m_parameter reactivity
- m_parameter sensitivity
- m_parameter max_rate
- m_parameter min_rate
- float alpha
- float e
- float t
- float rate
- m_eng_band_pass_filter_str filter

3.26.1 Field Documentation

3.26.1.1 alpha

```
float m_eng_warbler_str::alpha
```

3.26.1.2 center

```
m_parameter m_eng_warbler_str::center
```

3.26.1.3 e

```
float m_eng_warbler_str::e
```

3.26.1.4 filter

```
m_eng_band_pass_filter_str m_eng_warbler_str::filter
```

3.26.1.5 max_rate

```
m_parameter m_eng_warbler_str::max_rate
```

3.26.1.6 min_rate

```
m_parameter m_eng_warbler_str::min_rate
```

3.26.1.7 rate

```
float m_eng_warbler_str::rate
```

3.26.1.8 reactivity

```
m_parameter m_eng_warbler_str::reactivity
```

3.26.1.9 sensitivity

```
m_parameter m_eng_warbler_str::sensitivity
```

3.26.1.10 t

```
float m_eng_warbler_str::t
```

3.26.1.11 width

```
m_parameter m_eng_warbler_str::width
```

The documentation for this struct was generated from the following file:

- m_eng_warbler.h

3.27 m_eng_waveshaper_str Struct Reference

`#include <m_eng_waveshaper.h>`

Data Fields

- m_parameter coefficient
- float(* shape)(float x)

3.27.1 Field Documentation

3.27.1.1 coefficient

```
m_parameter m_eng_waveshaper_str::coefficient
```

3.27.1.2 shape

```
float(* m_eng_waveshaper_str::shape) (float x)
```

The documentation for this struct was generated from the following file:

- m_eng_waveshaper.h

3.28 m_lr_high_pass_filter_str Struct Reference

`#include <m_eng_linkowitz_riley.h>`

Data Fields

- m_parameter cutoff_frequency
- float a0
- float a1
- float a2
- float a3
- float a4
- float x_11
- float x_12
- float y_11
- float y_12
- float x_21
- float x_22
- float y_21
- float y_22

3.28.1 Field Documentation

3.28.1.1 a0

```
float m_lr_high_pass_filter_str::a0
```

3.28.1.2 a1

```
float m_lr_high_pass_filter_str::a1
```

3.28.1.3 a2

```
float m_lr_high_pass_filter_str::a2
```

3.28.1.4 a3

```
float m_lr_high_pass_filter_str::a3
```

3.28.1.5 a4

```
float m_lr_high_pass_filter_str::a4
```

3.28.1.6 cutoff_frequency

```
m_parameter m_lr_high_pass_filter_str::cutoff_frequency
```

3.28.1.7 x_11

```
float m_lr_high_pass_filter_str::x_11
```

3.28.1.8 x_12

```
float m_lr_high_pass_filter_str::x_12
```

3.28.1.9 x_21

```
float m_lr_high_pass_filter_str::x_21
```

3.28.1.10 x_22

```
float m_lr_high_pass_filter_str::x_22
```

3.28.1.11 y_11

```
float m_lr_high_pass_filter_str::y_11
```

3.28.1.12 y_12

```
float m_lr_high_pass_filter_str::y_12
```

3.28.1.13 y_21

```
float m_lr_high_pass_filter_str::y_21
```

3.28.1.14 y_22

```
float m_lr_high_pass_filter_str::y_22
```

The documentation for this struct was generated from the following file:

- m_eng_linkowitz_riley.h

3.29 m_lr_low_pass_filter_str Struct Reference

```
#include <m_eng_linkowitz_riley.h>
```

Data Fields

- m_parameter cutoff_frequency
- float a0
- float a1
- float a2
- float a3
- float a4
- float x_11
- float x_12
- float y_11
- float y_12
- float x_21
- float x_22
- float y_21
- float y_22

3.29.1 Field Documentation

3.29.1.1 a0

```
float m_lr_low_pass_filter_str::a0
```

3.29.1.2 a1

```
float m_lr_low_pass_filter_str::a1
```

3.29.1.3 a2

```
float m_lr_low_pass_filter_str::a2
```

3.29.1.4 a3

```
float m_lr_low_pass_filter_str::a3
```

3.29.1.5 a4

```
float m_lr_low_pass_filter_str::a4
```

3.29.1.6 cutoff_frequency

```
m_parameter m_lr_low_pass_filter_str::cutoff_frequency
```

3.29.1.7 x_11

```
float m_lr_low_pass_filter_str::x_11
```

3.29.1.8 x_12

```
float m_lr_low_pass_filter_str::x_12
```

3.29.1.9 x_21

```
float m_lr_low_pass_filter_str::x_21
```

3.29.1.10 x_22

```
float m_lr_low_pass_filter_str::x_22
```

3.29.1.11 y_11

```
float m_lr_low_pass_filter_str::y_11
```

3.29.1.12 y_12

```
float m_lr_low_pass_filter_str::y_12
```

3.29.1.13 y_21

```
float m_lr_low_pass_filter_str::y_21
```

3.29.1.14 y_22

```
float m_lr_low_pass_filter_str::y_22
```

The documentation for this struct was generated from the following file:

- m_eng_linkowitz_riley.h

3.30 m_message Struct Reference

```
#include <m_comms.h>
```

Data Fields

- uint8_t type
- uint8_t data [M_RESPONSE_MAX_DATA_LEN]
- void(∗ callback)(struct m_message msg, m_response response)
- void ∗ cb_arg
- int retries

3.30.1 Field Documentation

3.30.1.1 callback

```
void(* m_message::callback) (struct m_message msg, m_response response)
```

3.30.1.2 cb_arg

```
void* m_message::cb_arg
```

3.30.1.3 data

```
uint8_t m_message::data[M_RESPONSE_MAX_DATA_LEN]
```

3.30.1.4 retries

```
int m_message::retries
```

3.30.1.5 type

```
uint8_t m_message::type
```

The documentation for this struct was generated from the following file:

- m_comms.h

3.31 m_parameter Struct Reference

```
#include <m_parameter.h>
```

Data Fields

- float value
- float min
- float max
- int scale
- int updated
- float old_value
- float new_value

3.31.1 Field Documentation

3.31.1.1 max

```
float m_parameter::max
```

3.31.1.2 min

```
float m_parameter::min
```

3.31.1.3 new_value

```
float m_parameter::new_value
```

3.31.1.4 old_value

```
float m_parameter::old_value
```

3.31.1.5 scale

```
int m_parameter::scale
```

3.31.1.6 updated

```
int m_parameter::updated
```

3.31.1.7 value

```
float m_parameter::value
```

The documentation for this struct was generated from the following file:

- m_parameter.h

3.32 m_parameter_id Struct Reference

```
#include <m_parameter.h>
```

Data Fields

- uint16_t profile_id
- uint16_t transformer_id
- uint16_t parameter_id

3.32.1 Field Documentation

3.32.1.1 parameter_id

```
uint16_t m_parameter_id::parameter_id
```

3.32.1.2 profile_id

```
uint16_t m_parameter_id::profile_id
```

3.32.1.3 transformer_id

```
uint16_t m_parameter_id::transformer_id
```

The documentation for this struct was generated from the following file:

- m_parameter.h

3.33 m_pipeline Struct Reference

`#include <m_pipeline.h>`

The documentation for this struct was generated from the following file:

- m_pipeline.h

3.34 m_pipeline_mod Struct Reference

`#include <m_eng_pipeline_mod.h>`

Data Fields

- int type
- uint16_t tid
- uint16_t data
- int16_t sdata

3.34.1 Field Documentation

3.34.1.1 data

`uint16_t m_pipeline_mod::data`

3.34.1.2 sdata

`int16_t m_pipeline_mod::sdata`

3.34.1.3 tid

`uint16_t m_pipeline_mod::tid`

3.34.1.4 type

`int m_pipeline_mod::type`

The documentation for this struct was generated from the following file:

- m_eng_pipeline_mod.h

3.35 m_profile Struct Reference

`#include <m_profile.h>`

Data Fields

- int active

3.35.1 Field Documentation

3.35.1.1 active

`int m_profile::active`

The documentation for this struct was generated from the following file:

- m_profile.h

3.36 m_response Struct Reference

`#include <m_comms.h>`

Data Fields

- uint8_t type
- uint8_t data [M_RESPONSE_MAX_DATA_LEN]
- void * extra

3.36.1 Field Documentation

3.36.1.1 data

`uint8_t m_response::data[M_RESPONSE_MAX_DATA_LEN]`

3.36.1.2 extra

`void* m_response::extra`

3.36.1.3 type

`uint8_t m_response::type`

The documentation for this struct was generated from the following file:

- m_comms.h

3.37 m_setting Struct Reference

`#include <m_parameter.h>`

Data Fields

- int16_t value
- int updated
- int16_t old_value
- int16_t new_value

3.37.1 Field Documentation

3.37.1.1 new_value

`int16_t m_setting::new_value`

3.37.1.2 old_value

`int16_t m_setting::old_value`

3.37.1.3 updated

`int m_setting::updated`

3.37.1.4 value

`int16_t m_setting::value`

The documentation for this struct was generated from the following file:

- m_parameter.h

3.38 m_setting_id Struct Reference

`#include <m_parameter.h>`

Data Fields

- uint16_t profile_id
- uint16_t transformer_id
- uint16_t setting_id

3.38.1 Field Documentation

3.38.1.1 profile_id

```
uint16_t m_setting_id::profile_id
```

3.38.1.2 setting_id

```
uint16_t m_setting_id::setting_id
```

3.38.1.3 transformer_id

```
uint16_t m_setting_id::transformer_id
```

The documentation for this struct was generated from the following file:

- m_parameter.h

3.39 m_transformer Struct Reference

```
#include <m_transformer.h>
```

Data Fields

- uint16_t type
- uint16_t id
- m_parameter wet_mix
- m_setting band_mode
- m_parameter band_lp_cutoff
- m_parameter band_hp_cutoff
- m_parameter band_center
- m_parameter band_width

3.39.1 Field Documentation

3.39.1.1 band_center

```
m_parameter m_transformer::band_center
```

3.39.1.2 band_hp_cutoff

```
m_parameter m_transformer::band_hp_cutoff
```

3.39.1.3 band_lp_cutoff

```
m_parameter m_transformer::band_lp_cutoff
```

3.39.1.4 band_mode

```
m_setting m_transformer::band_mode
```

3.39.1.5 band_width

```
m_parameter m_transformer::band_width
```

3.39.1.6 id

```
uint16_t m_transformer::id
```

3.39.1.7 type

```
uint16_t m_transformer::type
```

3.39.1.8 wet_mix

```
m_parameter m_transformer::wet_mix
```

The documentation for this struct was generated from the following file:

- m_transformer.h

3.40 m_transformer_str Struct Reference

```
#include <m_eng_transformer_template.h>
```

Data Fields

- m_parameter param

3.40.1 Field Documentation

3.40.1.1 param

```
m_parameter m_transformer_str::param
```

The documentation for this struct was generated from the following file:

- m_eng_transformer_template.h

Chapter 4

File Documentation

4.1 m_eng.h File Reference

```
#include <arm_math.h>
#include "utility/imxrt_hw.h"
#include <stdint.h>
#include <string.h>
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "m_transformer_enum.h"
#include "m_error_codes.h"
#include "m_status.h"
#include "m_comms.h"
#include "m_alloc.h"
#include "m_linked_list.h"
#include "m_parameter.h"
#include "m_transformer.h"
#include "m_pipeline.h"
#include "m_profile.h"
#include "m_eng_logging.h"
#include "m_eng_flops.h"
#include "m_eng_useful_functions.h"
#include "m_eng_audio_block.h"
#include "m_eng_mempool.h"
#include "m_eng_globals.h"
#include "m_eng_parameter.h"
#include "m_eng_transformer.h"
#include "m_eng_transformer_init.h"
#include "m_eng_delay_buffer.h"
#include "m_eng_buffer_mixer_amp.h"
#include "m_eng_linkowitz_riley.h"
#include "m_eng_equaliser.h"
#include "m_eng_pass_filter.h"
#include "m_eng_biquad.h"
#include "m_eng_compressor.h"
#include "m_eng_low_end_compressor.h"
#include "m_eng_waveshaper.h"
#include "m_eng_adaptive_waveshaper.h"
```

```
#include "m_eng_simple_distortion.h"
#include "m_eng_distortion.h"
#include "m_eng_dirty_octave.h"
#include "m_eng_noise_suppressor.h"
#include "m_eng_percussifier.h"
#include "m_eng_envelope.h"
#include "m_eng_flanger.h"
#include "m_eng_warbler.h"
#include "m_eng_delay.h"
#include "m_eng_pipeline.h"
#include "m_eng_pipeline_mod.h"
#include "m_eng_profile.h"
#include "m_eng_update.h"
#include "m_eng_i2s_dma.h"
#include "m_eng_sgtl5000.h"
#include "m_eng_memcpy_audio.h"
#include "m_eng_context.h"
#include "m_eng_comms.h"
#include "m_eng_transition.h"
#include "m_eng_printf.h"
#include "m_eng_debugging.h"
```

Macros

- #define AUDIO_BLOCK_SAMPLES 128
- #define AUDIO_SAMPLE_RATE_EXACT 44100.0f
- #define AUDIO_SAMPLE_RATE AUDIO_SAMPLE_RATE_EXACT
- #define SAMPLE_FREQUENCY 0.0000226757369615
- #define NUM_MASKS (((MAX_AUDIO_MEMORY / AUDIO_BLOCK_SAMPLES / 2) + 31) / 32)
- #define M_ENGINE
- #define LN_2 0.69314718055994530942
- #define S_TO_SAMPLES(x)
- #define SAMPLES_TO_S(x)
- #define MS_TO_SAMPLES(x)
- #define SAMPLES_TO_MS(x)
- #define AUDIO_BLOCK_MS (((float)AUDIO_BLOCK_SAMPLES * 1000.0) / (float)AUDIO_SAMPLE_RATE)
- #define ALLOW_PRINTLINES
- #define M_VOICE_ERR 0
- #define M_VOICE_COMMS 1
- #define M_VOICE_CXT 2
- #define M_VOICE_TR 3
- #define M_VOICE_PL 4
- #define M_VOICE_PR 5
- #define M_VOICE_LOG 6
- #define M_VOICE_PRF 7
- #define LL_MALLOC m_alloc
- #define LL_FREE m_free
- #define sqr(x)
- #define binary_max(x, y)
- #define binary_min(x, y)

Functions

- float trig_transition_function (float x)

4.1.1 Macro Definition Documentation

4.1.1.1 ALLOW_PRINTLINES

```
#define ALLOW_PRINTLINES
```

4.1.1.2 AUDIO_BLOCK_MS

```
#define AUDIO_BLOCK_MS (((float)AUDIO_BLOCK_SAMPLES * 1000.0) / (float)AUDIO_SAMPLE_RATE)
```

4.1.1.3 AUDIO_BLOCK_SAMPLES

```
#define AUDIO_BLOCK_SAMPLES 128
```

4.1.1.4 AUDIO_SAMPLE_RATE

```
#define AUDIO_SAMPLE_RATE AUDIO_SAMPLE_RATE_EXACT
```

4.1.1.5 AUDIO_SAMPLE_RATE_EXACT

```
#define AUDIO_SAMPLE_RATE_EXACT 44100.0f
```

4.1.1.6 binary_max

```
#define binary_max(
              x,
              y)
```

Value:

```
((x > y) ? x : y)
```

4.1.1.7 binary_min

```
#define binary_min(
              x,
              y)
```

Value:

```
((x > y) ? y : x)
```

4.1.1.8 LL_FREE

```
#define LL_FREE m_free
```

4.1.1.9 LL_MALLOC

```
#define LL_MALLOC m_alloc
```

4.1.1.10 LN_2

```
#define LN_2 0.69314718055994530942
```

4.1.1.11 M_ENGINE

```
#define M_ENGINE
```

4.1.1.12 M_VOICE_COMMS

```
#define M_VOICE_COMMS 1
```

4.1.1.13 M_VOICE_CXT

```
#define M_VOICE_CXT 2
```

4.1.1.14 M_VOICE_ERR

```
#define M_VOICE_ERR 0
```

4.1.1.15 M_VOICE_LOG

```
#define M_VOICE_LOG 6
```

4.1.1.16 M_VOICE_PL

```
#define M_VOICE_PL 4
```

4.1.1.17 M_VOICE_PR

```
#define M_VOICE_PR 5
```

4.1.1.18 M_VOICE_PRF

```
#define M_VOICE_PRF 7
```

4.1.1.19 M_VOICE_TR

```
#define M_VOICE_TR 3
```

4.1.1.20 MS_TO_SAMPLES

```
#define MS_TO_SAMPLES(
              x)
```

Value:
```
((int)(ceilf((((float)(x) / 1000.0) * AUDIO_SAMPLE_RATE))))
```

4.1.1.21 NUM_MASKS

```
#define NUM_MASKS (((MAX_AUDIO_MEMORY / AUDIO_BLOCK_SAMPLES / 2) + 31) / 32)
```

4.1.1.22 S_TO_SAMPLES

```
#define S_TO_SAMPLES(
              x)
```

Value:
```
((int)(ceilf((float)(x) * AUDIO_SAMPLE_RATE)))
```

4.1.1.23 SAMPLE_FREQUENCY

```
#define SAMPLE_FREQUENCY 0.0000226757369615
```

4.1.1.24 SAMPLES_TO_MS

```
#define SAMPLES_TO_MS(
              x)
```

Value:
```
(((float)(x)  * 1000.0) / (float)AUDIO_SAMPLE_RATE)
```

4.1.1.25 SAMPLES_TO_S

```
#define SAMPLES_TO_S(
              x)
```

Value:
```
((float)(x) / (float)AUDIO_SAMPLE_RATE)
```

4.1.1.26 sqr

```
#define sqr(
              x)
```

Value:
```
(x * x)
```

4.1.2 Function Documentation

4.1.2.1 trig_transition_function()

```
float trig_transition_function (
             float x)
```

4.2 m_eng.h

Go to the documentation of this file.

```
00001 #ifndef M_MAIN_H_
00002 #define M_MAIN_H_
00003
00004 #define AUDIO_BLOCK_SAMPLES             128
00005
00006 #ifndef ARM_ASSEMBLY
00007 #define AUDIO_SAMPLE_RATE_EXACT         44100.0f
00008
00009 #define AUDIO_SAMPLE_RATE               AUDIO_SAMPLE_RATE_EXACT
00010 #define SAMPLE_FREQUENCY                0.0000226757369615
00011
00012 #define NUM_MASKS                       (((MAX_AUDIO_MEMORY / AUDIO_BLOCK_SAMPLES / 2) + 31) / 32)
00013
00014 #ifndef M_ENGINE
00015 #define M_ENGINE
00016 #endif
00017
00018 //#define NO_CMSIS
00019
00020 #ifdef M_SIMULATED
00021 #ifndef NO_CMSIS
00022 #define NO_CMSIS
00023 #endif
00024 #endif
00025
00026 #ifdef M_SIMULATED
00027  #define DMAMEM
00028  #define FLASHMEM
00029  #define __enable_irq()
00030  #define __disable_irq()
00031  #define PI   3.14159265
00032 #else
00033  #include <arm_math.h>
00034  #include "utility/imxrt_hw.h"
00035 #endif
00036
00037 #ifdef __cplusplus
00038  #include <cstdint>
00039  #include <cstring>
00040  #include <cstdarg>
00041  #include <cstdlib>
00042  #include <cstdio>
00043  #include <cmath>
00044
00045  extern "C" {
00046 #else
00047  #include <stdint.h>
00048  #include <string.h>
00049  #include <stdarg.h>
00050  #include <stdlib.h>
00051  #include <stdio.h>
00052  #include <math.h>
00053 #endif
```

```
00054
00055 #define LN_2 0.69314718055994530942
00056
00057 #define S_TO_SAMPLES(x)     ((int)(ceilf((float)(x) * AUDIO_SAMPLE_RATE)))
00058 #define SAMPLES_TO_S(x)     ((float)(x) / (float)AUDIO_SAMPLE_RATE)
00059
00060 #define MS_TO_SAMPLES(x)    ((int)(ceilf((((float)(x) / 1000.0) * AUDIO_SAMPLE_RATE))))
00061 #define SAMPLES_TO_MS(x)    (((float)(x)  * 1000.0) / (float)AUDIO_SAMPLE_RATE)
00062
00063 #define AUDIO_BLOCK_MS      (((float)AUDIO_BLOCK_SAMPLES * 1000.0) / (float)AUDIO_SAMPLE_RATE)
00064
00065 #define ALLOW_PRINTLINES
00066 #define M_VOICE_ERR     0
00067 #define M_VOICE_COMMS   1
00068 #define M_VOICE_CXT     2
00069 #define M_VOICE_TR      3
00070 #define M_VOICE_PL      4
00071 #define M_VOICE_PR      5
00072 #define M_VOICE_LOG     6
00073 #define M_VOICE_PRF     7
00074
00075 //#define ENABLE_LOGGING
00076
00077 #include "m_transformer_enum.h"
00078 #include "m_error_codes.h"
00079 #include "m_status.h"
00080 #include "m_comms.h"
00081
00082 #include "m_alloc.h"
00083
00084 #define LL_MALLOC m_alloc
00085 #define LL_FREE   m_free
00086
00087 #include "m_linked_list.h"
00088
00089 #include "m_parameter.h"
00090 #include "m_transformer.h"
00091 #include "m_pipeline.h"
00092 #include "m_profile.h"
00093
00094 #include "m_eng_logging.h"
00095
00096 #include "m_eng_flops.h"
00097 #include "m_eng_useful_functions.h"
00098
00099 #include "m_eng_audio_block.h"
00100 #include "m_eng_mempool.h"
00101
00102 #include "m_eng_globals.h"
00103
00104 #include "m_eng_parameter.h"
00105 #include "m_eng_transformer.h"
00106 #include "m_eng_transformer_init.h"
00107
00108 #include "m_eng_delay_buffer.h"
00109
00110 #include "m_eng_buffer_mixer_amp.h"
00111 #include "m_eng_linkowitz_riley.h"
00112 #include "m_eng_equaliser.h"
00113 #include "m_eng_pass_filter.h"
00114 #include "m_eng_biquad.h"
00115 #include "m_eng_compressor.h"
00116 #include "m_eng_low_end_compressor.h"
00117 #include "m_eng_waveshaper.h"
00118 #include "m_eng_adaptive_waveshaper.h"
00119 #include "m_eng_simple_distortion.h"
00120 #include "m_eng_distortion.h"
00121 #include "m_eng_dirty_octave.h"
00122 #include "m_eng_noise_suppressor.h"
00123 #include "m_eng_percussifier.h"
00124 #include "m_eng_envelope.h"
00125 #include "m_eng_flanger.h"
00126 #include "m_eng_warbler.h"
00127 #include "m_eng_delay.h"
00128
00129 #ifdef GRAPH_PIPELINE
00130 #include "m_eng_graph.h"
00131 #endif
00132 #include "m_eng_pipeline.h"
00133 #include "m_eng_pipeline_mod.h"
00134 #include "m_eng_profile.h"
00135 #include "m_eng_update.h"
00136
00137 #ifndef M_SIMULATED
00138 #include "m_eng_i2s_dma.h"
00139 #include "m_eng_sgtl5000.h"
00140 #include "m_eng_memcpy_audio.h"
```

```
00141 #endif
00142
00143 #include "m_eng_context.h"
00144 #include "m_eng_comms.h"
00145
00146 #include "m_eng_transition.h"
00147
00148 #include "m_eng_printf.h"
00149 #include "m_eng_debugging.h"
00150
00151 #ifdef M_SIMULATED
00152 #include "m_eng_sim.h"
00153 #endif
00154
00155 #define sqr(x) (x * x)
00156
00157 #define binary_max(x, y) ((x > y) ? x : y)
00158 #define binary_min(x, y) ((x > y) ? y : x)
00159
00160 float trig_transition_function(float x);
00161
00162 #ifdef __cplusplus
00163 }
00164 #endif
00165
00166 #endif
00167 #endif
```

4.3 m_eng_adaptive_waveshaper.h File Reference

Data Structures

- struct m_eng_adaptive_waveshaper_str

Macros

- #define ADAPTIVE_WAVESHAPER_ENVELOPE_ATTACK expf(-7.0/8.0)
- #define ADAPTIVE_WAVESHAPER_ENVELOPE_RELEASE expf(-7.0/2048.0)

Functions

- int init_adaptive_waveshaper_str (m_eng_adaptive_waveshaper_str ∗str)
- int calc_adaptive_waveshaper (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.3.1 Macro Definition Documentation

4.3.1.1 ADAPTIVE_WAVESHAPER_ENVELOPE_ATTACK

```
#define ADAPTIVE_WAVESHAPER_ENVELOPE_ATTACK expf(-7.0/8.0)
```

4.3.1.2 ADAPTIVE_WAVESHAPER_ENVELOPE_RELEASE

```
#define ADAPTIVE_WAVESHAPER_ENVELOPE_RELEASE expf(-7.0/2048.0)
```

4.3.2 Function Documentation

4.3.2.1 calc_adaptive_waveshaper()

```
int calc_adaptive_waveshaper (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.3.2.2 init_adaptive_waveshaper_str()

```
int init_adaptive_waveshaper_str (
            m_eng_adaptive_waveshaper_str * str)
```

4.4 m_eng_adaptive_waveshaper.h

Go to the documentation of this file.

```
00001 #ifndef M_ADAPTIVE_WAVESHAPER_H_
00002 #define M_ADAPTIVE_WAVESHAPER_H_
00003
00004 #define ADAPTIVE_WAVESHAPER_ENVELOPE_ATTACK     expf(-7.0/8.0)
00005 #define ADAPTIVE_WAVESHAPER_ENVELOPE_RELEASE expf(-7.0/2048.0)
00006
00007 typedef struct
00008 {
00009     m_parameter coefficient;
00010
00011     float (*shape)(float x);
00012
00013     float local_amplitude;
00014 } m_eng_adaptive_waveshaper_str;
00015
00016 int init_adaptive_waveshaper_str(m_eng_adaptive_waveshaper_str *str);
00017 int calc_adaptive_waveshaper(void *data_struct, float *dest, float *src, int n_samples);
00018
00019
00020 #endif
```

4.5 m_eng_audio_block.h File Reference

Macros

- #define AUDIO_BLOCK_SAMPLES 128

4.5.1 Macro Definition Documentation

4.5.1.1 AUDIO_BLOCK_SAMPLES

```
#define AUDIO_BLOCK_SAMPLES 128
```

4.6 m_eng_audio_block.h

Go to the documentation of this file.

```
00001 #ifndef M_AUDIO_BLOCK_H_
00002 #define M_AUDIO_BLOCK_H_
00003
00004 #ifndef AUDIO_BLOCK_SAMPLES
00005 #define AUDIO_BLOCK_SAMPLES  128
00006 #endif
00007
00008 #ifdef __cplusplus
00009 extern "C" {
00010 #endif
00011
00012
00013
00014 #ifdef __cplusplus
00015 }
00016 #endif
00017
00018 #endif
```

4.7 m_eng_band_splitter.h File Reference

Data Structures

- struct m_eng_3_band_splitter_str
- struct m_eng_n_band_splitter_str

Functions

- int init_3_band_splitter_str (m_eng_3_band_splitter_str *str)
- int reconfigure_3_band_splitter (void *data_struct)
- int calc_3_band_splitter (void *data_struct, float **dest, float **src, int n_samples)
- int init_n_band_splitter_str (m_eng_n_band_splitter_str *str)
- int reconfigure_n_band_splitter (void *data_struct)
- int calc_n_band_splitter (void *data_struct, float **dest, float **src, int n_samples)

4.7.1 Function Documentation

4.7.1.1 calc_3_band_splitter()

```
int calc_3_band_splitter (
            void * data_struct,
            float ** dest,
            float ** src,
            int n_samples)
```

4.7.1.2 calc_n_band_splitter()

```
int calc_n_band_splitter (
            void * data_struct,
            float ** dest,
            float ** src,
            int n_samples)
```

4.7.1.3 init_3_band_splitter_str()

```
int init_3_band_splitter_str (
            m_eng_3_band_splitter_str * str)
```

4.7.1.4 init_n_band_splitter_str()

```
int init_n_band_splitter_str (
            m_eng_n_band_splitter_str * str)
```

4.7.1.5 reconfigure_3_band_splitter()

```
int reconfigure_3_band_splitter (
            void * data_struct)
```

4.7.1.6 reconfigure_n_band_splitter()

```
int reconfigure_n_band_splitter (
            void * data_struct)
```

4.8 m_eng_band_splitter.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_TRANSFORMER_H_
00002 #define M_ENG_TRANSFORMER_H_
00003
00004 typedef struct
00005 {
00006     m_lr_low_pass_filter_str filters[3];
00007 } m_eng_3_band_splitter_str;
00008
00009 int init_3_band_splitter_str(m_eng_3_band_splitter_str *str);
00010 int reconfigure_3_band_splitter(void *data_struct);
00011 int calc_3_band_splitter(void *data_struct, float **dest, float **src, int n_samples);
00012
00013 typedef struct
00014 {
00015     m_lr_low_pass_filter_str *filters;
00016 } m_eng_n_band_splitter_str;
00017
00018 int init_n_band_splitter_str(m_eng_n_band_splitter_str *str);
00019 int reconfigure_n_band_splitter(void *data_struct);
00020 int calc_n_band_splitter(void *data_struct, float **dest, float **src, int n_samples);
00021
00022 #endif
00023
```

4.9 m_eng_biquad.h File Reference

Data Structures

- struct m_eng_biquad_str

Functions

- int init_biquad_str (m_eng_biquad_str ∗str)
- int reconfigure_biquad (void ∗data_struct)
- int calc_biquad (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.9.1 Function Documentation

4.9.1.1 calc_biquad()

```
int calc_biquad (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.9.1.2 init_biquad_str()

```
int init_biquad_str (
            m_eng_biquad_str * str)
```

4.9.1.3 reconfigure_biquad()

```
int reconfigure_biquad (
            void * data_struct)
```

4.10 m_eng_biquad.h

Go to the documentation of this file.

```
00001 #ifndef M_BIQUAD_H_
00002 #define M_BIQUAD_H_
00003
00004 typedef struct
00005 {
00006     float a0, a1, a2, a3, a4;
00007     float x1, x2, y1, y2;
00008
00009     m_setting type;
00010     m_parameter cutoff;
00011     m_parameter bandwidth;
00012     m_parameter db_gain;
00013 } m_eng_biquad_str;
00014
00015 int init_biquad_str(m_eng_biquad_str *str);
00016 int reconfigure_biquad(void *data_struct);
00017 int calc_biquad(void *data_struct, float *dest, float *src, int n_samples);
00018
00019 #endif
```

4.11 m_eng_buffer_mixer_amp.h File Reference

Data Structures

- struct m_eng_amplifier_str
- struct m_eng_mixer_str

Macros

- #define M_ENG_AMPLIFIER_LINEAR 0
- #define M_ENG_AMPLIFIER_DB 1

Functions

- int calc_buffer (void *data_struct, float *dest, float *src, int n_samples)
- int init_amplifier_str (m_eng_amplifier_str *str)
- int reconfigure_amplifier (void *data_struct)
- int calc_amplifier (void *data_struct, float *dest, float *src, int n_samples)

4.11.1 Macro Definition Documentation

4.11.1.1 M_ENG_AMPLIFIER_DB

```
#define M_ENG_AMPLIFIER_DB 1
```

4.11.1.2 M_ENG_AMPLIFIER_LINEAR

```
#define M_ENG_AMPLIFIER_LINEAR 0
```

4.11.2 Function Documentation

4.11.2.1 calc_amplifier()

```
int calc_amplifier (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.11.2.2 calc_buffer()

```
int calc_buffer (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.11.2.3 init_amplifier_str()

```
int init_amplifier_str (
            m_eng_amplifier_str * str)
```

4.11.2.4 reconfigure_amplifier()

```
int reconfigure_amplifier (
            void * data_struct)
```

4.12 m_eng_buffer_mixer_amp.h

Go to the documentation of this file.

```
00001 #ifndef M_BMA_H_
00002 #define M_BMA_H_
00003
00004 #define M_ENG_AMPLIFIER_LINEAR      0
00005 #define M_ENG_AMPLIFIER_DB          1
00006
00007 typedef struct
00008 {
00009     m_parameter gain;
00010     float g;
00011
00012     m_setting mode;
00013     int db;
00014 } m_eng_amplifier_str;
00015
00016 typedef struct
00017 {
00018     m_parameter ratio;
00019 } m_eng_mixer_str;
00020
00021
00022 int calc_buffer(void *data_struct, float *dest, float *src, int n_samples);
00023
00024 int init_amplifier_str(m_eng_amplifier_str *str);
00025 int reconfigure_amplifier(void *data_struct);
00026 int calc_amplifier(void *data_struct, float *dest, float *src, int n_samples);
00027
00028 /*
00029 int init_mixer_str(m_eng_mixer_str *str);
00030 int calc_mixer(void *data_struct, float **dest, float **src, int n_samples);
00031 */
00032
00033 #endif
```

4.13 m_eng_comms.h File Reference

Macros

- #define TEENSY_I2C_SLAVE_ADDR 0x08

Functions

- int init_esp32_link ()
- void esp32_message_check_handle ()
- void i2c_receive_isr (int n)
- void i2c_request_isr ()

4.13.1 Macro Definition Documentation

4.13.1.1 TEENSY_I2C_SLAVE_ADDR

```
#define TEENSY_I2C_SLAVE_ADDR 0x08
```

4.13.2 Function Documentation

4.13.2.1 esp32_message_check_handle()

```
void esp32_message_check_handle ()
```

4.13.2.2 i2c_receive_isr()

```
void i2c_receive_isr (
            int n)
```

4.13.2.3 i2c_request_isr()

```
void i2c_request_isr ()
```

4.13.2.4 init_esp32_link()

```
int init_esp32_link ()
```

4.14 m_eng_comms.h

Go to the documentation of this file.

```
00001 #ifndef M_INTERFACE_H_
00002 #define M_INTERFACE_H_
00003
00004 #ifdef __cplusplus
00005 extern "C" {
00006 #endif
00007
00008 #define TEENSY_I2C_SLAVE_ADDR 0x08
00009
00010 int init_esp32_link();
00011
00012 void esp32_message_check_handle();
00013
00014 void i2c_receive_isr(int n);
00015 void i2c_request_isr();
00016
00017 #ifdef __cplusplus
00018 }
00019 #endif
00020
00021 #endif
```

4.15 m_eng_compressor.h File Reference

Data Structures

- struct m_eng_compressor_str

Functions

- int init_compressor_str (m_eng_compressor_str ∗str)
- int reconfigure_compressor (void ∗data_struct)
- int calc_compressor (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.15.1 Function Documentation

4.15.1.1 calc_compressor()

```
int calc_compressor (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.15.1.2 init_compressor_str()

```
int init_compressor_str (
            m_eng_compressor_str * str)
```

4.15.1.3 reconfigure_compressor()

```
int reconfigure_compressor (
            void * data_struct)
```

4.16 m_eng_compressor.h

Go to the documentation of this file.

```
00001 #ifndef M_COMPRESSOR1_H_
00002 #define M_COMPRESSOR1_H_
00003
00004 typedef struct
00005 {
00006     m_parameter ratio;
00007     m_parameter threshold;
00008     m_parameter attack;
00009     m_parameter release;
00010
00011     float alpha;
00012     float rho;
00013
00014     float e_final;
00015 } m_eng_compressor_str;
00016
00017 int init_compressor_str(m_eng_compressor_str *str);
00018 int reconfigure_compressor(void *data_struct);
00019 int calc_compressor(void *data_struct, float *dest, float *src, int n_samples);
00020
00021 #endif
```

4.17 m_eng_context.h File Reference

```
#include "m_eng_profile.h"
```

Data Structures

- struct m_eng_context

Macros

- #define M_PROFILE_SWITCH_SAMPLES 256
- #define PROFILES_MALLOC_CHUNK_SIZE 16
- #define PROFILE_ARRAY_INITIAL_SIZE 64
- #define SILENCE_ENERGY_THRESHOLD 2000
- #define SILENCE_BLOCKS_THRESHOLD 64
- #define DECLICK_BUFSIZE 6
- #define CLICK_SLOPE_THRESHOLD 0.5
- #define DC_BLOCKER_ALPHA 0.999

Functions

- int init_m_eng_context (m_eng_context *cxt)
- int m_eng_context_new_profile (m_eng_context *cxt)
- int cxt_set_active_profile (m_eng_context *cxt, uint16_t pid)
- int cxt_switch_to_profile (m_eng_context *cxt, uint16_t pid)
- int cxt_profile_id_valid (m_eng_context *cxt, uint16_t pid)
- int cxt_transformer_id_valid (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_parameter_id_valid (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid)
- m_parameter * cxt_get_parameter_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid)
- int cxt_update_parameter_value_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid, float new_value)
- m_setting * cxt_get_setting_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t sid)
- int cxt_update_setting_value_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t sid, uint16_t new_value)
- int cxt_append_transformer_to_profile (m_eng_context *cxt, uint16_t pid, uint16_t type)
- int cxt_remove_transformer_from_profile (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_insert_transformer_to_profile (m_eng_context *cxt, uint16_t pid, uint16_t type, uint16_t pos)
- int cxt_prepend_transformer_to_profile (m_eng_context *cxt, uint16_t pid, uint16_t type)
- int cxt_get_n_profile_transformers (m_eng_context *cxt, uint16_t pid)
- int cxt_get_n_transformer_params (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_get_n_transformer_settings (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_get_transformer_type (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_get_tid_by_pos (m_eng_context *cxt, uint16_t pid, uint16_t pos)
- m_transformer * cxt_get_transformer_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_move_transformer (m_eng_context *cxt, uint16_t tid, uint16_t new_pos)
- int cxt_process (m_eng_context *cxt)
- void m_eng_safe_reboot (m_eng_context *cxt)
- int reset_context (m_eng_context *cxt)
- int cxt_run_scheduled_maintainance (m_eng_context *cxt)

Variables

- m_eng_context global_cxt

4.17.1 Macro Definition Documentation

4.17.1.1 CLICK_SLOPE_THRESHOLD

```
#define CLICK_SLOPE_THRESHOLD 0.5
```

4.17.1.2 DC_BLOCKER_ALPHA

```
#define DC_BLOCKER_ALPHA 0.999
```

4.17.1.3 DECLICK_BUFSIZE

```
#define DECLICK_BUFSIZE 6
```

4.17.1.4 M_PROFILE_SWITCH_SAMPLES

```
#define M_PROFILE_SWITCH_SAMPLES 256
```

4.17.1.5 PROFILE_ARRAY_INITIAL_SIZE

```
#define PROFILE_ARRAY_INITIAL_SIZE 64
```

4.17.1.6 PROFILES_MALLOC_CHUNK_SIZE

```
#define PROFILES_MALLOC_CHUNK_SIZE 16
```

4.17.1.7 SILENCE_BLOCKS_THRESHOLD

```
#define SILENCE_BLOCKS_THRESHOLD 64
```

4.17.1.8 SILENCE_ENERGY_THRESHOLD

```
#define SILENCE_ENERGY_THRESHOLD 2000
```

4.17.2 Function Documentation

4.17.2.1 cxt_append_transformer_to_profile()

```
int cxt_append_transformer_to_profile (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t type)
```

4.17.2.2 cxt_get_n_profile_transformers()

```
int cxt_get_n_profile_transformers (
            m_eng_context * cxt,
            uint16_t pid)
```

4.17.2.3 cxt_get_n_transformer_params()

```
int cxt_get_n_transformer_params (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.17.2.4 cxt_get_n_transformer_settings()

```
int cxt_get_n_transformer_settings (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.17.2.5 cxt_get_parameter_by_id()

```
m_parameter * cxt_get_parameter_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid)
```

4.17.2.6 cxt_get_setting_by_id()

```
m_setting * cxt_get_setting_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t sid)
```

4.17.2.7 cxt_get_tid_by_pos()

```
int cxt_get_tid_by_pos (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t pos)
```

4.17.2.8 cxt_get_transformer_by_id()

```
m_transformer * cxt_get_transformer_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.17.2.9 cxt_get_transformer_type()

```
int cxt_get_transformer_type (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.17.2.10 cxt_insert_transformer_to_profile()

```
int cxt_insert_transformer_to_profile (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t type,
            uint16_t pos)
```

4.17.2.11 cxt_move_transformer()

```
int cxt_move_transformer (
            m_eng_context * cxt,
            uint16_t tid,
            uint16_t new_pos)
```

4.17.2.12 cxt_parameter_id_valid()

```
int cxt_parameter_id_valid (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid)
```

4.17.2.13 cxt_prepend_transformer_to_profile()

```
int cxt_prepend_transformer_to_profile (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t type)
```

4.17.2.14 cxt_process()

```
int cxt_process (
            m_eng_context * cxt)
```

4.17.2.15 cxt_profile_id_valid()

```
int cxt_profile_id_valid (
            m_eng_context * cxt,
            uint16_t pid)
```

4.17.2.16 cxt_remove_transformer_from_profile()

```
int cxt_remove_transformer_from_profile (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.17.2.17 cxt_run_scheduled_maintainance()

```
int cxt_run_scheduled_maintainance (
            m_eng_context * cxt)
```

4.17.2.18 cxt_set_active_profile()

```
int cxt_set_active_profile (
            m_eng_context * cxt,
            uint16_t pid)
```

4.17.2.19 cxt_switch_to_profile()

```
int cxt_switch_to_profile (
            m_eng_context * cxt,
            uint16_t pid)
```

4.17.2.20 cxt_transformer_id_valid()

```
int cxt_transformer_id_valid (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.17.2.21 cxt_update_parameter_value_by_id()

```
int cxt_update_parameter_value_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid,
            float new_value)
```

4.17.2.22 cxt_update_setting_value_by_id()

```
int cxt_update_setting_value_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t sid,
            uint16_t new_value)
```

4.17.2.23 init_m_eng_context()

```
int init_m_eng_context (
            m_eng_context * cxt)
```

4.17.2.24 m_eng_context_new_profile()

```
int m_eng_context_new_profile (
            m_eng_context * cxt)
```

4.17.2.25 m_eng_safe_reboot()

```
void m_eng_safe_reboot (
            m_eng_context * cxt)
```

4.17.2.26 reset_context()

```
int reset_context (
            m_eng_context * cxt)
```

4.17.3 Variable Documentation

4.17.3.1 global_cxt

```
m_eng_context global_cxt  [extern]
```

4.18 m_eng_context.h

Go to the documentation of this file.

```
00001 #ifndef M_CONTEXT_H_
00002 #define M_CONTEXT_H_
00003
00004 #include "m_eng_profile.h"
00005
00006 #define M_PROFILE_SWITCH_SAMPLES    256
00007 #define PROFILES_MALLOC_CHUNK_SIZE  16
00008 #define PROFILE_ARRAY_INITIAL_SIZE  64
00009
00010 #define SILENCE_ENERGY_THRESHOLD    2000
00011 #define SILENCE_BLOCKS_THRESHOLD    64
00012
00013 #define DECLICK_BUFSIZE 6
00014
00015 #define CLICK_SLOPE_THRESHOLD 0.5
00016
00017 #define DC_BLOCKER_ALPHA 0.999
00018
00019 //#define PRINT_TIMES
00020 //#define PRINT_BLOCKS
00021 //#define PRETTY_PRINT_BLOCKS
00022 //#define PRINT_PROFILE
00023
00024 typedef struct
00025 {
00026     uint16_t status_flags;
00027
00028     int n_profiles;
```

```
00029     int profile_array_size;
00030     m_eng_profile *profiles;
00031
00032     int active_profile;
00033     int profile_switch_triggered;
00034     int new_profile;
00035
00036     #ifdef GRAPH_PIPELIME
00037     m_eng_graph *unconpfigured_pipeline;
00038     #endif
00039
00040     int profiles_switching;
00041     int profile_switch_progress;
00042     int profile_switch_samples;
00043     int profile_switch_type;
00044
00045     float declick_buffer[DECLICK_BUFSIZE];
00046
00047     int profile_maintainance_index;
00048     float prev_block[AUDIO_BLOCK_SAMPLES];
00049
00050     int runs;
00051
00052     m_eng_high_pass_filter_str output_hpf;
00053     m_eng_low_pass_filter_str input_lpf;
00054     m_transformer output_amp;
00055
00056     float dc_blocker_avg;
00057 } m_eng_context;
00058
00059 int init_m_eng_context(m_eng_context *cxt);
00060
00061 int m_eng_context_new_profile(m_eng_context *cxt);
00062
00063 extern m_eng_context global_cxt;
00064
00065 int cxt_set_active_profile(m_eng_context *cxt, uint16_t pid);
00066 int cxt_switch_to_profile (m_eng_context *cxt, uint16_t pid);
00067
00068 int cxt_profile_id_valid    (m_eng_context *cxt, uint16_t pid);
00069 int cxt_transformer_id_valid(m_eng_context *cxt, uint16_t pid, uint16_t tid);
00070 int cxt_parameter_id_valid    (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid);
00071
00072 m_parameter *cxt_get_parameter_by_id(m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid);
00073 int           cxt_update_parameter_value_by_id(m_eng_context *cxt, uint16_t pid, uint16_t tid,
      uint16_t ppid, float new_value);
00074
00075 m_setting *cxt_get_setting_by_id(m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t sid);
00076 int        cxt_update_setting_value_by_id(m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t
      sid, uint16_t new_value);
00077
00078 int cxt_append_transformer_to_profile (m_eng_context *cxt, uint16_t pid, uint16_t type);
00079 int cxt_remove_transformer_from_profile(m_eng_context *cxt, uint16_t pid, uint16_t tid);
00080 int cxt_insert_transformer_to_profile (m_eng_context *cxt, uint16_t pid, uint16_t type, uint16_t pos);
00081 int cxt_prepend_transformer_to_profile(m_eng_context *cxt, uint16_t pid, uint16_t type);
00082
00083 int cxt_get_n_profile_transformers(m_eng_context *cxt, uint16_t pid);
00084 int cxt_get_n_transformer_params  (m_eng_context *cxt, uint16_t pid, uint16_t tid);
00085 int cxt_get_n_transformer_settings (m_eng_context *cxt, uint16_t pid, uint16_t tid);
00086 int cxt_get_transformer_type      (m_eng_context *cxt, uint16_t pid, uint16_t tid);
00087 int cxt_get_tid_by_pos              (m_eng_context *cxt, uint16_t pid, uint16_t pos);
00088
00089 m_transformer *cxt_get_transformer_by_id(m_eng_context *cxt, uint16_t pid, uint16_t tid);
00090
00091 int cxt_move_transformer(m_eng_context *cxt, uint16_t tid, uint16_t new_pos);
00092
00093 int cxt_process(m_eng_context *cxt);
00094
00095 void m_eng_safe_reboot(m_eng_context *cxt);
00096 int reset_context(m_eng_context *cxt);
00097
00098 int cxt_run_scheduled_maintainance(m_eng_context *cxt);
00099
00100 #endif
```

4.19 m_eng_debugging.h File Reference

Functions

- void full_debug_print (m_eng_context *cxt)

- void print_context_info (m_eng_context *cxt, int depth)
- void print_profile_info (m_eng_profile *profile, int depth)
- void print_pipeline_info (m_pipeline *pipeline, int depth)
- void print_transformer_info (m_transformer *trans, int depth)

4.19.1 Function Documentation

4.19.1.1 full_debug_print()

```
void full_debug_print (
            m_eng_context * cxt)
```

4.19.1.2 print_context_info()

```
void print_context_info (
            m_eng_context * cxt,
            int depth)
```

4.19.1.3 print_pipeline_info()

```
void print_pipeline_info (
            m_pipeline * pipeline,
            int depth)
```

4.19.1.4 print_profile_info()

```
void print_profile_info (
            m_eng_profile * profile,
            int depth)
```

4.19.1.5 print_transformer_info()

```
void print_transformer_info (
            m_transformer * trans,
            int depth)
```

4.20 m_eng_debugging.h

Go to the documentation of this file.

```
00001 #ifndef M_DEBUGGING_H_
00002 #define M_DEBUGGING_H_
00003
00004 void full_debug_print(m_eng_context *cxt);
00005 void print_context_info(m_eng_context *cxt, int depth);
00006 void print_profile_info(m_eng_profile *profile, int depth);
00007 void print_pipeline_info(m_pipeline *pipeline, int depth);
00008 void print_transformer_info(m_transformer *trans, int depth);
00009
00010 #endif
```

4.21 m_eng_delay.h File Reference

Data Structures

- struct m_eng_delay_str

Functions

- int init_delay_str (m_eng_delay_str *str)
- int reconfigure_delay (void *data_struct)
- int calc_delay (void *data_struct, float *dest, float *src, int n_samples)

4.21.1 Function Documentation

4.21.1.1 calc_delay()

```
int calc_delay (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.21.1.2 init_delay_str()

```
int init_delay_str (
            m_eng_delay_str * str)
```

4.21.1.3 reconfigure_delay()

```
int reconfigure_delay (
            void * data_struct)
```

4.22 m_eng_delay.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_DELAY_H_
00002 #define M_ENG_DELAY_H_
00003
00004 typedef struct
00005 {
00006     m_setting tempo;
00007     m_setting note;
00008     m_parameter delay_gain;
00009
00010     float delay_samples;
00011     float g;
00012
00013     m_delay_buffer buf;
00014 } m_eng_delay_str;
00015
00016 int init_delay_str(m_eng_delay_str *str);
00017 int reconfigure_delay(void *data_struct);
00018 int calc_delay(void *data_struct, float *dest, float *src, int n_samples);
00019
00020 #endif
```

4.23 m_eng_delay_buffer.h File Reference

Data Structures

- struct m_delay_buffer

Functions

- int init_delay_buffer (m_delay_buffer *buf)
- int m_delay_buffer_resize_samples (m_delay_buffer *buf, int n_samples)
- int m_delay_buffer_resize_seconds (m_delay_buffer *buf, float seconds)
- int m_delay_buffer_resize_milliseconds (m_delay_buffer *buf, float ms)
- int m_delay_buffer_tick (m_delay_buffer *buf, float new_sample)
- int m_delay_buffer_advance (m_delay_buffer *buf, float *new_samples, unsigned int n)
- int m_delay_buffer_get_delayed_sample (m_delay_buffer *buf, float *dest, unsigned int delay)
- float * m_delay_buffer_get_delayed_sample_ptr (m_delay_buffer *buf, unsigned int delay)
- int m_delay_buffer_get_fractional_delayed_sample (m_delay_buffer *buf, float *dest, float delay)

4.23.1 Function Documentation

4.23.1.1 init_delay_buffer()

```
int init_delay_buffer (
            m_delay_buffer * buf)
```

4.23.1.2 m_delay_buffer_advance()

```
int m_delay_buffer_advance (
            m_delay_buffer * buf,
            float * new_samples,
            unsigned int n)
```

4.23.1.3 m_delay_buffer_get_delayed_sample()

```
int m_delay_buffer_get_delayed_sample (
            m_delay_buffer * buf,
            float * dest,
            unsigned int delay)
```

4.23.1.4 m_delay_buffer_get_delayed_sample_ptr()

```
float * m_delay_buffer_get_delayed_sample_ptr (
            m_delay_buffer * buf,
            unsigned int delay)
```

4.23.1.5 m_delay_buffer_get_fractional_delayed_sample()

```
int m_delay_buffer_get_fractional_delayed_sample (
            m_delay_buffer * buf,
            float * dest,
            float delay)
```

4.23.1.6 m_delay_buffer_resize_milliseconds()

```
int m_delay_buffer_resize_milliseconds (
            m_delay_buffer * buf,
            float ms)
```

4.23.1.7 m_delay_buffer_resize_samples()

```
int m_delay_buffer_resize_samples (
            m_delay_buffer * buf,
            int n_samples)
```

4.23.1.8 m_delay_buffer_resize_seconds()

```
int m_delay_buffer_resize_seconds (
            m_delay_buffer * buf,
            float seconds)
```

4.23.1.9 m_delay_buffer_tick()

```
int m_delay_buffer_tick (
            m_delay_buffer * buf,
            float new_sample)
```

4.24 m_eng_delay_buffer.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_DELAY_BUFFER_H_
00002 #define M_ENG_DELAY_BUFFER_H_
00003
00004 typedef struct
00005 {
00006     int valid;
00007
00008     int pos;
00009     int index;
00010
00011     int n_buffers;
00012     int buffer_array_size;
00013     float **buffers;
00014 } m_delay_buffer;
00015
00016 int init_delay_buffer(m_delay_buffer *buf);
00017
00018 int m_delay_buffer_resize_samples(m_delay_buffer *buf, int n_samples);
00019 int m_delay_buffer_resize_seconds(m_delay_buffer *buf, float seconds);
00020 int m_delay_buffer_resize_milliseconds(m_delay_buffer *buf, float ms);
00021
00022 int m_delay_buffer_tick(m_delay_buffer *buf, float new_sample);
00023 int m_delay_buffer_advance(m_delay_buffer *buf, float *new_samples, unsigned int n);
00024
00025 int m_delay_buffer_get_delayed_sample(m_delay_buffer *buf, float *dest, unsigned int delay);
00026 float *m_delay_buffer_get_delayed_sample_ptr(m_delay_buffer *buf, unsigned int delay);
00027 int m_delay_buffer_get_fractional_delayed_sample(m_delay_buffer *buf, float *dest, float delay);
00028
00029 #endif
```

4.25 m_eng_dirty_octave.h File Reference

Data Structures

- struct m_eng_dirty_octave_str

Functions

- int init_dirty_octave_str (m_eng_dirty_octave_str *str)
- int reconfigure_dirty_octave (void *data_struct)
- int calc_dirty_octave (void *data_struct, float *dest, float *src, int n_samples)

4.25.1 Function Documentation

4.25.1.1 calc_dirty_octave()

```
int calc_dirty_octave (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.25.1.2 init_dirty_octave_str()

```
int init_dirty_octave_str (
            m_eng_dirty_octave_str * str)
```

4.25.1.3 reconfigure_dirty_octave()

```
int reconfigure_dirty_octave (
            void * data_struct)
```

4.26 m_eng_dirty_octave.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_DIRTY_OCTAVE_H_
00002 #define M_ENG_DIRTY_OCTAVE_H_
00003
00004 typedef struct
00005 {
00006     m_parameter fuzz;
00007
00008     float dc_average;
00009
00010     float lpf_alpha;
00011     float last_out_sample;
00012 } m_eng_dirty_octave_str;
00013
00014 int init_dirty_octave_str(m_eng_dirty_octave_str *str);
00015 int reconfigure_dirty_octave(void *data_struct);
00016 int calc_dirty_octave(void *data_struct, float *dest, float *src, int n_samples);
00017
00018 #endif
```

4.27 m_eng_distortion.h File Reference

Data Structures

- struct m_eng_distortion_str

Macros

- #define USE_GLOBAL_TEMP_BUFFERS
- #define M_DISTORTION_CLIP 0
- #define M_DISTORTION_TANH 1
- #define M_DISTORTION_ARCTAN 2
- #define M_DISTORTION_FOLD 3

Functions

- int init_distortion_str (m_eng_distortion_str ∗str)
- int reconfigure_distortion (void ∗data_struct)
- int calc_distortion (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.27.1 Macro Definition Documentation

4.27.1.1 M_DISTORTION_ARCTAN

```
#define M_DISTORTION_ARCTAN 2
```

4.27.1.2 M_DISTORTION_CLIP

```
#define M_DISTORTION_CLIP 0
```

4.27.1.3 M_DISTORTION_FOLD

```
#define M_DISTORTION_FOLD 3
```

4.27.1.4 M_DISTORTION_TANH

```
#define M_DISTORTION_TANH 1
```

4.27.1.5 USE_GLOBAL_TEMP_BUFFERS

```
#define USE_GLOBAL_TEMP_BUFFERS
```

4.27.2 Function Documentation

4.27.2.1 calc_distortion()

```
int calc_distortion (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.27.2.2 init_distortion_str()

```
int init_distortion_str (
            m_eng_distortion_str * str)
```

4.27.2.3 reconfigure_distortion()

```
int reconfigure_distortion (
            void * data_struct)
```

4.28 m_eng_distortion.h

Go to the documentation of this file.

```
00001 #ifndef M_DISTORTION_H_
00002 #define M_DISTORTION_H_
00003
00004 #define USE_GLOBAL_TEMP_BUFFERS
00005
00006 #define M_DISTORTION_CLIP   0
00007 #define M_DISTORTION_TANH   1
00008 #define M_DISTORTION_ARCTAN 2
00009 #define M_DISTORTION_FOLD   3
00010
00011 typedef struct
00012 {
00013     m_setting function;
00014
00015     m_lr_low_pass_filter_str  low_pass;
00016
00017     m_eng_waveshaper_str dist;
00018
00019     m_parameter wet_mix;
00020     m_parameter bass_mix;
00021     m_parameter bass_cutoff;
00022 } m_eng_distortion_str;
00023
00024 int init_distortion_str(m_eng_distortion_str *str);
00025 int reconfigure_distortion(void *data_struct);
00026 int calc_distortion(void *data_struct, float *dest, float *src, int n_samples);
00027
00028 #endif
```

4.29 m_eng_envelope.h File Reference

Data Structures

- struct m_eng_envelope_str

Functions

- int init_envelope_str (m_eng_envelope_str ∗str)
- int reconfigure_envelope (void ∗data_struct)
- int calc_envelope (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.29.1 Function Documentation

4.29.1.1 calc_envelope()

```
int calc_envelope (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.29.1.2 init_envelope_str()

```
int init_envelope_str (
            m_eng_envelope_str * str)
```

4.29.1.3 reconfigure_envelope()

```
int reconfigure_envelope (
            void * data_struct)
```

4.30 m_eng_envelope.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_ENVELOPE_H_
00002 #define M_ENG_ENVELOPE_H_
00003
00004 typedef struct
00005 {
00006     m_parameter min_center;
00007     m_parameter max_center;
00008     m_parameter width;
00009     m_parameter speed;
00010     m_parameter sensitivity;
00011     m_parameter smoothness;
00012
00013     float alpha;
00014     float e;
00015
00016     int chunk_size;
00017
00018     m_eng_band_pass_filter_str filter;
00019 } m_eng_envelope_str;
00020
00021 int init_envelope_str(m_eng_envelope_str *str);
00022 int reconfigure_envelope(void *data_struct);
00023 int calc_envelope(void *data_struct, float *dest, float *src, int n_samples);
00024
00025 #endif
```

4.31 m_eng_equaliser.h File Reference

Data Structures

- struct m_eng_3_band_eq_str

Macros

- #define M_ENG_EQ_CONTROL_DIRECT 0
- #define M_ENG_EQ_CONTROL_DB_GAIN 1

Functions

- int init_3_band_eq_str (m_eng_3_band_eq_str ∗str)
- int reconfigure_3_band_eq (void ∗data_struct)
- int calc_3_band_eq (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.31.1 Macro Definition Documentation

4.31.1.1 M_ENG_EQ_CONTROL_DB_GAIN

```
#define M_ENG_EQ_CONTROL_DB_GAIN 1
```

4.31.1.2 M_ENG_EQ_CONTROL_DIRECT

```
#define M_ENG_EQ_CONTROL_DIRECT 0
```

4.31.2 Function Documentation

4.31.2.1 calc_3_band_eq()

```
int calc_3_band_eq (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.31.2.2 init_3_band_eq_str()

```
int init_3_band_eq_str (
            m_eng_3_band_eq_str * str)
```

4.31.2.3 reconfigure_3_band_eq()

```
int reconfigure_3_band_eq (
            void * data_struct)
```

4.32 m_eng_equaliser.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_EQUALISER_H_
00002 #define M_ENG_EQUALISER_H_
00003
00004 #define M_ENG_EQ_CONTROL_DIRECT     0
00005 #define M_ENG_EQ_CONTROL_DB_GAIN    1
00006
00007 typedef struct
00008 {
00009     m_parameter low;
00010     m_parameter mid;
00011     m_parameter high;
00012
00013     float coefs[3];
00014     int control_mode;
00015
00016     m_lr_low_pass_filter_str filters[2];
00017 } m_eng_3_band_eq_str;
00018
00019 int init_3_band_eq_str(m_eng_3_band_eq_str *str);
00020 int reconfigure_3_band_eq(void *data_struct);
00021 int calc_3_band_eq(void *data_struct, float *dest, float *src, int n_samples);
00022
00023 #endif
```

4.33 m_eng_flanger.h File Reference

Data Structures

- struct m_eng_flanger_str

Functions

- int init_flanger_str (m_eng_flanger_str ∗str)
- int reconfigure_flanger (void ∗data_struct)
- int calc_flanger (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.33.1 Function Documentation

4.33.1.1 calc_flanger()

```
int calc_flanger (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.33.1.2 init_flanger_str()

```
int init_flanger_str (
            m_eng_flanger_str * str)
```

4.33.1.3 reconfigure_flanger()

```
int reconfigure_flanger (
            void * data_struct)
```

4.34 m_eng_flanger.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_FLANGER_H_
00002 #define M_ENG_FLANGER_H_
00003
00004 typedef struct
00005 {
00006     m_parameter range;
00007     m_parameter tempo;
00008     m_parameter depth;
00009     m_parameter mix;
00010
00011     m_setting note;
00012
00013     float r;
00014     float s;
00015     float d;
00016
00017     float wet_mix;
00018     float dry_mix;
00019
00020     float period;
00021
00022     float t;
00023
00024     m_delay_buffer buf;
00025 } m_eng_flanger_str;
00026
00027 int init_flanger_str(m_eng_flanger_str *str);
00028 int reconfigure_flanger(void *data_struct);
00029 int calc_flanger(void *data_struct, float *dest, float *src, int n_samples);
00030
00031 #endif
```

4.35 m_eng_flops.h File Reference

Macros

- #define RESTRICT

4.35.1 Macro Definition Documentation

4.35.1.1 RESTRICT

```
#define RESTRICT
```

4.36 m_eng_flops.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_FLOPS_H_
00002 #define M_ENG_FLOPS_H_
00003
00004 #ifndef RESTRICT
00005 #  if defined(__STDC_VERSION__) && __STDC_VERSION__ >= 199901L
00006 #    define RESTRICT restrict
00007 #  elif defined(__GNUC__) || defined(__clang__)
00008 #    define RESTRICT __restrict__
00009 #  else
00010 #    define RESTRICT
00011 #  endif
00012 #endif
00013
00014 static inline void m_add_f32(float *dest, float *RESTRICT x, float *RESTRICT y, uint32_t n)
00015 {
00016     if (dest && x && y)
00017     {
00018         #ifdef NO_CMSIS
00019         int i = 0;
00020         for (; i + 3 < n; i += 4)
00021         {
00022             dest[i    ] = x[i    ] + y[i    ];
00023             dest[i + 1] = x[i + 1] + y[i + 1];
00024             dest[i + 2] = x[i + 2] + y[i + 2];
00025             dest[i + 3] = x[i + 3] + y[i + 3];
00026         }
00027         for (; i < n; i++) dest[i] = x[i] + y[i];
00028         #else
00029         arm_add_f32(x, y, dest, n);
00030         #endif
00031     }
00032 }
00033
00034 static inline void m_sub_f32(float *dest, float *RESTRICT x, float *RESTRICT y, uint32_t n)
00035 {
00036     if (dest && x && y)
00037     {
00038         #ifdef NO_CMSIS
00039         int i = 0;
00040         for (; i + 3 < n; i += 4)
00041         {
00042             dest[i    ] = x[i    ] - y[i    ];
00043             dest[i + 1] = x[i + 1] - y[i + 1];
00044             dest[i + 2] = x[i + 2] - y[i + 2];
00045             dest[i + 3] = x[i + 3] - y[i + 3];
00046         }
00047         for (; i < n; i++) dest[i] = x[i] - y[i];
00048         #else
00049         arm_sub_f32(x, y, dest, n);
00050         #endif
00051     }
00052 }
00053
00054
00055 static inline void m_add_to_f32(float *dest, float *x, uint32_t n)
00056 {
00057     if (dest && x)
00058     {
00059         #ifdef NO_CMSIS
00060         int i = 0;
00061         for (; i + 3 < n; i += 4)
00062         {
00063             dest[i    ] += x[i    ];
00064             dest[i + 1] += x[i + 1];
00065             dest[i + 2] += x[i + 2];
00066             dest[i + 3] += x[i + 3];
00067         }
00068         for (; i < n; i++) dest[i] += x[i];
00069         #else
00070         arm_add_f32(dest, x, dest, n);
00071         #endif
00072     }
00073 }
00074
00075 static inline void m_sub_from_f32(float *dest, float *x, uint32_t n)
00076 {
00077     if (dest && x)
00078     {
00079         #ifdef NO_CMSIS
00080         int i = 0;
00081         for (; i + 3 < n; i += 4)
00082         {
```

```
00083             dest[i    ] -= x[i    ];
00084             dest[i + 1] -= x[i + 1];
00085             dest[i + 2] -= x[i + 2];
00086             dest[i + 3] -= x[i + 3];
00087         }
00088         for (; i < n; i++) dest[i] -= x[i];
00089         #else
00090         arm_sub_f32(dest, x, dest, n);
00091         #endif
00092     }
00093 }
00094
00095
00096 static inline void m_mul_and_add_f32(float *dest, float *RESTRICT x, float *RESTRICT y, uint32_t n)
00097 {
00098     if (dest && x && y)
00099     {
00100         #ifdef NO_CMSIS
00101         int i = 0;
00102         for (; i + 3 < n; i += 4)
00103         {
00104             dest[i]     += x[i]     * y[i];
00105             dest[i + 1] += x[i + 1] * y[i + 1];
00106             dest[i + 2] += x[i + 2] * y[i + 2];
00107             dest[i + 3] += x[i + 3] * y[i + 3];
00108         }
00109         for (; i < n; i++) dest[i] += x[i] * y[i];
00110         #else
00111         float temp[n];
00112         arm_mult_f32(x, y, temp, n);
00113         arm_add_f32(dest, temp, dest, n);
00114         #endif
00115     }
00116 }
00117
00118 static inline void m_scale_and_add_to_f32(float *RESTRICT dest, float *RESTRICT x, float s, uint32_t
      n)
00119 {
00120     if (dest && x)
00121     {
00122         #ifdef NO_CMSIS
00123         int i = 0;
00124         for (; i + 3 < n; i += 4)
00125         {
00126             dest[i]     += x[i]     * s;
00127             dest[i + 1] += x[i + 1] * s;
00128             dest[i + 2] += x[i + 2] * s;
00129             dest[i + 3] += x[i + 3] * s;
00130         }
00131         for (; i < n; i++) dest[i] += x[i] * s;
00132         #else
00133         float temp[n];
00134         arm_scale_f32(x, s, temp, n);
00135         arm_add_f32(dest, temp, dest, n);
00136         #endif
00137     }
00138 }
00139
00140
00141 static inline void m_scale_f32(float *dest, float *x, float s, uint32_t n)
00142 {
00143     if (dest && x)
00144     {
00145         #ifdef NO_CMSIS
00146         int i = 0;
00147         for (; i + 3 < n; i += 4)
00148         {
00149             dest[i]     = x[i]     * s;
00150             dest[i + 1] = x[i + 1] * s;
00151             dest[i + 2] = x[i + 2] * s;
00152             dest[i + 3] = x[i + 3] * s;
00153         }
00154         for (; i < n; i++) dest[i] = x[i] * s;
00155         #else
00156         arm_scale_f32(x, s, dest, n);
00157         #endif
00158     }
00159 }
00160
00161 static inline void m_scale_in_place_f32(float *x, float s, uint32_t n)
00162 {
00163     if (x)
00164     {
00165         #ifdef NO_CMSIS
00166         int i = 0;
00167         for (; i + 3 < n; i += 4)
00168         {
```

```
00169             x[i    ] *= s;
00170             x[i + 1] *= s;
00171             x[i + 2] *= s;
00172             x[i + 3] *= s;
00173         }
00174         for (; i < n; i++) x[i] *= s;
00175         #else
00176         arm_scale_f32(x, s, x, n);
00177         #endif
00178     }
00179 }
00180
00181 static inline void m_linterpolate_f32(float *dest, float *x, float *y, float t, uint32_t n)
00182 {
00183     if (dest && x && y)
00184     {
00185         if (t <= 0.0)
00186         {
00187             memcpy(dest, x, n * sizeof(float));
00188         }
00189         if (t >= 1.0)
00190         {
00191             memcpy(dest, y, n * sizeof(float));
00192         }
00193         else
00194         {
00195             int i = 0;
00196             for (; i + 3 < n; i += 4)
00197             {
00198                 dest[i    ] = (1.0 - t) * x[i    ] + t * y[i    ];
00199                 dest[i + 1] = (1.0 - t) * x[i + 1] + t * y[i + 1];
00200                 dest[i + 2] = (1.0 - t) * x[i + 2] + t * y[i + 2];
00201                 dest[i + 3] = (1.0 - t) * x[i + 3] + t * y[i + 3];
00202             }
00203             for (; i < n; i++) dest[i] = (1.0 - t) * x[i] + t * y[i];
00204         }
00205     }
00206 }
00207
00208 static inline void m_mix_in_f32(float *x, float *y, float t, uint32_t n)
00209 {
00210     if (x && y)
00211     {
00212         if (t <= 0.0)
00213         {
00214             return;
00215         }
00216         if (t >= 1.0)
00217         {
00218             memcpy(x, y, n * sizeof(float));
00219         }
00220         else
00221         {
00222             int i = 0;
00223             for (; i + 3 < n; i += 4)
00224             {
00225                 x[i    ] = (1.0 - t) * x[i    ] + t * y[i    ];
00226                 x[i + 1] = (1.0 - t) * x[i + 1] + t * y[i + 1];
00227                 x[i + 2] = (1.0 - t) * x[i + 2] + t * y[i + 2];
00228                 x[i + 3] = (1.0 - t) * x[i + 3] + t * y[i + 3];
00229             }
00230             for (; i < n; i++) x[i] = (1.0 - t) * x[i] + t * y[i];
00231         }
00232     }
00233 }
00234
00235 #endif
```

4.37 m_eng_globals.h File Reference

Functions

- void update_upper_cycles ()
- double cycles_to_seconds (uint64_t cycles)
- uint64_t current_cycle ()

Variables

- uint16_t cpu_cycles_total
- uint16_t cpu_cycles_total_max
- uint16_t memory_used
- uint16_t memory_used_max
- int update_scheduled
- uint32_t trace_depth

4.37.1 Function Documentation

4.37.1.1 current_cycle()

```
uint64_t current_cycle ()  [inline]
```

4.37.1.2 cycles_to_seconds()

```
double cycles_to_seconds (
            uint64_t cycles)
```

4.37.1.3 update_upper_cycles()

```
void update_upper_cycles ()
```

4.37.2 Variable Documentation

4.37.2.1 cpu_cycles_total

```
uint16_t cpu_cycles_total  [extern]
```

4.37.2.2 cpu_cycles_total_max

```
uint16_t cpu_cycles_total_max  [extern]
```

4.37.2.3 memory_used

```
uint16_t memory_used  [extern]
```

4.37.2.4 memory_used_max

```
uint16_t memory_used_max  [extern]
```

4.37.2.5 trace_depth

```
uint32_t trace_depth  [extern]
```

4.37.2.6 update_scheduled

```
int update_scheduled  [extern]
```

4.38 m_eng_globals.h

Go to the documentation of this file.

```
00001 #ifndef M_GLOBAL_VARS_H_
00002 #define M_GLOBAL_VARS_H_
00003
00004 extern uint16_t cpu_cycles_total;
00005 extern uint16_t cpu_cycles_total_max;
00006 extern uint16_t memory_used;
00007 extern uint16_t memory_used_max;
00008
00009 extern int update_scheduled;
00010
00011 extern uint32_t trace_depth;
00012
00013 void update_upper_cycles();
00014
00015 double cycles_to_seconds(uint64_t cycles);
00016
00017 uint64_t current_cycle();
00018
00019 #endif
```

4.39 m_eng_i2s_dma.h File Reference

Typedefs

- typedef int16_t raw_sample_t

Functions

- void init_i2s_dma ()
- void m_eng_i2s_input_isr ()
- void m_eng_i2s_output_isr ()
- void i2s_input_update ()
- void i2s_output_update ()
- void i2s_output_transmit_mono_int (raw_sample_t *block)
- void i2s_output_transmit_mono_float (float *block)

Variables

- raw_sample_t i2s_input_blocks [2][AUDIO_BLOCK_SAMPLES]

4.39.1 Typedef Documentation

4.39.1.1 raw_sample_t

```
typedef int16_t raw_sample_t
```

4.39.2 Function Documentation

4.39.2.1 i2s_input_update()

```
void i2s_input_update ()
```

4.39.2.2 i2s_output_transmit_mono_float()

```
void i2s_output_transmit_mono_float (
            float * block)
```

4.39.2.3 i2s_output_transmit_mono_int()

```
void i2s_output_transmit_mono_int (
            raw_sample_t * block)
```

4.39.2.4 i2s_output_update()

```
void i2s_output_update ()
```

4.39.2.5 init_i2s_dma()

```
void init_i2s_dma ()
```

4.39.2.6 m_eng_i2s_input_isr()

```
void m_eng_i2s_input_isr ()
```

4.39.2.7 m_eng_i2s_output_isr()

```
void m_eng_i2s_output_isr ()
```

4.39.3 Variable Documentation

4.39.3.1 i2s_input_blocks

```
raw_sample_t i2s_input_blocks[2][AUDIO_BLOCK_SAMPLES]  [extern]
```

4.40 m_eng_i2s_dma.h

Go to the documentation of this file.

```
00001 #ifndef M_I2S_DMA_H_
00002 #define M_I2S_DMA_H_
00003
00004 typedef int16_t raw_sample_t;
00005
00006 void init_i2s_dma();
00007
00008 void m_eng_i2s_input_isr();
00009 void m_eng_i2s_output_isr();
00010
00011 void i2s_input_update();
00012 void i2s_output_update();
00013
00014 void i2s_output_transmit_mono_int  (raw_sample_t *block);
00015 void i2s_output_transmit_mono_float(float *block);
00016
00017 extern raw_sample_t i2s_input_blocks[2][AUDIO_BLOCK_SAMPLES];
00018
00019 #endif
```

4.41 m_eng_linkowitz_riley.h File Reference

Data Structures

- struct m_lr_low_pass_filter_str
- struct m_lr_high_pass_filter_str

Functions

- int init_lr_low_pass_filter_str (m_lr_low_pass_filter_str *str)
- int reconfigure_lr_low_pass_filter (void *data_struct)
- int calc_lr_low_pass_filter (void *data_struct, float *dest, float *src, int n_samples)
- int init_lr_high_pass_filter_str (m_lr_high_pass_filter_str *str)
- int reconfigure_lr_high_pass_filter (void *data_struct)
- int calc_lr_high_pass_filter (void *data_struct, float *dest, float *src, int n_samples)

4.41.1 Function Documentation

4.41.1.1 calc_lr_high_pass_filter()

```
int calc_lr_high_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.41.1.2 calc_lr_low_pass_filter()

```
int calc_lr_low_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.41.1.3 init_lr_high_pass_filter_str()

```
int init_lr_high_pass_filter_str (
            m_lr_high_pass_filter_str * str)
```

4.41.1.4 init_lr_low_pass_filter_str()

```
int init_lr_low_pass_filter_str (
            m_lr_low_pass_filter_str * str)
```

4.41.1.5 reconfigure_lr_high_pass_filter()

```
int reconfigure_lr_high_pass_filter (
            void * data_struct)
```

4.41.1.6 reconfigure_lr_low_pass_filter()

```
int reconfigure_lr_low_pass_filter (
            void * data_struct)
```

4.42 m_eng_linkowitz_riley.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_LR_FILTER_H_
00002 #define M_ENG_LR_FILTER_H_
00003
00004 typedef struct m_lr_low_pass_filter_str
00005 {
00006     m_parameter cutoff_frequency;
00007
00008     float a0, a1, a2, a3, a4;
00009     float x_11, x_12, y_11, y_12;
00010     float x_21, x_22, y_21, y_22;
00011
00012 } m_lr_low_pass_filter_str;
00013
00014 int init_lr_low_pass_filter_str(m_lr_low_pass_filter_str *str);
00015 int reconfigure_lr_low_pass_filter(void *data_struct);
00016 int calc_lr_low_pass_filter(void *data_struct, float *dest, float *src, int n_samples);
00017
00018 typedef struct m_lr_high_pass_filter_str
00019 {
00020     m_parameter cutoff_frequency;
00021
00022     float a0, a1, a2, a3, a4;
00023     float x_11, x_12, y_11, y_12;
00024     float x_21, x_22, y_21, y_22;
00025
00026 } m_lr_high_pass_filter_str;
00027
00028 int init_lr_high_pass_filter_str(m_lr_high_pass_filter_str *str);
00029 int reconfigure_lr_high_pass_filter(void *data_struct);
00030 int calc_lr_high_pass_filter(void *data_struct, float *dest, float *src, int n_samples);
00031
00032 #endif
```

4.43 m_eng_logging.h File Reference

Data Structures

- struct m_eng_log_entry
- struct m_eng_profiler_entry

Macros

- #define M_ENG_TRACE_FUNCTION_ENTER 0
- #define M_ENG_TRACE_FUNCTION_RETURN 1
- #define M_ENG_LOG_ENTRY_ERROR 2
- #define M_ENG_LOG_ENTRY_RETURN_ERR 3
- #define M_ENG_LOG_ENTRY_RETURN_PTR 4
- #define M_ENG_LOG_ENTRY_RETURN_INT 5
- #define M_ENG_LOG_ENTRY_RETURN 6
- #define M_ENG_LOG_ENTRY_MESSAGE 7
- #define M_ENG_LOG_ERRORS 0b00001
- #define M_ENG_LOG_TRACE 0b00010
- #define M_ENG_LOG_RETURNS 0b00100
- #define M_ENG_LOG_SPIKES 0b01000
- #define M_ENG_LOG_EVERYTHING (M_ENG_LOG_ERRORS | M_ENG_LOG_TRACE | M_ENG_LOG_RETURNS | M_ENG_LOG_SPIKES)
- #define STR(x)
- #define XSTR(x)
- #define M_ENG_LOG_LEVEL (M_ENG_LOG_ERRORS)
- #define M_ENG_LOG_INDENT_TRACE
- #define M_ENG_LOG_ERROR(x)
- #define M_ENG_PROFILER_LOG_ENTRY()
- #define M_ENG_PROFILER_LOG_RETURN()
- #define M_ENG_TRACE_LOG_ENTRY()
- #define M_ENG_TRACE_LOG_RETURN()
- #define M_ENG_LOG_RETURN_ERR_CODE(x)
- #define M_ENG_LOG_RETURN_PTR(x)
- #define M_ENG_LOG_RETURN_INT(x)
- #define M_ENG_LOG_RETURN(x)
- #define FUNCTION_START()
- #define RETURN_VOID()
- #define RETURN_ERR_CODE(x)
- #define RETURN_NEG_ERR_CODE(x)
- #define RETURN_PTR(x)
- #define RETURN_INT(x)
- #define RETURN(x)
- #define M_LOG(fmt, ...)
- #define M_LOG_ERROR(x)
- #define CYCLES_TO_SECONDS(x)
- #define SECONDS_TO_CYCLES(x)

Functions

- void m_eng_log_message (const char *fname, const char *line, const char *function, uint64_t cycle, const char *fmt,...)
- void m_eng_log (const char *fmt,...)
- void m_eng_print_log ()
- void m_eng_print_flush_log ()
- void m_eng_trace_log_begin (const char *fname, const char *line, const char *function, int local_trace_↩ depth, uint64_t cycle)
- void m_eng_trace_log_return (const char *fname, const char *line, const char *function, int local_trace_↩ depth, uint64_t cycle)
- void m_eng_log_error_code (const char *fname, const char *line, const char *function, int error_code, uint64_t cycle)
- void m_eng_log_return_err (const char *fname, const char *line, const char *function, int error_code, uint64_t cycle)
- void m_eng_log_return_ptr (const char *fname, const char *line, const char *function, void *ptr, uint64_t cycle)
- void m_eng_log_return_int (const char *fname, const char *line, const char *function, int val, uint64_t cycle)
- void m_eng_log_return_ (const char *fname, const char *line, const char *function, uint64_t cycle)

4.43.1 Macro Definition Documentation

4.43.1.1 CYCLES_TO_SECONDS

```
#define CYCLES_TO_SECONDS(
            x)
```

Value:
```
((double)(x) * 1.66666667e-9)
```

4.43.1.2 FUNCTION_START

```
#define FUNCTION_START()
```

Value:
```
M_ENG_PROFILER_LOG_ENTRY();  M_ENG_TRACE_LOG_ENTRY();
```

4.43.1.3 M_ENG_LOG_ENTRY_ERROR

```
#define M_ENG_LOG_ENTRY_ERROR 2
```

4.43.1.4 M_ENG_LOG_ENTRY_MESSAGE

```
#define M_ENG_LOG_ENTRY_MESSAGE 7
```

4.43.1.5 M_ENG_LOG_ENTRY_RETURN

```
#define M_ENG_LOG_ENTRY_RETURN 6
```

4.43.1.6 M_ENG_LOG_ENTRY_RETURN_ERR

```
#define M_ENG_LOG_ENTRY_RETURN_ERR 3
```

4.43.1.7 M_ENG_LOG_ENTRY_RETURN_INT

```
#define M_ENG_LOG_ENTRY_RETURN_INT 5
```

4.43.1.8 M_ENG_LOG_ENTRY_RETURN_PTR

```
#define M_ENG_LOG_ENTRY_RETURN_PTR 4
```

4.43.1.9 M_ENG_LOG_ERROR

```
#define M_ENG_LOG_ERROR(
              x)
```

Value:

```
if (x != NO_ERROR) \
  m_eng_log_error_code(FNAME, XSTR(__LINE__), __func__, x, current_cycle());
```

4.43.1.10 M_ENG_LOG_ERRORS

```
#define M_ENG_LOG_ERRORS 0b00001
```

4.43.1.11 M_ENG_LOG_EVERYTHING

```
#define M_ENG_LOG_EVERYTHING (M_ENG_LOG_ERRORS | M_ENG_LOG_TRACE | M_ENG_LOG_RETURNS | M_ENG_LOG_SPIKES)
```

4.43.1.12 M_ENG_LOG_INDENT_TRACE

```
#define M_ENG_LOG_INDENT_TRACE
```

4.43.1.13 M_ENG_LOG_LEVEL

```
#define M_ENG_LOG_LEVEL (M_ENG_LOG_ERRORS)
```

4.43.1.14 M_ENG_LOG_RETURN

```
#define M_ENG_LOG_RETURN(
              x)
```

4.43.1.15 M_ENG_LOG_RETURN_ERR_CODE

```
#define M_ENG_LOG_RETURN_ERR_CODE(
              x)
```

4.43.1.16 M_ENG_LOG_RETURN_INT

```
#define M_ENG_LOG_RETURN_INT(
              x)
```

4.43.1.17 M_ENG_LOG_RETURN_PTR

```
#define M_ENG_LOG_RETURN_PTR(
              x)
```

4.43.1.18 M_ENG_LOG_RETURNS

```
#define M_ENG_LOG_RETURNS 0b00100
```

4.43.1.19 M_ENG_LOG_SPIKES

```
#define M_ENG_LOG_SPIKES 0b01000
```

4.43.1.20 M_ENG_LOG_TRACE

```
#define M_ENG_LOG_TRACE 0b00010
```

4.43.1.21 M_ENG_PROFILER_LOG_ENTRY

```
#define M_ENG_PROFILER_LOG_ENTRY()
```

4.43.1.22 M_ENG_PROFILER_LOG_RETURN

```
#define M_ENG_PROFILER_LOG_RETURN()
```

4.43.1.23 M_ENG_TRACE_FUNCTION_ENTER

```
#define M_ENG_TRACE_FUNCTION_ENTER 0
```

4.43.1.24 M_ENG_TRACE_FUNCTION_RETURN

```
#define M_ENG_TRACE_FUNCTION_RETURN 1
```

4.43.1.25 M_ENG_TRACE_LOG_ENTRY

```
#define M_ENG_TRACE_LOG_ENTRY()
```

4.43.1.26 M_ENG_TRACE_LOG_RETURN

```
#define M_ENG_TRACE_LOG_RETURN()
```

4.43.1.27 M_LOG

```
#define M_LOG(
               fmt,
               ...)
```

Value:

```
m_eng_log_message(FNAME, XSTR(__LINE__), __func__, current_cycle(), fmt, ##__VA_ARGS__);
```

4.43.1.28 M_LOG_ERROR

```
#define M_LOG_ERROR(
               x)
```

Value:

```
m_eng_log_error_code(FNAME, XSTR(__LINE__), __func__, x, current_cycle());
```

4.43.1.29 RETURN

```
#define RETURN(
               x)
```

Value:

```
M_ENG_PROFILER_LOG_RETURN(); M_ENG_TRACE_LOG_RETURN(); M_ENG_LOG_RETURN(); return x;
```

4.43.1.30 RETURN_ERR_CODE

```
#define RETURN_ERR_CODE(
               x)
```

Value:

```
M_ENG_PROFILER_LOG_RETURN(); M_ENG_LOG_ERROR(x); M_ENG_TRACE_LOG_RETURN(); M_ENG_LOG_RETURN_ERR_CODE(x);
      return x;
```

4.43.1.31 RETURN_INT

```
#define RETURN_INT(
               x)
```

Value:

```
M_ENG_PROFILER_LOG_RETURN(); M_ENG_TRACE_LOG_RETURN(); M_ENG_LOG_RETURN_INT(x); return x;
```

4.43.1.32 RETURN_NEG_ERR_CODE

```
#define RETURN_NEG_ERR_CODE(
                x)
```

Value:
```
M_ENG_PROFILER_LOG_RETURN(); M_ENG_LOG_ERROR(x); M_ENG_TRACE_LOG_RETURN(); M_ENG_LOG_RETURN_ERR_CODE(x);
      return -x;
```

4.43.1.33 RETURN_PTR

```
#define RETURN_PTR(
                x)
```

Value:
```
M_ENG_PROFILER_LOG_RETURN(); M_ENG_TRACE_LOG_RETURN(); M_ENG_LOG_RETURN_PTR(x); return x;
```

4.43.1.34 RETURN_VOID

```
#define RETURN_VOID()
```

Value:
```
M_ENG_PROFILER_LOG_RETURN(); M_ENG_TRACE_LOG_RETURN(); return;
```

4.43.1.35 SECONDS_TO_CYCLES

```
#define SECONDS_TO_CYCLES(
                x)
```

Value:
```
((double)(x) / 1.66666667e-9)
```

4.43.1.36 STR

```
#define STR(
                x)
```

Value:
```
#x
```

4.43.1.37 XSTR

```
#define XSTR(
                x)
```

Value:
```
(STR(x))
```

4.43.2 Function Documentation

4.43.2.1 m_eng_log()

```
void m_eng_log (
            const char * fmt,
             ...)
```

4.43.2.2 m_eng_log_error_code()

```
void m_eng_log_error_code (
            const char * fname,
            const char * line,
            const char * function,
            int error_code,
            uint64_t cycle)
```

4.43.2.3 m_eng_log_message()

```
void m_eng_log_message (
            const char * fname,
            const char * line,
            const char * function,
            uint64_t cycle,
            const char * fmt,
             ...)
```

4.43.2.4 m_eng_log_return_()

```
void m_eng_log_return_ (
            const char * fname,
            const char * line,
            const char * function,
            uint64_t cycle)
```

4.43.2.5 m_eng_log_return_err()

```
void m_eng_log_return_err (
            const char * fname,
            const char * line,
            const char * function,
            int error_code,
            uint64_t cycle)
```

4.43.2.6 m_eng_log_return_int()

```
void m_eng_log_return_int (
            const char * fname,
            const char * line,
            const char * function,
            int val,
            uint64_t cycle)
```

4.43.2.7 m_eng_log_return_ptr()

```
void m_eng_log_return_ptr (
            const char * fname,
            const char * line,
            const char * function,
            void * ptr,
            uint64_t cycle)
```

4.43.2.8 m_eng_print_flush_log()

```
void m_eng_print_flush_log ()
```

4.43.2.9 m_eng_print_log()

```
void m_eng_print_log ()
```

4.43.2.10 m_eng_trace_log_begin()

```
void m_eng_trace_log_begin (
            const char * fname,
            const char * line,
            const char * function,
            int local_trace_depth,
            uint64_t cycle)
```

4.43.2.11 m_eng_trace_log_return()

```
void m_eng_trace_log_return (
            const char * fname,
            const char * line,
            const char * function,
            int local_trace_depth,
            uint64_t cycle)
```

4.44 m_eng_logging.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_LOGGING_H_
00002 #define M_ENG_LOGGING_H_
00003
00004 #define M_ENG_TRACE_FUNCTION_ENTER      0
00005 #define M_ENG_TRACE_FUNCTION_RETURN     1
00006 #define M_ENG_LOG_ENTRY_ERROR           2
00007 #define M_ENG_LOG_ENTRY_RETURN_ERR      3
00008 #define M_ENG_LOG_ENTRY_RETURN_PTR      4
00009 #define M_ENG_LOG_ENTRY_RETURN_INT      5
00010 #define M_ENG_LOG_ENTRY_RETURN          6
00011 #define M_ENG_LOG_ENTRY_MESSAGE         7
00012
00013 typedef struct
00014 {
00015     int type;
```

```
00016     const char *file_name;
00017     const char *line;
00018     const char *data_type;
00019     const char *function;
00020     const char *message;
00021     uint64_t cycle;
00022     uint32_t data;
00023     uint32_t trace_depth;
00024 } m_eng_log_entry;
00025
00026 typedef struct
00027 {
00028     const char *function_name;
00029     uint64_t open_cycle;
00030     uint64_t calls;
00031     uint64_t total_cycles;
00032     uint64_t peak_cycles;
00033     double ra_cycles;
00034 } m_eng_profiler_entry;
00035
00036 #define M_ENG_LOG_ERRORS    0b00001
00037 #define M_ENG_LOG_TRACE     0b00010
00038 #define M_ENG_LOG_RETURNS   0b00100
00039 #define M_ENG_LOG_SPIKES    0b01000
00040
00041 #define M_ENG_LOG_EVERYTHING (M_ENG_LOG_ERRORS | M_ENG_LOG_TRACE | M_ENG_LOG_RETURNS |
      M_ENG_LOG_SPIKES)
00042
00043 #define STR(x) #x
00044 #define XSTR(x) (STR(x))
00045
00046 #ifndef M_SIMULATED
00047 #define M_ENG_LOG_LEVEL (M_ENG_LOG_ERRORS)
00048 //#define M_ENG_PROFILER
00049 #else
00050 #define M_ENG_LOG_LEVEL M_ENG_LOG_ERRORS
00051 #define M_ENG_PROFILER
00052 #endif
00053
00054 #define M_ENG_LOG_INDENT_TRACE
00055
00056 #ifdef  M_ENG_LOG_LEVEL
00057   #if M_ENG_LOG_LEVEL & M_ENG_LOG_TRACE
00058     #define M_ENG_TRACE_LOG_ENTRY()  m_eng_trace_log_begin (FNAME, XSTR(__LINE__), __func__,
      trace_depth, current_cycle());
00059     #define M_ENG_TRACE_LOG_RETURN() m_eng_trace_log_return(FNAME, XSTR(__LINE__), __func__,
      trace_depth, current_cycle());
00060   #endif
00061
00062   #if M_ENG_LOG_LEVEL & M_ENG_LOG_RETURNS
00063     #define M_ENG_LOG_RETURN_ERR_CODE(x)    m_eng_log_return_err(FNAME, XSTR(__LINE__), __func__, x,
      current_cycle());
00064     #define M_ENG_LOG_RETURN_PTR(x)         m_eng_log_return_ptr(FNAME, XSTR(__LINE__), __func__, x,
      current_cycle());
00065     #define M_ENG_LOG_RETURN_INT(x)         m_eng_log_return_int(FNAME, XSTR(__LINE__), __func__, x,
      current_cycle());
00066     #define M_ENG_LOG_RETURN(x)             m_eng_log_return_   (FNAME, XSTR(__LINE__), __func__,
      current_cycle());
00067   #endif
00068
00069   #if M_ENG_LOG_LEVEL & M_ENG_LOG_ERRORS
00070     #define M_ENG_LOG_ERROR(x) \
00071       if (x != NO_ERROR) \
00072         m_eng_log_error_code(FNAME, XSTR(__LINE__), __func__, x, current_cycle());
00073   #endif
00074 #endif
00075
00076 #ifdef M_ENG_PROFILER
00077   #define M_ENG_PROFILER_LOG_ENTRY()  m_eng_profiler_log_entry ( __func__);
00078   #define M_ENG_PROFILER_LOG_RETURN() m_eng_profiler_log_return( __func__, current_cycle());
00079 #else
00080   #define M_ENG_PROFILER_LOG_ENTRY()
00081   #define M_ENG_PROFILER_LOG_RETURN()
00082 #endif
00083
00084 #ifndef M_ENG_TRACE_LOG_ENTRY
00085 #define M_ENG_TRACE_LOG_ENTRY()
00086 #endif
00087
00088 #ifndef M_ENG_TRACE_LOG_RETURN
00089 #define M_ENG_TRACE_LOG_RETURN()
00090 #endif
00091
00092 #ifndef M_ENG_LOG_RETURN_ERR_CODE
00093 #define M_ENG_LOG_RETURN_ERR_CODE(x)
00094 #endif
00095
```

```
00096 #ifndef M_ENG_LOG_RETURN_PTR
00097 #define M_ENG_LOG_RETURN_PTR(x)
00098 #endif
00099
00100 #ifndef M_ENG_LOG_RETURN_INT
00101 #define M_ENG_LOG_RETURN_INT(x)
00102 #endif
00103
00104 #ifndef M_ENG_LOG_RETURN
00105 #define M_ENG_LOG_RETURN(x)
00106 #endif
00107
00108 #ifndef M_ENG_LOG_ERROR
00109 #define M_ENG_LOG_ERROR(x)
00110 #endif
00111
00112 void m_eng_log_message(const char *fname, const char *line, const char *function, uint64_t cycle,
      const char *fmt, ...);
00113
00114
00115 #define FUNCTION_START()        M_ENG_PROFILER_LOG_ENTRY();  M_ENG_TRACE_LOG_ENTRY();
00116 #define RETURN_VOID()           M_ENG_PROFILER_LOG_RETURN(); M_ENG_TRACE_LOG_RETURN(); return;
00117 #define RETURN_ERR_CODE(x)      M_ENG_PROFILER_LOG_RETURN(); M_ENG_LOG_ERROR(x);
      M_ENG_TRACE_LOG_RETURN(); M_ENG_LOG_RETURN_ERR_CODE(x); return x;
00118 #define RETURN_NEG_ERR_CODE(x)  M_ENG_PROFILER_LOG_RETURN(); M_ENG_LOG_ERROR(x);
      M_ENG_TRACE_LOG_RETURN(); M_ENG_LOG_RETURN_ERR_CODE(x); return -x;
00119 #define RETURN_PTR(x)           M_ENG_PROFILER_LOG_RETURN(); M_ENG_TRACE_LOG_RETURN();
      M_ENG_LOG_RETURN_PTR(x); return x;
00120 #define RETURN_INT(x)           M_ENG_PROFILER_LOG_RETURN(); M_ENG_TRACE_LOG_RETURN();
      M_ENG_LOG_RETURN_INT(x); return x;
00121 #define RETURN(x)               M_ENG_PROFILER_LOG_RETURN(); M_ENG_TRACE_LOG_RETURN();
      M_ENG_LOG_RETURN(); return x;
00122
00123 void m_eng_log(const char *fmt, ...);
00124 void m_eng_print_log();
00125 void m_eng_print_flush_log();
00126
00127 void m_eng_trace_log_begin (const char *fname, const char *line, const char *function, int
      local_trace_depth, uint64_t cycle);
00128 void m_eng_trace_log_return (const char *fname, const char *line, const char *function, int
      local_trace_depth, uint64_t cycle);
00129 void m_eng_log_error_code   (const char *fname, const char *line, const char *function, int
      error_code, uint64_t cycle);
00130 void m_eng_log_return_err   (const char *fname, const char *line, const char *function, int
      error_code, uint64_t cycle);
00131 void m_eng_log_return_ptr   (const char *fname, const char *line, const char *function, void *ptr,
      uint64_t cycle);
00132 void m_eng_log_return_int   (const char *fname, const char *line, const char *function, int val,
      uint64_t cycle);
00133 void m_eng_log_return_     (const char *fname, const char *line, const char *function, uint64_t
      cycle);
00134
00135 #define M_LOG(fmt, ...) m_eng_log_message(FNAME, XSTR(__LINE__), __func__, current_cycle(), fmt,
      ##__VA_ARGS__);
00136 #define M_LOG_ERROR(x) m_eng_log_error_code(FNAME, XSTR(__LINE__), __func__, x, current_cycle());
00137
00138 #ifdef M_ENG_PROFILER
00139 void m_eng_init_profiler();
00140 void m_eng_profiler_log_entry (const char *function_name);
00141 void m_eng_profiler_log_return(const char *function_name, uint64_t cycle);
00142
00143 void m_eng_profiler_print();
00144 #endif
00145
00146 #define CYCLES_TO_SECONDS(x) ((double)(x) * 1.66666667e-9)
00147 #define SECONDS_TO_CYCLES(x) ((double)(x) / 1.66666667e-9)
00148
00149 #endif
```

4.45 m_eng_low_end_compressor.h File Reference

Data Structures

- struct m_eng_low_end_compressor_str

Functions

- int init_low_end_compressor_str (m_eng_low_end_compressor_str *str)
- int reconfigure_low_end_compressor (void *data_struct)
- int calc_low_end_compressor (void *data_struct, float *dest, float *src, int n_samples)

4.45.1 Function Documentation

4.45.1.1 calc_low_end_compressor()

```
int calc_low_end_compressor (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.45.1.2 init_low_end_compressor_str()

```
int init_low_end_compressor_str (
            m_eng_low_end_compressor_str * str)
```

4.45.1.3 reconfigure_low_end_compressor()

```
int reconfigure_low_end_compressor (
            void * data_struct)
```

4.46 m_eng_low_end_compressor.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_LOW_END_COMPRESSOR_H_
00002 #define M_ENG_LOW_END_COMPRESSOR_H_
00003
00004 typedef struct
00005 {
00006     m_eng_compressor_str bass_comp;
00007     m_eng_compressor_str mids_comp;
00008
00009     m_lr_low_pass_filter_str low_pass;
00010     m_lr_low_pass_filter_str mid_pass;
00011
00012 } m_eng_low_end_compressor_str;
00013
00014 int init_low_end_compressor_str(m_eng_low_end_compressor_str *str);
00015 int reconfigure_low_end_compressor(void *data_struct);
00016 int calc_low_end_compressor(void *data_struct, float *dest, float *src, int n_samples);
00017
00018 #endif
```

4.47 m_eng_memcpy_audio.h File Reference

Functions

- void memcpy_tointerleaveLR (int16_t *dst, const int16_t *srcL, const int16_t *srcR)
- void memcpy_tointerleaveL (int16_t *dst, const int16_t *srcL)
- void memcpy_tointerleaveR (int16_t *dst, const int16_t *srcR)
- void memcpy_tointerleaveQuad (int16_t *dst, const int16_t *src1, const int16_t *src2, const int16_t *src3, const int16_t *src4)

4.47.1 Function Documentation

4.47.1.1 memcpy_tointerleaveL()

```
void memcpy_tointerleaveL (
            int16_t * dst,
            const int16_t * srcL)
```

4.47.1.2 memcpy_tointerleaveLR()

```
void memcpy_tointerleaveLR (
            int16_t * dst,
            const int16_t * srcL,
            const int16_t * srcR)
```

4.47.1.3 memcpy_tointerleaveQuad()

```
void memcpy_tointerleaveQuad (
            int16_t * dst,
            const int16_t * src1,
            const int16_t * src2,
            const int16_t * src3,
            const int16_t * src4)
```

4.47.1.4 memcpy_tointerleaveR()

```
void memcpy_tointerleaveR (
            int16_t * dst,
            const int16_t * srcR)
```

4.48 m_eng_memcpy_audio.h

Go to the documentation of this file.

```
00001 /* Teensyduino m_eng_Audio Memcpy
00002  * Copyright (c) 2016 Frank Bösing
00003  *
00004  * Permission is hereby granted, free of charge, to any person obtaining
00005  * a copy of this software and associated documentation files (the
00006  * "Software"), to deal in the Software without restriction, including
00007  * without limitation the rights to use, copy, modify, merge, publish,
00008  * distribute, sublicense, and/or sell copies of the Software, and to
00009  * permit persons to whom the Software is furnished to do so, subject to
00010  * the following conditions:
00011  *
00012  * 1. The above copyright notice and this permission notice shall be
00013  * included in all copies or substantial portions of the Software.
00014  *
00015  * 2. If the Software is incorporated into a build system that allows
00016  * selection among a list of target devices, then similar target
00017  * devices manufactured by PJRC.COM must be included in the list of
00018  * target devices and selectable in the same manner.
00019  *
00020  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
00021  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
00022  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
00023  * NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS
00024  * BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN
00025  * ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
```

```
00026  * CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00027  * SOFTWARE.
00028  */
00029
00030 #ifndef memcpy_audio_h_
00031 #define memcpy_audio_h_
00032
00033 #ifdef __cplusplus
00034 extern "C" {
00035 #endif
00036 void memcpy_tointerleaveLR(int16_t *dst, const int16_t *srcL, const int16_t *srcR);
00037 void memcpy_tointerleaveL(int16_t *dst, const int16_t *srcL);
00038 void memcpy_tointerleaveR(int16_t *dst, const int16_t *srcR);
00039 void memcpy_tointerleaveQuad(int16_t *dst, const int16_t *src1, const int16_t *src2,
00040     const int16_t *src3, const int16_t *src4);
00041 #ifdef __cplusplus
00042 }
00043 #endif
00044
00045
00046 #endif
```

4.49 m_eng_mempool.h File Reference

Macros

- #define MAX_AUDIO_MEMORY 229376
- #define MEM_SIZE 48
- #define M_BUFFER_POOL_SIZE 128

Functions

- float * allocate_buffer ()
- void release_buffer (float *buffer)
- void init_mem_pools ()
- void print_mempool_info ()

Variables

- float zero_buffer [AUDIO_BLOCK_SAMPLES]
- float sink_buffer [AUDIO_BLOCK_SAMPLES]

4.49.1 Macro Definition Documentation

4.49.1.1 M_BUFFER_POOL_SIZE

```
#define M_BUFFER_POOL_SIZE 128
```

4.49.1.2 MAX_AUDIO_MEMORY

```
#define MAX_AUDIO_MEMORY 229376
```

4.49.1.3 MEM_SIZE

```
#define MEM_SIZE 48
```

4.49.2 Function Documentation

4.49.2.1 allocate_buffer()

```
float * allocate_buffer ()
```

4.49.2.2 init_mem_pools()

```
void init_mem_pools ()
```

4.49.2.3 print_mempool_info()

```
void print_mempool_info ()
```

4.49.2.4 release_buffer()

```
void release_buffer (
            float * buffer)
```

4.49.3 Variable Documentation

4.49.3.1 sink_buffer

```
float sink_buffer[AUDIO_BLOCK_SAMPLES]  [extern]
```

4.49.3.2 zero_buffer

```
float zero_buffer[AUDIO_BLOCK_SAMPLES]  [extern]
```

4.50 m_eng_mempool.h

Go to the documentation of this file.

```
00001 #ifndef M_MEMPOOL_H_
00002 #define M_MEMPOOL_H_
00003
00004 #define MAX_AUDIO_MEMORY 229376
00005
00006 #define MEM_SIZE 48
00007
00008 #ifdef M_SIMULATED
00009 #define M_BUFFER_POOL_SIZE 8192
00010 #else
00011 #define M_BUFFER_POOL_SIZE 128
00012 #endif
00013
00014 float *allocate_buffer();
00015 void release_buffer(float *buffer);
00016
00017 void init_mem_pools();
00018
00019 extern float zero_buffer[AUDIO_BLOCK_SAMPLES];
00020 extern float sink_buffer[AUDIO_BLOCK_SAMPLES];
00021
00022 void print_mempool_info();
00023
00024 #endif
```

4.51 m_eng_noise_suppressor.h File Reference

Data Structures

- struct m_eng_noise_suppressor_str

Functions

- int reconfigure_noise_suppressor (void *data_struct)
- int calc_noise_suppressor (void *data_struct, float *dest, float *src, int n_samples)
- int init_noise_suppressor_str (m_eng_noise_suppressor_str *str)

4.51.1 Function Documentation

4.51.1.1 calc_noise_suppressor()

```
int calc_noise_suppressor (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.51.1.2 init_noise_suppressor_str()

```
int init_noise_suppressor_str (
            m_eng_noise_suppressor_str * str)
```

4.51.1.3 reconfigure_noise_suppressor()

```
int reconfigure_noise_suppressor (
            void * data_struct)
```

4.52 m_eng_noise_suppressor.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_NOISE_SUPRESSOR_H_
00002 #define M_ENG_NOISE_SUPRESSOR_H_
00003
00004 typedef struct
00005 {
00006     m_parameter threshold;
00007     m_parameter ratio;
00008     m_parameter max_reduction;
00009     float e_final;
00010
00011     int r;
00012 } m_eng_noise_suppressor_str;
00013
00014 int reconfigure_noise_suppressor(void *data_struct);
00015 int calc_noise_suppressor(void *data_struct, float *dest, float *src, int n_samples);
00016 int init_noise_suppressor_str(m_eng_noise_suppressor_str *str);
00017
00018 #endif
```

4.53 m_eng_parameter.h File Reference

```
#include <stddef.h>
#include "m_parameter.h"
```

Macros

- #define PARAM_NAM_ENG_MAX_LEN 16
- #define PARAMETER_UPDATE_INSTANT 0
- #define PARAMETER_UPDATE_MONOBLOCK_LINEAR 1
- #define PARAMETER_UPDATE_BIBLOCK_LINEAR 2
- #define PARAMETER_UPDATE_QUADBLOCK_LINEAR 3
- #define DEFAULT_MAX_JUMP 0.01

Functions

- void init_parameter (m_parameter *param, float initial, float min, float max, float max_jump, int scale)
- int init_setting (m_setting *setting, int16_t initial)

4.53.1 Macro Definition Documentation

4.53.1.1 DEFAULT_MAX_JUMP

```
#define DEFAULT_MAX_JUMP 0.01
```

4.53.1.2 PARAM_NAM_ENG_MAX_LEN

```
#define PARAM_NAM_ENG_MAX_LEN 16
```

4.53.1.3 PARAMETER_UPDATE_BIBLOCK_LINEAR

```
#define PARAMETER_UPDATE_BIBLOCK_LINEAR 2
```

4.53.1.4 PARAMETER_UPDATE_INSTANT

```
#define PARAMETER_UPDATE_INSTANT 0
```

4.53.1.5 PARAMETER_UPDATE_MONOBLOCK_LINEAR

```
#define PARAMETER_UPDATE_MONOBLOCK_LINEAR 1
```

4.53.1.6 PARAMETER_UPDATE_QUADBLOCK_LINEAR

```
#define PARAMETER_UPDATE_QUADBLOCK_LINEAR 3
```

4.53.2 Function Documentation

4.53.2.1 init_parameter()

```
void init_parameter (
            m_parameter * param,
            float initial,
            float min,
            float max,
            float max_jump,
            int scale)
```

4.53.2.2 init_setting()

```
int init_setting (
            m_setting * setting,
            int16_t initial)
```

4.54 m_eng_parameter.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_PARAMETER_H_
00002 #define M_ENG_PARAMETER_H_
00003
00004 #ifdef __cplusplus
00005 #include <cstddef>
00006 #else
00007 #include <stddef.h>
00008 #endif
00009
00010 #define PARAM_NAM_ENG_MAX_LEN 16
00011
00012 #define PARAMETER_UPDATE_INSTANT            0
00013 #define PARAMETER_UPDATE_MONOBLOCK_LINEAR   1
00014 #define PARAMETER_UPDATE_BIBLOCK_LINEAR     2
00015 #define PARAMETER_UPDATE_QUADBLOCK_LINEAR   3
00016
00017 #define DEFAULT_MAX_JUMP 0.01
00018
00019 #include "m_parameter.h"
00020
00021 void init_parameter(m_parameter *param, float initial, float min, float max, float max_jump, int
      scale);
00022 int init_setting(m_setting *setting, int16_t initial);
00023
00024 #endif
```

4.55 m_eng_pass_filter.h File Reference

Data Structures

- struct m_eng_low_pass_filter_str
- struct m_eng_high_pass_filter_str
- struct m_eng_band_pass_filter_str

Functions

- int init_low_pass_filter_str (m_eng_low_pass_filter_str ∗str)
- int reconfigure_low_pass_filter (void ∗data_struct)
- int calc_low_pass_filter (void ∗data_struct, float ∗dest, float ∗src, int n_samples)
- int init_high_pass_filter_str (m_eng_high_pass_filter_str ∗str)
- int reconfigure_high_pass_filter (void ∗data_struct)
- int calc_high_pass_filter (void ∗data_struct, float ∗dest, float ∗src, int n_samples)
- int init_band_pass_filter_str (m_eng_band_pass_filter_str ∗str)
- int reconfigure_band_pass_filter (void ∗data_struct)
- int calc_band_pass_filter (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.55.1 Function Documentation

4.55.1.1 calc_band_pass_filter()

```
int calc_band_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.55.1.2 calc_high_pass_filter()

```
int calc_high_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.55.1.3 calc_low_pass_filter()

```
int calc_low_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.55.1.4 init_band_pass_filter_str()

```
int init_band_pass_filter_str (
            m_eng_band_pass_filter_str * str)
```

4.55.1.5 init_high_pass_filter_str()

```
int init_high_pass_filter_str (
            m_eng_high_pass_filter_str * str)
```

4.55.1.6 init_low_pass_filter_str()

```
int init_low_pass_filter_str (
            m_eng_low_pass_filter_str * str)
```

4.55.1.7 reconfigure_band_pass_filter()

```
int reconfigure_band_pass_filter (
            void * data_struct)
```

4.55.1.8 reconfigure_high_pass_filter()

```
int reconfigure_high_pass_filter (
            void * data_struct)
```

4.55.1.9 reconfigure_low_pass_filter()

```
int reconfigure_low_pass_filter (
            void * data_struct)
```

4.56 m_eng_pass_filter.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_LOW_PASS_H_
00002 #define M_ENG_LOW_PASS_H_
00003
00004 typedef struct
00005 {
00006     m_parameter cutoff_frequency;
00007
00008     float a0, a1, a2, a3, a4;
00009     float x1, x2, y1, y2;
00010 } m_eng_low_pass_filter_str;
00011
00012 int init_low_pass_filter_str(m_eng_low_pass_filter_str *str);
00013 int reconfigure_low_pass_filter(void *data_struct);
00014 int calc_low_pass_filter(void *data_struct, float *dest, float *src, int n_samples);
00015
00016 typedef struct
00017 {
00018     m_parameter cutoff_frequency;
00019
00020     float a0, a1, a2, a3, a4;
00021     float x1, x2, y1, y2;
00022 } m_eng_high_pass_filter_str;
00023
00024 int init_high_pass_filter_str(m_eng_high_pass_filter_str *str);
00025 int reconfigure_high_pass_filter(void *data_struct);
00026 int calc_high_pass_filter(void *data_struct, float *dest, float *src, int n_samples);
00027
00028 typedef struct
00029 {
00030     m_parameter center;
00031     m_parameter bandwidth;
00032
00033     float a0, a1, a2, a3, a4;
00034     float x1, x2, y1, y2;
00035 } m_eng_band_pass_filter_str;
00036
00037 int init_band_pass_filter_str(m_eng_band_pass_filter_str *str);
00038 int reconfigure_band_pass_filter(void *data_struct);
00039 int calc_band_pass_filter(void *data_struct, float *dest, float *src, int n_samples);
00040
00041 #endif
00042
```

4.57 m_eng_percussifier.h File Reference

Data Structures

- struct m_eng_percussifier_str

Macros

- #define PERCUSSIFIER_MUTE 0
- #define PERCUSSIFIER_FADE_IN 1
- #define PERCUSSIFIER_HOLD 2
- #define PERCUSSIFIER_FADE_OUT 3
- #define PERCUSSIFIER_REFRACTORY 4

Functions

- int init_percussifier_str (m_eng_percussifier_str *str)
- int reconfigure_percussifier (void *data_struct)
- int calc_percussifier (void *data_struct, float *dest, float *src, int n_samples)

4.57.1 Macro Definition Documentation

4.57.1.1 PERCUSSIFIER_FADE_IN

```
#define PERCUSSIFIER_FADE_IN 1
```

4.57.1.2 PERCUSSIFIER_FADE_OUT

```
#define PERCUSSIFIER_FADE_OUT 3
```

4.57.1.3 PERCUSSIFIER_HOLD

```
#define PERCUSSIFIER_HOLD 2
```

4.57.1.4 PERCUSSIFIER_MUTE

```
#define PERCUSSIFIER_MUTE 0
```

4.57.1.5 PERCUSSIFIER_REFRACTORY

```
#define PERCUSSIFIER_REFRACTORY 4
```

4.57.2 Function Documentation

4.57.2.1 calc_percussifier()

```
int calc_percussifier (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.57.2.2 init_percussifier_str()

```
int init_percussifier_str (
            m_eng_percussifier_str * str)
```

4.57.2.3 reconfigure_percussifier()

```
int reconfigure_percussifier (
            void * data_struct)
```

4.58 m_eng_percussifier.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_PERCUSSIFIER_H_
00002 #define M_ENG_PERCUSSIFIER_H_
00003
00004 #define PERCUSSIFIER_MUTE       0
00005 #define PERCUSSIFIER_FADE_IN    1
00006 #define PERCUSSIFIER_HOLD       2
00007 #define PERCUSSIFIER_FADE_OUT   3
00008 #define PERCUSSIFIER_REFRACTORY 4
00009
00010 typedef struct
00011 {
00012     m_parameter tempo;
00013     m_parameter note;
00014     m_parameter trigger_threshold;
00015     m_parameter arm_threshold;
00016     m_parameter refractory_period;
00017     m_parameter fade_in;
00018     m_parameter fade_out;
00019
00020     int state;
00021     int fade_in_samples;
00022     int hold_samples;
00023     int refractory_samples;
00024
00025     int timer;
00026
00027     float decay_rate;
00028     float rms_short;
00029     float rms_long;
00030
00031     float alpha_short;
00032     float alpha_long;
00033
00034     float gain;
00035     float fade_alpha;
00036
00037     int r;
00038 } m_eng_percussifier_str;
00039
00040 int init_percussifier_str(m_eng_percussifier_str *str);
00041 int reconfigure_percussifier(void *data_struct);
00042 int calc_percussifier(void *data_struct, float *dest, float *src, int n_samples);
00043
00044 #endif
```

4.59 m_eng_pipeline.h File Reference

```
#include "m_transformer.h"
#include "m_pipeline.h"
```

Macros

- #define INITIAL_TRANSFORMER_ARRAY_LENGTH 8

Functions

- int init_pipeline (m_pipeline *pipeline)
- int compute_pipeline (m_pipeline *pipeline, float *dest, float *src)
- int pipeline_expand_transformer_array (m_pipeline *pipeline)
- int pipeline_expand_transformer_array_to (m_pipeline *pipeline, int n)
- int pipeline_append_transformer (m_pipeline *pipeline, m_transformer *trans)
- int pipeline_remove_transformer (m_pipeline *pipeline, uint16_t tid)
- int pipeline_insert_transformer (m_pipeline *pipeline, m_transformer *trans, int pos)
- int pipeline_prepend_transformer (m_pipeline *pipeline, m_transformer *trans)
- int pipeline_append_transformer_type (m_pipeline *pipeline, uint16_t type)
- int pipeline_insert_transformer_type (m_pipeline *pipeline, uint16_t type, uint16_t pos)
- int pipeline_prepend_transformer_type (m_pipeline *pipeline, uint16_t type)
- int pipeline_change_transformer_setting (m_pipeline *pipeline, uint16_t tid, uint16_t sid, int16_t new_val)
- int pipeline_get_transformer_position (m_pipeline *pipeline, uint16_t id)
- m_transformer * pipeline_get_transformer_by_id (m_pipeline *pipeline, uint16_t id)
- int pipeline_move_transformer (m_pipeline *pipeline, uint16_t id, int positio)
- int pipeline_update_transition_policy (m_pipeline *pipeline)
- int pipeline_swap_transformers (m_pipeline *pipeline, uint16_t id1, uint16_t id2)
- int pipeline_valid (m_pipeline *pipeline)
- int pipeline_compare (m_pipeline *pipeline_a, m_pipeline *pipeline_b)
- int gut_pipeline (m_pipeline *pipeline)
- int clone_pipeline (m_pipeline **dest, m_pipeline *src)
- int pipeline_clone_transformer_into_position (m_pipeline *pipeline, m_transformer *transformer, int pot)

4.59.1 Macro Definition Documentation

4.59.1.1 INITIAL_TRANSFORMER_ARRAY_LENGTH

```
#define INITIAL_TRANSFORMER_ARRAY_LENGTH 8
```

4.59.2 Function Documentation

4.59.2.1 clone_pipeline()

```
int clone_pipeline (
            m_pipeline ** dest,
            m_pipeline * src)
```

4.59.2.2 compute_pipeline()

```
int compute_pipeline (
            m_pipeline * pipeline,
            float * dest,
            float * src)
```

4.59.2.3 gut_pipeline()

```
int gut_pipeline (
            m_pipeline * pipeline)
```

4.59.2.4 init_pipeline()

```
int init_pipeline (
            m_pipeline * pipeline)
```

4.59.2.5 pipeline_append_transformer()

```
int pipeline_append_transformer (
            m_pipeline * pipeline,
            m_transformer * trans)
```

4.59.2.6 pipeline_append_transformer_type()

```
int pipeline_append_transformer_type (
            m_pipeline * pipeline,
            uint16_t type)
```

4.59.2.7 pipeline_change_transformer_setting()

```
int pipeline_change_transformer_setting (
            m_pipeline * pipeline,
            uint16_t tid,
            uint16_t sid,
            int16_t new_val)
```

4.59.2.8 pipeline_clone_transformer_into_position()

```
int pipeline_clone_transformer_into_position (
            m_pipeline * pipeline,
            m_transformer * transformer,
            int pot)
```

4.59.2.9 pipeline_compare()

```
int pipeline_compare (
            m_pipeline * pipeline_a,
            m_pipeline * pipeline_b)
```

4.59.2.10 pipeline_expand_transformer_array()

```
int pipeline_expand_transformer_array (
            m_pipeline * pipeline)
```

4.59.2.11 pipeline_expand_transformer_array_to()

```
int pipeline_expand_transformer_array_to (
            m_pipeline * pipeline,
            int n)
```

4.59.2.12 pipeline_get_transformer_by_id()

```
m_transformer * pipeline_get_transformer_by_id (
            m_pipeline * pipeline,
            uint16_t id)
```

4.59.2.13 pipeline_get_transformer_position()

```
int pipeline_get_transformer_position (
            m_pipeline * pipeline,
            uint16_t id)
```

4.59.2.14 pipeline_insert_transformer()

```
int pipeline_insert_transformer (
            m_pipeline * pipeline,
            m_transformer * trans,
            int pos)
```

4.59.2.15 pipeline_insert_transformer_type()

```
int pipeline_insert_transformer_type (
            m_pipeline * pipeline,
            uint16_t type,
            uint16_t pos)
```

4.59.2.16 pipeline_move_transformer()

```
int pipeline_move_transformer (
            m_pipeline * pipeline,
            uint16_t id,
            int positio)
```

4.59.2.17 pipeline_prepend_transformer()

```
int pipeline_prepend_transformer (
            m_pipeline * pipeline,
            m_transformer * trans)
```

4.59.2.18 pipeline_prepend_transformer_type()

```
int pipeline_prepend_transformer_type (
            m_pipeline * pipeline,
            uint16_t type)
```

4.59.2.19 pipeline_remove_transformer()

```
int pipeline_remove_transformer (
            m_pipeline * pipeline,
            uint16_t tid)
```

4.59.2.20 pipeline_swap_transformers()

```
int pipeline_swap_transformers (
            m_pipeline * pipeline,
            uint16_t id1,
            uint16_t id2)
```

4.59.2.21 pipeline_update_transition_policy()

```
int pipeline_update_transition_policy (
            m_pipeline * pipeline)
```

4.59.2.22 pipeline_valid()

```
int pipeline_valid (
            m_pipeline * pipeline)
```

4.60 m_eng_pipeline.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_PIPELINE_H_
00002 #define M_ENG_PIPELINE_H_
00003
00004 #include "m_transformer.h"
00005
00006 #define INITIAL_TRANSFORMER_ARRAY_LENGTH 8
00007
00008 #include "m_pipeline.h"
00009
00010 int init_pipeline(m_pipeline *pipeline);
00011 int compute_pipeline(m_pipeline *pipeline, float *dest, float *src);
00012
00013 int pipeline_expand_transformer_array(m_pipeline *pipeline);
00014 int pipeline_expand_transformer_array_to(m_pipeline *pipeline, int n);
00015 int pipeline_append_transformer(m_pipeline *pipeline, m_transformer *trans);
00016 int pipeline_remove_transformer(m_pipeline *pipeline, uint16_t tid);
00017 int pipeline_insert_transformer(m_pipeline *pipeline, m_transformer *trans, int pos);
00018 int pipeline_prepend_transformer(m_pipeline *pipeline, m_transformer *trans);
00019 int pipeline_append_transformer_type(m_pipeline *pipeline, uint16_t type);
00020 int pipeline_insert_transformer_type(m_pipeline *pipeline, uint16_t type, uint16_t pos);
00021 int pipeline_prepend_transformer_type(m_pipeline *pipeline, uint16_t type);
00022 int pipeline_change_transformer_setting(m_pipeline *pipeline, uint16_t tid, uint16_t sid, int16_t
      new_val);
00023
00024 int pipeline_get_transformer_position(m_pipeline *pipeline, uint16_t id);
00025 m_transformer *pipeline_get_transformer_by_id(m_pipeline *pipeline, uint16_t id);
00026
00027 int pipeline_move_transformer(m_pipeline *pipeline, uint16_t id, int positio);
00028
00029 int pipeline_update_transition_policy(m_pipeline *pipeline);
00030
00031 int pipeline_swap_transformers(m_pipeline *pipeline, uint16_t id1, uint16_t id2);
00032
00033 int pipeline_valid     (m_pipeline *pipeline);
00034 int pipeline_compare(m_pipeline *pipeline_a, m_pipeline *pipeline_b);
00035 int gut_pipeline     (m_pipeline *pipeline);
00036 int clone_pipeline     (m_pipeline **dest, m_pipeline *src);
00037
00038 int pipeline_clone_transformer_into_position(m_pipeline *pipeline, m_transformer *transformer, int
      pot);
00039
00040 #endif
```

4.61 m_eng_pipeline_mod.h File Reference

Data Structures

- struct m_pipeline_mod

Macros

- #define PIPELINE_MOD_APPEND_TRANSFORMER 0
- #define PIPELINE_MOD_MOVE_TRANSFORMER 1
- #define PIPELINE_MOD_REMOVE_TRANSFORMER 2
- #define PIPELINE_MOD_CHANGE_TRANSFORMER_SETTING 3

Functions

- int init_pipeline_mod (m_pipeline_mod *mod)
- m_pipeline_mod create_pipeline_mod_append_transformer (uint16_t type)
- m_pipeline_mod create_pipeline_mod_move_transformer (uint16_t tid, uint16_t position)
- m_pipeline_mod create_pipeline_mod_remove_transformer (uint16_t tid)
- m_pipeline_mod create_pipeline_mod_change_transformer_setting (uint16_t tid, uint16_t setting_id, int16↩ _t new_value)
- int apply_pipeline_mod (m_pipeline *pipeline, m_pipeline_mod mod, int *err_code)
- const char * pipeline_mod_type_string (int type)
- DECLARE_LINKED_LIST (m_pipeline_mod)

4.61.1 Macro Definition Documentation

4.61.1.1 PIPELINE_MOD_APPEND_TRANSFORMER

```
#define PIPELINE_MOD_APPEND_TRANSFORMER 0
```

4.61.1.2 PIPELINE_MOD_CHANGE_TRANSFORMER_SETTING

```
#define PIPELINE_MOD_CHANGE_TRANSFORMER_SETTING 3
```

4.61.1.3 PIPELINE_MOD_MOVE_TRANSFORMER

```
#define PIPELINE_MOD_MOVE_TRANSFORMER 1
```

4.61.1.4 PIPELINE_MOD_REMOVE_TRANSFORMER

```
#define PIPELINE_MOD_REMOVE_TRANSFORMER 2
```

4.61.2 Function Documentation

4.61.2.1 apply_pipeline_mod()

```
int apply_pipeline_mod (
            m_pipeline * pipeline,
            m_pipeline_mod mod,
            int * err_code)
```

4.61.2.2 create_pipeline_mod_append_transformer()

```
m_pipeline_mod create_pipeline_mod_append_transformer (
            uint16_t type)
```

4.61.2.3 create_pipeline_mod_change_transformer_setting()

```
m_pipeline_mod create_pipeline_mod_change_transformer_setting (
            uint16_t tid,
            uint16_t setting_id,
            int16_t new_value)
```

4.61.2.4 create_pipeline_mod_move_transformer()

```
m_pipeline_mod create_pipeline_mod_move_transformer (
            uint16_t tid,
            uint16_t position)
```

4.61.2.5 create_pipeline_mod_remove_transformer()

```
m_pipeline_mod create_pipeline_mod_remove_transformer (
            uint16_t tid)
```

4.61.2.6 DECLARE_LINKED_LIST()

```
DECLARE_LINKED_LIST (
            m_pipeline_mod )
```

4.61.2.7 init_pipeline_mod()

```
int init_pipeline_mod (
            m_pipeline_mod * mod)
```

4.61.2.8 pipeline_mod_type_string()

```
const char * pipeline_mod_type_string (
            int type)
```

4.62 m_eng_pipeline_mod.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_PIPELINE_MOD_H_
00002 #define M_ENG_PIPELINE_MOD_H_
00003
00004 #define PIPELINE_MOD_APPEND_TRANSFORMER         0
00005 #define PIPELINE_MOD_MOVE_TRANSFORMER           1
00006 #define PIPELINE_MOD_REMOVE_TRANSFORMER         2
00007 #define PIPELINE_MOD_CHANGE_TRANSFORMER_SETTING 3
00008
00009 typedef struct m_pipeline_mod
00010 {
00011     int type;
00012     uint16_t tid;
00013     uint16_t data;
00014     int16_t sdata;
00015 } m_pipeline_mod;
00016
00017 int init_pipeline_mod(m_pipeline_mod *mod);
00018
00019 m_pipeline_mod create_pipeline_mod_append_transformer           (uint16_t type);
00020 m_pipeline_mod create_pipeline_mod_move_transformer           (uint16_t tid, uint16_t position);
00021 m_pipeline_mod create_pipeline_mod_remove_transformer          (uint16_t tid);
00022 m_pipeline_mod create_pipeline_mod_change_transformer_setting (uint16_t tid, uint16_t setting_id,
      int16_t new_value);
00023
00024 int apply_pipeline_mod(m_pipeline *pipeline, m_pipeline_mod mod, int *err_code);
00025
00026 const char *pipeline_mod_type_string(int type);
00027
00028 DECLARE_LINKED_LIST(m_pipeline_mod);
00029
00030 #endif
```

4.63 m_eng_printf.h File Reference

Functions

- void m_voice_printf (int who, const char *fmt,...)
- void m_mute_voice (int who)
- void m_unmute_voice (int who)
- void m_printf (const char *fmt,...)
- void pretty_print_block (int16_t *data, const char *start)
- void pretty_print_block_float (float *data, const char *start)
- void serial_print_blocks (int n,...)

4.63.1 Function Documentation

4.63.1.1 m_mute_voice()

```
void m_mute_voice (
            int who)
```

4.63.1.2 m_printf()

```
void m_printf (
            const char * fmt,
             ...)
```

4.63.1.3 m_unmute_voice()

```
void m_unmute_voice (
            int who)
```

4.63.1.4 m_voice_printf()

```
void m_voice_printf (
            int who,
            const char * fmt,
             ...)
```

4.63.1.5 pretty_print_block()

```
void pretty_print_block (
            int16_t * data,
            const char * start)
```

4.63.1.6 pretty_print_block_float()

```
void pretty_print_block_float (
            float * data,
            const char * start)
```

4.63.1.7 serial_print_blocks()

```
void serial_print_blocks (
            int n,
             ...)
```

4.64 m_eng_printf.h

Go to the documentation of this file.

```
00001 #ifndef MY_PRINTF_H_
00002 #define MY_PRINTF_H_
00003
00004 void m_voice_printf(int who, const char *fmt, ...);
00005
00006 void m_mute_voice(int who);
00007 void m_unmute_voice(int who);
00008
00009 void m_printf(const char *fmt, ...);
00010 void pretty_print_block(int16_t *data, const char *start);
00011 void pretty_print_block_float(float *data, const char *start);
00012
00013 void serial_print_blocks(int n, ...);
00014
00015 #endif
```

4.65 m_eng_profile.h File Reference

```
#include "m_eng_pipeline.h"
```

Data Structures

- struct m_eng_profile

Functions

- int nullify_profile (m_eng_profile *profile)
- int init_profile (m_eng_profile *profile)
- int init_bypass_profile (m_eng_profile *profile)
- int profile_process (m_eng_profile *profile, float *dest, float *src)
- int profile_update (m_eng_profile *profile)
- int profile_apply_pipeline_mod (m_eng_profile *profile, m_pipeline_mod mod)

  Applies the given modification to the given profile.
- int profile_trigger_pipeline_swap (m_eng_profile *profile)
- int profile_scheduled_maintainance (m_eng_profile *profile)

4.65.1 Function Documentation

4.65.1.1 init_bypass_profile()

```
int init_bypass_profile (
            m_eng_profile * profile)
```

4.65.1.2 init_profile()

```
int init_profile (
            m_eng_profile * profile)
```

4.65.1.3 nullify_profile()

```
int nullify_profile (
            m_eng_profile * profile)
```

4.65.1.4 profile_apply_pipeline_mod()

```
int profile_apply_pipeline_mod (
            m_eng_profile * profile,
            m_pipeline_mod mod)
```

Applies the given modification to the given profile.

If the current profile is active, to prevent audio artifacts, pipeline modifications ("mods", encoded by m_pipeline_mod) are applied in the background - to an inactive copy of the profile's pipeline (the "back pipeline"), and the outputs from the front and back pipeline are smoothly cross-faded between, after which it can be safely applied to the (now back, previously front) pipeline, keeping the two in sync.

To achieve this, profile_apply_pipeline_mod calls apply_pipeline_mod to apply the mod to the back pipeline, calls profile_trigger_pipeline_swap to trigger the pipeline swap, and saves the mod in a linked list (m_eng_profile::jobs). After the swap is complete, profile_update will retrieve the job from the list and apply it to the back (previously front) pipeline.

If, however, when profile_apply_pipeline_mod is called, a pipeline swap is already in progress, the mod is *not* applied, but rather stored in a second linked list, (m_eng_profile::blocked_jobs), whereupon profile_update will apply it (after the swap is finished and any jobs waiting in (m_eng_profile::jobs) have been applied) by calling profile_apply_pipeline_mod.

Parameters

profile	The profile to be modified
mod	A struct describing the modification to apply

4.65.1.5 profile_process()

```
int profile_process (
            m_eng_profile * profile,
            float * dest,
            float * src)
```

4.65.1.6 profile_scheduled_maintainance()

```
int profile_scheduled_maintainance (
            m_eng_profile * profile)
```

4.65.1.7 profile_trigger_pipeline_swap()

```
int profile_trigger_pipeline_swap (
            m_eng_profile * profile)
```

4.65.1.8 profile_update()

```
int profile_update (
            m_eng_profile * profile)
```

4.66 m_eng_profile.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_PROFILE_H_
00002 #define M_ENG_PROFILE_H_
00003
00004 #include "m_eng_pipeline.h"
00005
00006 typedef struct
00007 {
00008     m_pipeline *front_pipeline;
00009     m_pipeline *back_pipeline;
00010
00011     int pipelines_swapping;
00012     int pipeline_swap_progress;
00013     int pipeline_swap_samples;
00014     int pipeline_swap_type;
00015     int back_pipeline_warmed_up;
00016
00017     m_pipeline_mod_ll *jobs;
00018     m_pipeline_mod_ll *blocked_jobs;
00019
00020     int active;
00021     int transition_policy;
00022
00023     float *prev_block;
00024
00025     m_transformer output_amp;
00026
00027     int runs;
00028 } m_eng_profile;
00029
00030 int nullify_profile(m_eng_profile *profile);
00031 int init_profile(m_eng_profile *profile);
00032 int init_bypass_profile(m_eng_profile *profile);
00033
00034 int profile_process(m_eng_profile *profile, float *dest, float *src);
00035 int profile_update(m_eng_profile *profile);
00036
00037 int profile_apply_pipeline_mod(m_eng_profile *profile, m_pipeline_mod mod);
00038
00039 int profile_trigger_pipeline_swap(m_eng_profile *profile);
00040
00041 int profile_scheduled_maintainance(m_eng_profile *profile);
00042
00043 #endif
```

4.67 m_eng_sgtl5000.h File Reference

Functions

- int sgtl5000_start ()
- int sgtl5000_soft_reboot ()
- int sgtl5000_healthy ()
- int sgtl5000_enable ()

- void sgtl5000_set_address (uint8_t level)
- int sgtl5000_volume (float n)
- int sgtl5000_mute_headphone ()
- int sgtl5000_unmute_headphone ()
- int sgtl5000_mute_line_out ()
- int sgtl5000_unmute_line_out ()
- int sgtl5000_mic_gain (unsigned int dB)
- int sgtl5000_line_in_level (uint8_t n)
- unsigned short sgtl5000_line_out_level (uint8_t n)
- unsigned short sgtl5000_adc_high_pass_filter_enable ()
- unsigned short sgtl5000_adc_high_pass_filter_freeze ()
- unsigned short sgtl5000_adc_high_pass_filter_disable ()
- void sgtl5000_kill_automation ()
- void sgtl5000_set_master_mode (uint32_t freqMCLK_in)
- int sgtl5000_volum_eng_integer (unsigned int n)
- unsigned int sgtl5000_read_reg (unsigned int reg)
- int sgtl5000_write_reg (unsigned int reg, unsigned int val)
- unsigned int sgtl5000_modify_reg (unsigned int reg, unsigned int val, unsigned int i_mask)
- unsigned short sgtl5000_dap_audio_eq_band (uint8_t band_num, float n)
- void sgtl5000_automate (uint8_t dap, uint8_t eq)
- unsigned char calc_vol (float n, unsigned char range)

4.67.1 Function Documentation

4.67.1.1 calc_vol()

```
unsigned char calc_vol (
            float n,
            unsigned char range)
```

4.67.1.2 sgtl5000_adc_high_pass_filter_disable()

```
unsigned short sgtl5000_adc_high_pass_filter_disable ()
```

4.67.1.3 sgtl5000_adc_high_pass_filter_enable()

```
unsigned short sgtl5000_adc_high_pass_filter_enable ()
```

4.67.1.4 sgtl5000_adc_high_pass_filter_freeze()

```
unsigned short sgtl5000_adc_high_pass_filter_freeze ()
```

4.67.1.5 sgtl5000_automate()

```
void sgtl5000_automate (
            uint8_t dap,
            uint8_t eq)
```

4.67.1.6 sgtl5000_dap_audio_eq_band()

```
unsigned short sgtl5000_dap_audio_eq_band (
            uint8_t band_num,
            float n)
```

4.67.1.7 sgtl5000_enable()

```
int sgtl5000_enable ()
```

4.67.1.8 sgtl5000_healthy()

```
int sgtl5000_healthy ()
```

4.67.1.9 sgtl5000_kill_automation()

```
void sgtl5000_kill_automation ()
```

4.67.1.10 sgtl5000_line_in_level()

```
int sgtl5000_line_in_level (
            uint8_t n)
```

4.67.1.11 sgtl5000_line_out_level()

```
unsigned short sgtl5000_line_out_level (
            uint8_t n)
```

4.67.1.12 sgtl5000_mic_gain()

```
int sgtl5000_mic_gain (
            unsigned int dB)
```

4.67.1.13 sgtl5000_modify_reg()

```
unsigned int sgtl5000_modify_reg (
            unsigned int reg,
            unsigned int val,
            unsigned int i_mask)
```

4.67.1.14 sgtl5000_mute_headphone()

```
int sgtl5000_mute_headphone ()
```

4.67.1.15 sgtl5000_mute_line_out()

```
int sgtl5000_mute_line_out ()
```

4.67.1.16 sgtl5000_read_reg()

```
unsigned int sgtl5000_read_reg (
            unsigned int reg)
```

4.67.1.17 sgtl5000_set_address()

```
void sgtl5000_set_address (
            uint8_t level)
```

4.67.1.18 sgtl5000_set_master_mode()

```
void sgtl5000_set_master_mode (
            uint32_t freqMCLK_in)
```

4.67.1.19 sgtl5000_soft_reboot()

```
int sgtl5000_soft_reboot ()
```

4.67.1.20 sgtl5000_start()

```
int sgtl5000_start ()
```

4.67.1.21 sgtl5000_unmute_headphone()

```
int sgtl5000_unmute_headphone ()
```

4.67.1.22 sgtl5000_unmute_line_out()

```
int sgtl5000_unmute_line_out ()
```

4.67.1.23 sgtl5000_volum_eng_integer()

```
int sgtl5000_volum_eng_integer (
            unsigned int n)
```

4.67.1.24 sgtl5000_volume()

```
int sgtl5000_volume (
            float n)
```

4.67.1.25 sgtl5000_write_reg()

```
int sgtl5000_write_reg (
            unsigned int reg,
            unsigned int val)
```

4.68 m_eng_sgtl5000.h

Go to the documentation of this file.

```
00001 #ifndef control_sgtl5000_h_
00002 #define control_sgtl5000_h_
00003
00004 int sgtl5000_start();
00005 int sgtl5000_soft_reboot();
00006
00007 int sgtl5000_healthy();
00008
00009 int sgtl5000_enable();
00010
00011 void sgtl5000_set_address(uint8_t level);
00012
00013 int sgtl5000_volume(float n);
00014 int sgtl5000_mute_headphone();
00015 int sgtl5000_unmute_headphone();
00016 int sgtl5000_mute_line_out();
00017 int sgtl5000_unmute_line_out();
00018
00019 int sgtl5000_mic_gain(unsigned int dB);
00020 int sgtl5000_line_in_level(uint8_t n);
00021 unsigned short sgtl5000_line_out_level(uint8_t n);
00022 unsigned short sgtl5000_adc_high_pass_filter_enable();
00023 unsigned short sgtl5000_adc_high_pass_filter_freeze();
00024 unsigned short sgtl5000_adc_high_pass_filter_disable();
00025 void sgtl5000_kill_automation();
00026 void sgtl5000_set_master_mode(uint32_t freqMCLK_in);
00027
00028 int             sgtl5000_volum_eng_integer      (unsigned int n); // range: 0x00 to 0x80
00029 unsigned int    sgtl5000_read_reg         (unsigned int reg);
00030 int             sgtl5000_write_reg           (unsigned int reg, unsigned int val);
00031 unsigned int    sgtl5000_modify_reg        (unsigned int reg, unsigned int val, unsigned int i_mask);
00032 unsigned short  sgtl5000_dap_audio_eq_band(uint8_t band_num, float n);
00033
00034 void sgtl5000_automate(uint8_t dap, uint8_t eq);
00035
00036 unsigned char   calc_vol(float n, unsigned char range);
00037
00038 #endif
```

4.69 m_eng_sgtl5000_defs.h File Reference

Macros

- #define CHIP_ID 0x0000
- #define CHIP_DIG_POWER 0x0002
- #define CHIP_CLK_CTRL 0x0004
- #define CHIP_I2S_CTRL 0x0006
- #define CHIP_SSS_CTRL 0x000A
- #define CHIP_ADCDAC_CTRL 0x000E

- #define CHIP_DAC_VOL 0x0010
- #define CHIP_PAD_STRENGTH 0x0014
- #define CHIP_ANA_ADC_CTRL 0x0020
- #define CHIP_ANA_HP_CTRL 0x0022
- #define CHIP_ANA_CTRL 0x0024
- #define CHIP_LINREG_CTRL 0x0026
- #define CHIP_REF_CTRL 0x0028
- #define CHIP_MIC_CTRL 0x002A
- #define CHIP_LINE_OUT_CTRL 0x002C
- #define CHIP_LINE_OUT_VOL 0x002E
- #define CHIP_ANA_POWER 0x0030
- #define CHIP_PLL_CTRL 0x0032
- #define CHIP_CLK_TOP_CTRL 0x0034
- #define CHIP_ANA_STATUS 0x0036
- #define CHIP_ANA_TEST1 0x0038
- #define CHIP_ANA_TEST2 0x003A
- #define CHIP_SHORT_CTRL 0x003C
- #define DAP_CONTROL 0x0100
- #define DAP_PEQ 0x0102
- #define DAP_BASS_ENHANCE 0x0104
- #define DAP_BASS_ENHANCE_CTRL 0x0106
- #define DAP_AUDIO_EQ 0x0108
- #define DAP_SGTL_SURROUND 0x010A
- #define DAP_FILTER_COEF_ACCESS 0x010C
- #define DAP_COEF_WR_B0_MSB 0x010E
- #define DAP_COEF_WR_B0_LSB 0x0110
- #define DAP_AUDIO_EQ_BASS_BAND0 0x0116
- #define DAP_AUDIO_EQ_BAND1 0x0118
- #define DAP_AUDIO_EQ_BAND2 0x011A
- #define DAP_AUDIO_EQ_BAND3 0x011C
- #define DAP_AUDIO_EQ_TREBLE_BAND4 0x011E
- #define DAP_MAIN_CHAN 0x0120
- #define DAP_MIX_CHAN 0x0122
- #define DAP_AVC_CTRL 0x0124
- #define DAP_AVC_THRESHOLD 0x0126
- #define DAP_AVC_ATTACK 0x0128
- #define DAP_AVC_DECAY 0x012A
- #define DAP_COEF_WR_B1_MSB 0x012C
- #define DAP_COEF_WR_B1_LSB 0x012E
- #define DAP_COEF_WR_B2_MSB 0x0130
- #define DAP_COEF_WR_B2_LSB 0x0132
- #define DAP_COEF_WR_A1_MSB 0x0134
- #define DAP_COEF_WR_A1_LSB 0x0136
- #define DAP_COEF_WR_A2_MSB 0x0138
- #define DAP_COEF_WR_A2_LSB 0x013A
- #define SGTL5000_I2C_ADDR_CS_LOW 0x0A
- #define SGTL5000_I2C_ADDR_CS_HIGH 0x2A
- #define FILTER_LOPASS 0
- #define FILTER_HIPASS 1
- #define FILTER_BANDPASS 2
- #define FILTER_NOTCH 3
- #define FILTER_PARAEQ 4
- #define FILTER_LOSHELF 5
- #define FILTER_HISHELF 6
- #define FLAT_FREQUENCY 0

- #define PARAMETRIC_EQUALIZER 1
- #define TONE_CONTROLS 2
- #define GRAPHIC_EQUALIZER 3
- #define AUDIO_HEADPHONE_DAC 0
- #define AUDIO_HEADPHONE_LINEIN 1

4.69.1 Macro Definition Documentation

4.69.1.1 AUDIO_HEADPHONE_DAC

```
#define AUDIO_HEADPHONE_DAC 0
```

4.69.1.2 AUDIO_HEADPHONE_LINEIN

```
#define AUDIO_HEADPHONE_LINEIN 1
```

4.69.1.3 CHIP_ADCDAC_CTRL

```
#define CHIP_ADCDAC_CTRL 0x000E
```

4.69.1.4 CHIP_ANA_ADC_CTRL

```
#define CHIP_ANA_ADC_CTRL 0x0020
```

4.69.1.5 CHIP_ANA_CTRL

```
#define CHIP_ANA_CTRL 0x0024
```

4.69.1.6 CHIP_ANA_HP_CTRL

```
#define CHIP_ANA_HP_CTRL 0x0022
```

4.69.1.7 CHIP_ANA_POWER

```
#define CHIP_ANA_POWER 0x0030
```

4.69.1.8 CHIP_ANA_STATUS

```
#define CHIP_ANA_STATUS 0x0036
```

4.69.1.9 CHIP_ANA_TEST1

```
#define CHIP_ANA_TEST1 0x0038
```

4.69.1.10 CHIP_ANA_TEST2

```
#define CHIP_ANA_TEST2 0x003A
```

4.69.1.11 CHIP_CLK_CTRL

```
#define CHIP_CLK_CTRL 0x0004
```

4.69.1.12 CHIP_CLK_TOP_CTRL

```
#define CHIP_CLK_TOP_CTRL 0x0034
```

4.69.1.13 CHIP_DAC_VOL

```
#define CHIP_DAC_VOL 0x0010
```

4.69.1.14 CHIP_DIG_POWER

```
#define CHIP_DIG_POWER 0x0002
```

4.69.1.15 CHIP_I2S_CTRL

```
#define CHIP_I2S_CTRL 0x0006
```

4.69.1.16 CHIP_ID

```
#define CHIP_ID 0x0000
```

4.69.1.17 CHIP_LINE_OUT_CTRL

```
#define CHIP_LINE_OUT_CTRL 0x002C
```

4.69.1.18 CHIP_LINE_OUT_VOL

```
#define CHIP_LINE_OUT_VOL 0x002E
```

4.69.1.19 CHIP_LINREG_CTRL

```
#define CHIP_LINREG_CTRL 0x0026
```

4.69.1.20 CHIP_MIC_CTRL

```
#define CHIP_MIC_CTRL 0x002A
```

4.69.1.21 CHIP_PAD_STRENGTH

```
#define CHIP_PAD_STRENGTH 0x0014
```

4.69.1.22 CHIP_PLL_CTRL

```
#define CHIP_PLL_CTRL 0x0032
```

4.69.1.23 CHIP_REF_CTRL

```
#define CHIP_REF_CTRL 0x0028
```

4.69.1.24 CHIP_SHORT_CTRL

```
#define CHIP_SHORT_CTRL 0x003C
```

4.69.1.25 CHIP_SSS_CTRL

```
#define CHIP_SSS_CTRL 0x000A
```

4.69.1.26 DAP_AUDIO_EQ

```
#define DAP_AUDIO_EQ 0x0108
```

4.69.1.27 DAP_AUDIO_EQ_BAND1

```
#define DAP_AUDIO_EQ_BAND1 0x0118
```

4.69.1.28 DAP_AUDIO_EQ_BAND2

```
#define DAP_AUDIO_EQ_BAND2 0x011A
```

4.69.1.29 DAP_AUDIO_EQ_BAND3

```
#define DAP_AUDIO_EQ_BAND3 0x011C
```

4.69.1.30 DAP_AUDIO_EQ_BASS_BAND0

```
#define DAP_AUDIO_EQ_BASS_BAND0 0x0116
```

4.69.1.31 DAP_AUDIO_EQ_TREBLE_BAND4

```
#define DAP_AUDIO_EQ_TREBLE_BAND4 0x011E
```

4.69.1.32 DAP_AVC_ATTACK

```
#define DAP_AVC_ATTACK 0x0128
```

4.69.1.33 DAP_AVC_CTRL

```
#define DAP_AVC_CTRL 0x0124
```

4.69.1.34 DAP_AVC_DECAY

```
#define DAP_AVC_DECAY 0x012A
```

4.69.1.35 DAP_AVC_THRESHOLD

```
#define DAP_AVC_THRESHOLD 0x0126
```

4.69.1.36 DAP_BASS_ENHANCE

```
#define DAP_BASS_ENHANCE 0x0104
```

4.69.1.37 DAP_BASS_ENHANCE_CTRL

```
#define DAP_BASS_ENHANCE_CTRL 0x0106
```

4.69.1.38 DAP_COEF_WR_A1_LSB

```
#define DAP_COEF_WR_A1_LSB 0x0136
```

4.69.1.39 DAP_COEF_WR_A1_MSB

```
#define DAP_COEF_WR_A1_MSB 0x0134
```

4.69.1.40 DAP_COEF_WR_A2_LSB

```
#define DAP_COEF_WR_A2_LSB 0x013A
```

4.69.1.41 DAP_COEF_WR_A2_MSB

```
#define DAP_COEF_WR_A2_MSB 0x0138
```

4.69.1.42 DAP_COEF_WR_B0_LSB

```
#define DAP_COEF_WR_B0_LSB 0x0110
```

4.69.1.43 DAP_COEF_WR_B0_MSB

```
#define DAP_COEF_WR_B0_MSB 0x010E
```

4.69.1.44 DAP_COEF_WR_B1_LSB

```
#define DAP_COEF_WR_B1_LSB 0x012E
```

4.69.1.45 DAP_COEF_WR_B1_MSB

```
#define DAP_COEF_WR_B1_MSB 0x012C
```

4.69.1.46 DAP_COEF_WR_B2_LSB

```
#define DAP_COEF_WR_B2_LSB 0x0132
```

4.69.1.47 DAP_COEF_WR_B2_MSB

```
#define DAP_COEF_WR_B2_MSB 0x0130
```

4.69.1.48 DAP_CONTROL

```
#define DAP_CONTROL 0x0100
```

4.69.1.49 DAP_FILTER_COEF_ACCESS

```
#define DAP_FILTER_COEF_ACCESS 0x010C
```

4.69.1.50 DAP_MAIN_CHAN

```
#define DAP_MAIN_CHAN 0x0120
```

4.69.1.51 DAP_MIX_CHAN

```
#define DAP_MIX_CHAN 0x0122
```

4.69.1.52 DAP_PEQ

```
#define DAP_PEQ 0x0102
```

4.69.1.53 DAP_SGTL_SURROUND

```
#define DAP_SGTL_SURROUND 0x010A
```

4.69.1.54 FILTER_BANDPASS

```
#define FILTER_BANDPASS 2
```

4.69.1.55 FILTER_HIPASS

```
#define FILTER_HIPASS 1
```

4.69.1.56 FILTER_HISHELF

```
#define FILTER_HISHELF 6
```

4.69.1.57 FILTER_LOPASS

```
#define FILTER_LOPASS 0
```

4.69.1.58 FILTER_LOSHELF

```
#define FILTER_LOSHELF 5
```

4.69.1.59 FILTER_NOTCH

```
#define FILTER_NOTCH 3
```

4.69.1.60 FILTER_PARAEQ

```
#define FILTER_PARAEQ 4
```

4.69.1.61 FLAT_FREQUENCY

```
#define FLAT_FREQUENCY 0
```

4.69.1.62 GRAPHIC_EQUALIZER

```
#define GRAPHIC_EQUALIZER 3
```

4.69.1.63 PARAMETRIC_EQUALIZER

```
#define PARAMETRIC_EQUALIZER 1
```

4.69.1.64 SGTL5000_I2C_ADDR_CS_HIGH

```
#define SGTL5000_I2C_ADDR_CS_HIGH 0x2A
```

4.69.1.65 SGTL5000_I2C_ADDR_CS_LOW

```
#define SGTL5000_I2C_ADDR_CS_LOW 0x0A
```

4.69.1.66 TONE_CONTROLS

```
#define TONE_CONTROLS 2
```

4.70 m_eng_sgtl5000_defs.h

Go to the documentation of this file.

```
00001 #ifndef M_SGTL5000_DEFS_H_
00002 #define M_SGTL5000_DEFS_H_
00003
00004 #define CHIP_ID             0x0000
00005 // 15:8 PARTID      0xA0 - 8 bit identifier for SGTL5000
00006 // 7:0  REVID       0x00 - revision number for SGTL5000.
00007
00008 #define CHIP_DIG_POWER          0x0002
00009 // 6    ADC_POWERUP 1=Enable, 0=disable the ADC block, both digital & analog,
00010 // 5    DAC_POWERUP 1=Enable, 0=disable the DAC block, both analog and digital
00011 // 4    DAP_POWERUP 1=Enable, 0=disable the DAP block
00012 // 1    I2S_OUT_POWERUP 1=Enable, 0=disable the I2S data output
00013 // 0    I2S_IN_POWERUP  1=Enable, 0=disable the I2S data input
00014
00015 #define CHIP_CLK_CTRL           0x0004
00016 // 5:4  RATE_MODE   Sets the sample rate mode. MCLK_FREQ is still specified
00017 //          relative to the rate in SYS_FS
00018 //              0x0 = SYS_FS specifies the rate
00019 //              0x1 = Rate is 1/2 of the SYS_FS rate
00020 //              0x2 = Rate is 1/4 of the SYS_FS rate
00021 //              0x3 = Rate is 1/6 of the SYS_FS rate
00022 // 3:2  SYS_FS      Sets the internal system sample rate (default=2)
00023 //              0x0 = 32 kHz
00024 //              0x1 = 44.1 kHz
00025 //              0x2 = 48 kHz
00026 //              0x3 = 96 kHz
00027 // 1:0  MCLK_FREQ   Identifies incoming SYS_MCLK frequency and if the PLL should be used
00028 //              0x0 = 256*Fs
00029 //              0x1 = 384*Fs
00030 //              0x2 = 512*Fs
00031 //              0x3 = Use PLL
00032 //              The 0x3 (Use PLL) setting must be used if the SYS_MCLK is not
00033 //              a standard multiple of Fs (256, 384, or 512). This setting can
00034 //              also be used if SYS_MCLK is a standard multiple of Fs.
00035 //              Before this field is set to 0x3 (Use PLL), the PLL must be
00036 //              powered up by setting CHIP_ANA_POWER->PLL_POWERUP and
00037 //              CHIP_ANA_POWER->VCOAMP_POWERUP.  Also, the PLL dividers must
00038 //              be calculated based on the external MCLK rate and
00039 //              CHIP_PLL_CTRL register must be set (see CHIP_PLL_CTRL register
00040 //              description details on how to calculate the divisors).
00041
00042 #define CHIP_I2S_CTRL           0x0006
00043 // 8    SCLKFREQ    Sets frequency of I2S_SCLK when in master mode (MS=1). When in slave
00044 //          mode (MS=0), this field must be set appropriately to match SCLK input
00045 //          rate.
00046 //              0x0 = 64Fs
00047 //              0x1 = 32Fs - Not supported for RJ mode (I2S_MODE = 1)
00048 // 7    MS      Configures master or slave of I2S_LRCLK and I2S_SCLK.
00049 //              0x0 = Slave: I2S_LRCLK an I2S_SCLK are inputs
00050 //              0x1 = Master: I2S_LRCLK and I2S_SCLK are outputs
00051 //              NOTE: If the PLL is used (CHIP_CLK_CTRL->MCLK_FREQ==0x3),
00052 //              the SGTL5000 must be a master of the I2S port (MS==1)
00053 // 6    SCLK_INV    Sets the edge that data (input and output) is clocked in on for I2S_SCLK
00054 //              0x0 = data is valid on rising edge of I2S_SCLK
00055 //              0x1 = data is valid on falling edge of I2S_SCLK
00056 // 5:4  DLEN        I2S data length (default=1)
00057 //              0x0 = 32 bits (only valid when SCLKFREQ=0),
00058 //                  not valid for Right Justified Mode
00059 //              0x1 = 24 bits (only valid when SCLKFREQ=0)
00060 //              0x2 = 20 bits
00061 //              0x3 = 16 bits
00062 // 3:2  I2S_MODE    Sets the mode for the I2S port
00063 //              0x0 = I2S mode or Left Justified (Use LRALIGN to select)
00064 //              0x1 = Right Justified Mode
00065 //              0x2 = PCM Format A/B
00066 //              0x3 = RESERVED
00067 // 1    LRALIGN     I2S_LRCLK Alignment to data word. Not used for Right Justified mode
00068 //              0x0 = Data word starts 1 I2S_SCLK delay after I2S_LRCLK
00069 //                  transition (I2S format, PCM format A)
00070 //              0x1 = Data word starts after I2S_LRCLK transition (left
00071 //                  justified format, PCM format B)
00072 // 0    LRPOL       I2S_LRCLK Polarity when data is presented.
00073 //              0x0 = I2S_LRCLK = 0 - Left, 1 - Right
00074 //              1x0 = I2S_LRCLK = 0 - Right, 1 - Left
00075 //              The left subframe should be presented first regardless of
00076 //              the setting of LRPOL.
00077
00078 #define CHIP_SSS_CTRL           0x000A
00079 // 14   DAP_MIX_LRSWAP  DAP Mixer Input Swap
00080 //              0x0 = Normal Operation
00081 //              0x1 = Left and Right channels for the DAP MIXER Input are swapped.
00082 // 13   DAP_LRSWAP  DAP Mixer Input Swap
```

```
00083 //              0x0 = Normal Operation
00084 //              0x1 = Left and Right channels for the DAP Input are swapped
00085 // 12   DAC_LRSWAP  DAC Input Swap
00086 //              0x0 = Normal Operation
00087 //              0x1 = Left and Right channels for the DAC are swapped
00088 // 10   I2S_LRSWAP  I2S_DOUT Swap
00089 //              0x0 = Normal Operation
00090 //              0x1 = Left and Right channels for the I2S_DOUT are swapped
00091 // 9:8  DAP_MIX_SELECT  Select data source for DAP mixer
00092 //              0x0 = ADC
00093 //              0x1 = I2S_IN
00094 //              0x2 = Reserved
00095 //              0x3 = Reserved
00096 // 7:6  DAP_SELECT  Select data source for DAP
00097 //              0x0 = ADC
00098 //              0x1 = I2S_IN
00099 //              0x2 = Reserved
00100 //              0x3 = Reserved
00101 // 5:4  DAC_SELECT  Select data source for DAC (default=1)
00102 //              0x0 = ADC
00103 //              0x1 = I2S_IN
00104 //              0x2 = Reserved
00105 //              0x3 = DAP
00106 // 1:0  I2S_SELECT  Select data source for I2S_DOUT
00107 //              0x0 = ADC
00108 //              0x1 = I2S_IN
00109 //              0x2 = Reserved
00110 //              0x3 = DAP
00111
00112 #define CHIP_ADCDAC_CTRL        0x000E
00113 // 13   VOL_BUSY_DAC_RIGHT Volume Busy DAC Right
00114 //              0x0 = Ready
00115 //              0x1 = Busy - This indicates the channel has not reached its
00116 //                  programmed volume/mute level
00117 // 12   VOL_BUSY_DAC_LEFT  Volume Busy DAC Left
00118 //              0x0 = Ready
00119 //              0x1 = Busy - This indicates the channel has not reached its
00120 //                  programmed volume/mute level
00121 // 9    VOL_RAMP_EN Volume Ramp Enable (default=1)
00122 //              0x0 = Disables volume ramp. New volume settings take immediate
00123 //                  effect without a ramp
00124 //              0x1 = Enables volume ramp
00125 //              This field affects DAC_VOL. The volume ramp effects both
00126 //              volume settings and mute When set to 1 a soft mute is enabled.
00127 // 8    VOL_EXPO_RAMP   Exponential Volume Ramp Enable
00128 //              0x0 = Linear ramp over top 4 volume octaves
00129 //              0x1 = Exponential ramp over full volume range
00130 //              This bit only takes effect if VOL_RAMP_EN is 1.
00131 // 3    DAC_MUTE_RIGHT  DAC Right Mute (default=1)
00132 //              0x0 = Unmute
00133 //              0x1 = Muted
00134 //              If VOL_RAMP_EN = 1, this is a soft mute.
00135 // 2    DAC_MUTE_LEFT   DAC Left Mute (default=1)
00136 //              0x0 = Unmute
00137 //              0x1 = Muted
00138 //              If VOL_RAMP_EN = 1, this is a soft mute.
00139 // 1    ADC_HPF_FREEZE  ADC High Pass Filter Freeze
00140 //              0x0 = Normal operation
00141 //              0x1 = Freeze the ADC high-pass filter offset register.  The
00142 //              offset continues to be subtracted from the ADC data stream.
00143 // 0    ADC_HPF_BYPASS  ADC High Pass Filter Bypass
00144 //              0x0 = Normal operation
00145 //              0x1 = Bypassed and offset not updated
00146
00147 #define CHIP_DAC_VOL            0x0010
00148 // 15:8 DAC_VOL_RIGHT   DAC Right Channel Volume.  Set the Right channel DAC volume
00149 //          with 0.5017 dB steps from 0 to -90 dB
00150 //              0x3B and less = Reserved
00151 //              0x3C = 0 dB
00152 //              0x3D = -0.5 dB
00153 //              0xF0 = -90 dB
00154 //              0xFC and greater = Muted
00155 //              If VOL_RAMP_EN = 1, there is an automatic ramp to the
00156 //              new volume setting.
00157 // 7:0  DAC_VOL_LEFT    DAC Left Channel Volume.  Set the Left channel DAC volume
00158 //          with 0.5017 dB steps from 0 to -90 dB
00159 //              0x3B and less = Reserved
00160 //              0x3C = 0 dB
00161 //              0x3D = -0.5 dB
00162 //              0xF0 = -90 dB
00163 //              0xFC and greater = Muted
00164 //              If VOL_RAMP_EN = 1, there is an automatic ramp to the
00165 //              new volume setting.
00166
00167 #define CHIP_PAD_STRENGTH       0x0014
00168 // 9:8  I2S_LRCLK   I2S LRCLK Pad Drive Strength (default=1)
00169 //              Sets drive strength for output pads per the table below.
```

```
00170 //               VDDIO  1.8 V    2.5 V   3.3 V
00171 //              0x0 = Disable
00172 //              0x1 =    1.66 mA   2.87 mA   4.02 mA
00173 //              0x2 =    3.33 mA   5.74 mA   8.03 mA
00174 //              0x3 =    4.99 mA   8.61 mA   12.05 mA
00175 // 7:6  I2S_SCLK    I2S SCLK Pad Drive Strength (default=1)
00176 // 5:4  I2S_DOUT    I2S DOUT Pad Drive Strength (default=1)
00177 // 3:2  CTRL_DATA   I2C DATA Pad Drive Strength (default=3)
00178 // 1:0  CTRL_CLK    I2C CLK Pad Drive Strength (default=3)
00179 //              (all use same table as I2S_LRCLK)
00180
00181 #define CHIP_ANA_ADC_CTRL       0x0020
00182 // 8    ADC_VOL_M6DB    ADC Volume Range Reduction
00183 //              This bit shifts both right and left analog ADC volume
00184 //              range down by 6.0 dB.
00185 //              0x0 = No change in ADC range
00186 //              0x1 = ADC range reduced by 6.0 dB
00187 // 7:4  ADC_VOL_RIGHT   ADC Right Channel Volume
00188 //              Right channel analog ADC volume control in 1.5 dB steps.
00189 //              0x0 = 0 dB
00190 //              0x1 = +1.5 dB
00191 //              ...
00192 //              0xF = +22.5 dB
00193 //              This range is -6.0 dB to +16.5 dB if ADC_VOL_M6DB is set to 1.
00194 // 3:0  ADC_VOL_LEFT    ADC Left Channel Volume
00195 //              (same scale as ADC_VOL_RIGHT)
00196
00197 #define CHIP_ANA_HP_CTRL        0x0022
00198 // 14:8 HP_VOL_RIGHT    Headphone Right Channel Volume  (default 0x18)
00199 //              Right channel headphone volume control with 0.5 dB steps.
00200 //              0x00 = +12 dB
00201 //              0x01 = +11.5 dB
00202 //              0x18 = 0 dB
00203 //              ...
00204 //              0x7F = -51.5 dB
00205 // 6:0  HP_VOL_LEFT Headphone Left Channel Volume  (default 0x18)
00206 //              (same scale as HP_VOL_RIGHT)
00207
00208 #define CHIP_ANA_CTRL           0x0024
00209 // 8    MUTE_LO     LINEOUT Mute, 0 = Unmute, 1 = Mute  (default 1)
00210 // 6    SELECT_HP   Select the headphone input, 0 = DAC, 1 = LINEIN
00211 // 5    EN_ZCD_HP   Enable the headphone zero cross detector (ZCD)
00212 //              0x0 = HP ZCD disabled
00213 //              0x1 = HP ZCD enabled
00214 // 4    MUTE_HP     Mute the headphone outputs, 0 = Unmute, 1 = Mute (default)
00215 // 2    SELECT_ADC  Select the ADC input, 0 = Microphone, 1 = LINEIN
00216 // 1    EN_ZCD_ADC  Enable the ADC analog zero cross detector (ZCD)
00217 //              0x0 = ADC ZCD disabled
00218 //              0x1 = ADC ZCD enabled
00219 // 0    MUTE_ADC    Mute the ADC analog volume, 0 = Unmute, 1 = Mute (default)
00220
00221 #define CHIP_LINREG_CTRL        0x0026
00222 // 6    VDDC_MAN_ASSN   Determines chargepump source when VDDC_ASSN_OVRD is set.
00223 //              0x0 = VDDA
00224 //              0x1 = VDDIO
00225 // 5    VDDC_ASSN_OVRD  Charge pump Source Assignment Override
00226 //              0x0 = Charge pump source is automatically assigned based
00227 //                  on higher of VDDA and VDDIO
00228 //              0x1 = the source of charge pump is manually assigned by
00229 //                  VDDC_MAN_ASSN If VDDIO and VDDA are both the same
00230 //                  and greater than 3.1 V, VDDC_ASSN_OVRD and
00231 //                  VDDC_MAN_ASSN should be used to manually assign
00232 //                  VDDIO as the source for charge pump.
00233 // 3:0  D_PROGRAMMING   Sets the VDDD linear regulator output voltage in 50 mV steps.
00234 //          Must clear the LINREG_SIMPLE_POWERUP and STARTUP_POWERUP bits
00235 //          in the 0x0030 (CHIP_ANA_POWER) register after power-up, for
00236 //          this setting to produce the proper VDDD voltage.
00237 //              0x0 = 1.60
00238 //              0xF = 0.85
00239
00240 #define CHIP_REF_CTRL           0x0028 // bandgap reference bias voltage and currents
00241 // 8:4  VAG_VAL     Analog Ground Voltage Control
00242 //              These bits control the analog ground voltage in 25 mV steps.
00243 //              This should usually be set to VDDA/2 or lower for best
00244 //              performance (maximum output swing at minimum THD). This VAG
00245 //              reference is also used for the DAC and ADC voltage reference.
00246 //              So changing this voltage scales the output swing of the DAC
00247 //              and the output signal of the ADC.
00248 //              0x00 = 0.800 V
00249 //              0x1F = 1.575 V
00250 // 3:1  BIAS_CTRL   Bias control
00251 //              These bits adjust the bias currents for all of the analog
00252 //              blocks. By lowering the bias current a lower quiescent power
00253 //              is achieved. It should be noted that this mode can affect
00254 //              performance by 3-4 dB.
00255 //              0x0 = Nominal
00256 //              0x1-0x3=+12.5%
```

```
00257 //              0x4=-12.5%
00258 //              0x5=-25%
00259 //              0x6=-37.5%
00260 //              0x7=-50%
00261 // 0    SMALL_POP   VAG Ramp Control
00262 //              Setting this bit slows down the VAG ramp from ~200 to ~400 ms
00263 //              to reduce the startup pop, but increases the turn on/off time.
00264 //              0x0 = Normal VAG ramp
00265 //              0x1 = Slow down VAG ramp
00266
00267 #define CHIP_MIC_CTRL           0x002A // microphone gain & internal microphone bias
00268 // 9:8  BIAS_RESISTOR   MIC Bias Output Impedance Adjustment
00269 //              Controls an adjustable output impedance for the microphone bias.
00270 //              If this is set to zero the micbias block is powered off and
00271 //              the output is highZ.
00272 //              0x0 = Powered off
00273 //              0x1 = 2.0 kohm
00274 //              0x2 = 4.0 kohm
00275 //              0x3 = 8.0 kohm
00276 // 6:4  BIAS_VOLT   MIC Bias Voltage Adjustment
00277 //              Controls an adjustable bias voltage for the microphone bias
00278 //              amp in 250 mV steps. This bias voltage setting should be no
00279 //              more than VDDA-200 mV for adequate power supply rejection.
00280 //              0x0 = 1.25 V
00281 //              ...
00282 //              0x7 = 3.00 V
00283 // 1:0  GAIN        MIC Amplifier Gain
00284 //              Sets the microphone amplifier gain. At 0 dB setting the THD
00285 //              can be slightly higher than other paths- typically around
00286 //              ~65 dB. At other gain settings the THD are better.
00287 //              0x0 = 0 dB
00288 //              0x1 = +20 dB
00289 //              0x2 = +30 dB
00290 //              0x3 = +40 dB
00291
00292 #define CHIP_LINE_OUT_CTRL      0x002C
00293 // 11:8 OUT_CURRENT Controls the output bias current for the LINEOUT amplifiers.  The
00294 //          nominal recommended setting for a 10 kohm load with 1.0 nF load cap
00295 //          is 0x3. There are only 5 valid settings.
00296 //              0x0=0.18 mA
00297 //              0x1=0.27 mA
00298 //              0x3=0.36 mA
00299 //              0x7=0.45 mA
00300 //              0xF=0.54 mA
00301 // 5:0  LO_VAGCNTRL LINEOUT Amplifier Analog Ground Voltage
00302 //              Controls the analog ground voltage for the LINEOUT amplifiers
00303 //              in 25 mV steps. This should usually be set to VDDIO/2.
00304 //              0x00 = 0.800 V
00305 //              ...
00306 //              0x1F = 1.575 V
00307 //              ...
00308 //              0x23 = 1.675 V
00309 //              0x24-0x3F are invalid
00310
00311 #define CHIP_LINE_OUT_VOL       0x002E
00312 // 12:8 LO_VOL_RIGHT    LINEOUT Right Channel Volume (default=4)
00313 //              Controls the right channel LINEOUT volume in 0.5 dB steps.
00314 //              Higher codes have more attenuation.
00315 // 4:0  LO_VOL_LEFT LINEOUT Left Channel Output Level (default=4)
00316 //              Used to normalize the output level of the left line output
00317 //              to full scale based on the values used to set
00318 //              LINE_OUT_CTRL->LO_VAGCNTRL and CHIP_REF_CTRL->VAG_VAL.
00319 //              In general this field should be set to:
00320 //              40*log((VAG_VAL)/(LO_VAGCNTRL)) + 15
00321 //              Suggested values based on typical VDDIO and VDDA voltages.
00322 //                  VDDA  VAG_VAL VDDIO  LO_VAGCNTRL LO_VOL_*
00323 //                  1.8 V   0.9   3.3 V  1.55     0x06
00324 //                  1.8 V   0.9   1.8 V   0.9     0x0F
00325 //                  3.3 V   1.55   1.8 V      0.9     0x19
00326 //                  3.3 V   1.55   3.3 V     1.55     0x0F
00327 //              After setting to the nominal voltage, this field can be used
00328 //              to adjust the output level in +/-0.5 dB increments by using
00329 //              values higher or lower than the nominal setting.
00330
00331 #define CHIP_ANA_POWER          0x0030 // power down controls for the analog blocks.
00332         // The only other power-down controls are BIAS_RESISTOR in the MIC_CTRL register
00333         //  and the EN_ZCD control bits in ANA_CTRL.
00334 // 14   DAC_MONO    While DAC_POWERUP is set, this allows the DAC to be put into left only
00335 //              mono operation for power savings. 0=mono, 1=stereo (default)
00336 // 13   LINREG_SIMPLE_POWERUP  Power up the simple (low power) digital supply regulator.
00337 //              After reset, this bit can be cleared IF VDDD is driven
00338 //              externally OR the primary digital linreg is enabled with
00339 //              LINREG_D_POWERUP
00340 // 12   STARTUP_POWERUP Power up the circuitry needed during the power up ramp and reset.
00341 //              After reset this bit can be cleared if VDDD is coming from
00342 //              an external source.
00343 // 11   VDDC_CHRGPMP_POWERUP Power up the VDDC charge pump block. If neither VDDA or VDDIO
```

```
00344 //              is 3.0 V or larger this bit should be cleared before analog
00345 //              blocks are powered up.
00346 // 10   PLL_POWERUP PLL Power Up, 0 = Power down, 1 = Power up
00347 //              When cleared, the PLL is turned off. This must be set before
00348 //              CHIP_CLK_CTRL->MCLK_FREQ is programmed to 0x3. The
00349 //              CHIP_PLL_CTRL register must be configured correctly before
00350 //              setting this bit.
00351 // 9    LINREG_D_POWERUP Power up the primary VDDD linear regulator, 0 = Power down, 1 = Power up
00352 // 8    VCOAMP_POWERUP  Power up the PLL VCO amplifier, 0 = Power down, 1 = Power up
00353 // 7    VAG_POWERUP Power up the VAG reference buffer.
00354 //              Setting this bit starts the power up ramp for the headphone
00355 //              and LINEOUT. The headphone (and/or LINEOUT) powerup should
00356 //              be set BEFORE clearing this bit. When this bit is cleared
00357 //              the power-down ramp is started. The headphone (and/or LINEOUT)
00358 //              powerup should stay set until the VAG is fully ramped down
00359 //              (200 to 400 ms after clearing this bit).
00360 //              0x0 = Power down, 0x1 = Power up
00361 // 6    ADC_MONO    While ADC_POWERUP is set, this allows the ADC to be put into left only
00362 //              mono operation for power savings. This mode is useful when
00363 //              only using the microphone input.
00364 //              0x0 = Mono (left only), 0x1 = Stereo
00365 // 5    REFTOP_POWERUP  Power up the reference bias currents
00366 //              0x0 = Power down, 0x1 = Power up
00367 //              This bit can be cleared when the part is a sleep state
00368 //              to minimize analog power.
00369 // 4    HEADPHONE_POWERUP Power up the headphone amplifiers
00370 //              0x0 = Power down, 0x1 = Power up
00371 // 3    DAC_POWERUP Power up the DACs
00372 //              0x0 = Power down, 0x1 = Power up
00373 // 2    CAPLESS_HEADPHONE_POWERUP Power up the capless headphone mode
00374 //              0x0 = Power down, 0x1 = Power up
00375 // 1    ADC_POWERUP Power up the ADCs
00376 //              0x0 = Power down, 0x1 = Power up
00377 // 0    LINEOUT_POWERUP Power up the LINEOUT amplifiers
00378 //              0x0 = Power down, 0x1 = Power up
00379
00380 #define CHIP_PLL_CTRL           0x0032
00381 // 15:11 INT_DIVISOR
00382 // 10:0 FRAC_DIVISOR
00383
00384 #define CHIP_CLK_TOP_CTRL       0x0034
00385 // 11   ENABLE_INT_OSC  Setting this bit enables an internal oscillator to be used for the
00386 //              zero cross detectors, the short detect recovery, and the
00387 //              charge pump. This allows the I2S clock to be shut off while
00388 //              still operating an analog signal path. This bit can be kept
00389 //              on when the I2S clock is enabled, but the I2S clock is more
00390 //              accurate so it is preferred to clear this bit when I2S is present.
00391 // 3    INPUT_FREQ_DIV2 SYS_MCLK divider before PLL input
00392 //              0x0 = pass through
00393 //              0x1 = SYS_MCLK is divided by 2 before entering PLL
00394 //              This must be set when the input clock is above 17 Mhz. This
00395 //              has no effect when the PLL is powered down.
00396
00397 #define CHIP_ANA_STATUS         0x0036
00398 // 9    LRSHORT_STS This bit is high whenever a short is detected on the left or right
00399 //              channel headphone drivers.
00400 // 8    CSHORT_STS  This bit is high whenever a short is detected on the capless headphone
00401 //              common/center channel driver.
00402 // 4    PLL_IS_LOCKED   This bit goes high after the PLL is locked.
00403
00404 #define CHIP_ANA_TEST1          0x0038 //  intended only for debug.
00405 #define CHIP_ANA_TEST2          0x003A //  intended only for debug.
00406
00407 #define CHIP_SHORT_CTRL         0x003C
00408 // 14:12 LVLADJR    Right channel headphone short detector in 25 mA steps.
00409 //              0x3=25 mA
00410 //              0x2=50 mA
00411 //              0x1=75 mA
00412 //              0x0=100 mA
00413 //              0x4=125 mA
00414 //              0x5=150 mA
00415 //              0x6=175 mA
00416 //              0x7=200 mA
00417 //              This trip point can vary by ~30% over process so leave plenty
00418 //              of guard band to avoid false trips.  This short detect trip
00419 //              point is also effected by the bias current adjustments made
00420 //              by CHIP_REF_CTRL->BIAS_CTRL and by CHIP_ANA_TEST1->HP_IALL_ADJ.
00421 // 10:8 LVLADJL     Left channel headphone short detector in 25 mA steps.
00422 //              (same scale as LVLADJR)
00423 // 6:4  LVLADJC     Capless headphone center channel short detector in 50 mA steps.
00424 //              0x3=50 mA
00425 //              0x2=100 mA
00426 //              0x1=150 mA
00427 //              0x0=200 mA
00428 //              0x4=250 mA
00429 //              0x5=300 mA
00430 //              0x6=350 mA
```

```
00431 //              0x7=400 mA
00432 // 3:2  MODE_LR     Behavior of left/right short detection
00433 //              0x0 = Disable short detector, reset short detect latch,
00434 //                  software view non-latched short signal
00435 //              0x1 = Enable short detector and reset the latch at timeout
00436 //                  (every ~50 ms)
00437 //              0x2 = This mode is not used/invalid
00438 //              0x3 = Enable short detector with only manual reset (have
00439 //                  to return to 0x0 to reset the latch)
00440 // 1:0  MODE_CM     Behavior of capless headphone central short detection
00441 //              (same settings as MODE_LR)
00442
00443 #define DAP_CONTROL         0x0100
00444 #define DAP_PEQ             0x0102
00445 #define DAP_BASS_ENHANCE        0x0104
00446 #define DAP_BASS_ENHANCE_CTRL       0x0106
00447 #define DAP_AUDIO_EQ            0x0108
00448 #define DAP_SGTL_SURROUND       0x010A
00449 #define DAP_FILTER_COEF_ACCESS      0x010C
00450 #define DAP_COEF_WR_B0_MSB      0x010E
00451 #define DAP_COEF_WR_B0_LSB      0x0110
00452 #define DAP_AUDIO_EQ_BASS_BAND0     0x0116 // 115 Hz
00453 #define DAP_AUDIO_EQ_BAND1      0x0118 // 330 Hz
00454 #define DAP_AUDIO_EQ_BAND2      0x011A // 990 Hz
00455 #define DAP_AUDIO_EQ_BAND3      0x011C // 3000 Hz
00456 #define DAP_AUDIO_EQ_TREBLE_BAND4   0x011E // 9900 Hz
00457 #define DAP_MAIN_CHAN           0x0120
00458 #define DAP_MIX_CHAN            0x0122
00459 #define DAP_AVC_CTRL            0x0124
00460 #define DAP_AVC_THRESHOLD       0x0126
00461 #define DAP_AVC_ATTACK          0x0128
00462 #define DAP_AVC_DECAY           0x012A
00463 #define DAP_COEF_WR_B1_MSB      0x012C
00464 #define DAP_COEF_WR_B1_LSB      0x012E
00465 #define DAP_COEF_WR_B2_MSB      0x0130
00466 #define DAP_COEF_WR_B2_LSB      0x0132
00467 #define DAP_COEF_WR_A1_MSB      0x0134
00468 #define DAP_COEF_WR_A1_LSB      0x0136
00469 #define DAP_COEF_WR_A2_MSB      0x0138
00470 #define DAP_COEF_WR_A2_LSB      0x013A
00471
00472 #define SGTL5000_I2C_ADDR_CS_LOW    0x0A  // CTRL_ADR0_CS pin low (normal configuration)
00473 #define SGTL5000_I2C_ADDR_CS_HIGH   0x2A  // CTRL_ADR0_CS  pin highz
00474
00475 //For Filter Type: 0 = LPF, 1 = HPF, 2 = BPF, 3 = NOTCH, 4 = PeakingEQ, 5 = LowShelf, 6 = HighShelf
00476 #define FILTER_LOPASS   0
00477 #define FILTER_HIPASS   1
00478 #define FILTER_BANDPASS 2
00479 #define FILTER_NOTCH    3
00480 #define FILTER_PARAEQ   4
00481 #define FILTER_LOSHELF  5
00482 #define FILTER_HISHELF  6
00483
00484 //For frequency adjustment
00485 #define FLAT_FREQUENCY 0
00486 #define PARAMETRIC_EQUALIZER 1
00487 #define TONE_CONTROLS 2
00488 #define GRAPHIC_EQUALIZER 3
00489
00490 // SGTL5000-specific defines for headphones
00491 #define AUDIO_HEADPHONE_DAC 0
00492 #define AUDIO_HEADPHONE_LINEIN 1
00493
00494 #endif
```

4.71 m_eng_simple_distortion.h File Reference

Data Structures

- struct m_eng_simple_distortion_str

Functions

- int init_simple_distortion_str (m_eng_simple_distortion_str *str)
- int reconfigure_simple_distortion (void *data_struct)
- int calc_simple_distortion (void *data_struct, float *dest, float *src, int n_samples)

4.71.1 Function Documentation

4.71.1.1 calc_simple_distortion()

```
int calc_simple_distortion (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.71.1.2 init_simple_distortion_str()

```
int init_simple_distortion_str (
            m_eng_simple_distortion_str * str)
```

4.71.1.3 reconfigure_simple_distortion()

```
int reconfigure_simple_distortion (
            void * data_struct)
```

4.72 m_eng_simple_distortion.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_SIMPLE_DISTORTION_H_
00002 #define M_ENG_SIMPLE_DISTORTION_H_
00003
00004 typedef struct
00005 {
00006     m_parameter pregain;
00007     m_parameter postgain;
00008
00009 } m_eng_simple_distortion_str;
00010
00011 int init_simple_distortion_str(m_eng_simple_distortion_str *str);
00012 int reconfigure_simple_distortion(void *data_struct);
00013 int calc_simple_distortion(void *data_struct, float *dest, float *src, int n_samples);
00014
00015 #endif
```

4.73 m_eng_transformer.h File Reference

```
#include "m_transformer_enum.h"
#include "m_parameter.h"
#include "m_eng_linkowitz_riley.h"
#include "m_transformer.h"
```

Macros

- #define TRANSFORMER_MAX_INPUTS 4
- #define TRANSFORMER_MAX_OUTPUTS 4
- #define FADER_FADE_IN 0
- #define FADER_FADE_OUT 1
- #define PRINT_TRANSFORMER_INFO
- #define TRANSFORMER_SWITCH_ACTION_BYPASS 0
- #define TRANSFORMER_TRANSITION_INSTANT 0
- #define TRANSFORMER_TRANSITION_MONOBLOCK_LINEAR 1
- #define TRANSFORMER_TRANSITION_BIBLOCK_LINEAR 2
- #define TRANSFORMER_TRANSITION_QUADBLOCK_LINEAR 3
- #define TRANSFORMER_TRANSITION_TAIL 4
- #define N_NATIVE_PARAMETERS 3
- #define N_NATIVE_SETTINGS 1

Functions

- int transformer_add_setting (m_transformer *trans, m_setting *setting)
- int transformer_add_parameter (m_transformer *trans, m_parameter *param)
- m_parameter * transformer_get_parameter (m_transformer *trans, uint16_t ppid)
- m_setting * transformer_get_setting (m_transformer *trans, uint16_t sid)
- int transformer_init_parameter_array (m_transformer *trans, int n)
- int transformer_init_setting_array (m_transformer *trans, int n)
- int run_transformer (m_transformer *trans, float *dest, float *src)
- void free_transformer (m_transformer *trans)
- int clone_transformer (m_transformer **dest_ptr, m_transformer *src)
- const char * transformer_type_to_string (uint16_t type)
- int transformer_init_controls (m_transformer *trans)

4.73.1 Macro Definition Documentation

4.73.1.1 FADER_FADE_IN

```
#define FADER_FADE_IN 0
```

4.73.1.2 FADER_FADE_OUT

```
#define FADER_FADE_OUT 1
```

4.73.1.3 N_NATIVE_PARAMETERS

```
#define N_NATIVE_PARAMETERS 3
```

4.73.1.4 N_NATIVE_SETTINGS

```
#define N_NATIVE_SETTINGS 1
```

4.73.1.5 PRINT_TRANSFORMER_INFO

```
#define PRINT_TRANSFORMER_INFO
```

4.73.1.6 TRANSFORMER_MAX_INPUTS

```
#define TRANSFORMER_MAX_INPUTS 4
```

4.73.1.7 TRANSFORMER_MAX_OUTPUTS

```
#define TRANSFORMER_MAX_OUTPUTS 4
```

4.73.1.8 TRANSFORMER_SWITCH_ACTION_BYPASS

```
#define TRANSFORMER_SWITCH_ACTION_BYPASS 0
```

4.73.1.9 TRANSFORMER_TRANSITION_BIBLOCK_LINEAR

```
#define TRANSFORMER_TRANSITION_BIBLOCK_LINEAR 2
```

4.73.1.10 TRANSFORMER_TRANSITION_INSTANT

```
#define TRANSFORMER_TRANSITION_INSTANT 0
```

4.73.1.11 TRANSFORMER_TRANSITION_MONOBLOCK_LINEAR

```
#define TRANSFORMER_TRANSITION_MONOBLOCK_LINEAR 1
```

4.73.1.12 TRANSFORMER_TRANSITION_QUADBLOCK_LINEAR

```
#define TRANSFORMER_TRANSITION_QUADBLOCK_LINEAR 3
```

4.73.1.13 TRANSFORMER_TRANSITION_TAIL

```
#define TRANSFORMER_TRANSITION_TAIL 4
```

4.73.2 Function Documentation

4.73.2.1 clone_transformer()

```
int clone_transformer (
            m_transformer ** dest_ptr,
            m_transformer * src)
```

4.73.2.2 free_transformer()

```
void free_transformer (
            m_transformer * trans)
```

4.73.2.3 run_transformer()

```
int run_transformer (
            m_transformer * trans,
            float * dest,
            float * src)
```

4.73.2.4 transformer_add_parameter()

```
int transformer_add_parameter (
            m_transformer * trans,
            m_parameter * param)
```

4.73.2.5 transformer_add_setting()

```
int transformer_add_setting (
            m_transformer * trans,
            m_setting * setting)
```

4.73.2.6 transformer_get_parameter()

```
m_parameter * transformer_get_parameter (
            m_transformer * trans,
            uint16_t ppid)
```

4.73.2.7 transformer_get_setting()

```
m_setting * transformer_get_setting (
            m_transformer * trans,
            uint16_t sid)
```

4.73.2.8 transformer_init_controls()

```
int transformer_init_controls (
            m_transformer * trans)
```

4.73.2.9 transformer_init_parameter_array()

```
int transformer_init_parameter_array (
            m_transformer * trans,
            int n)
```

4.73.2.10 transformer_init_setting_array()

```
int transformer_init_setting_array (
            m_transformer * trans,
            int n)
```

4.73.2.11 transformer_type_to_string()

```
const char * transformer_type_to_string (
            uint16_t type)
```

4.74 m_eng_transformer.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_TRANSFORMER_H_
00002 #define M_ENG_TRANSFORMER_H_
00003
00004 #include "m_transformer_enum.h"
00005
00006 #include "m_parameter.h"
00007
00008 #define TRANSFORMER_MAX_INPUTS      4
00009 #define TRANSFORMER_MAX_OUTPUTS     4
00010
00011 #define FADER_FADE_IN  0
00012 #define FADER_FADE_OUT 1
00013
00014 #define PRINT_TRANSFORMER_INFO
00015
00016 #define TRANSFORMER_SWITCH_ACTION_BYPASS 0
00017
00018 #define TRANSFORMER_TRANSITION_INSTANT          0
00019 #define TRANSFORMER_TRANSITION_MONOBLOCK_LINEAR 1
00020 #define TRANSFORMER_TRANSITION_BIBLOCK_LINEAR   2
00021 #define TRANSFORMER_TRANSITION_QUADBLOCK_LINEAR 3
00022 #define TRANSFORMER_TRANSITION_TAIL             4
00023
00024 #define N_NATIVE_PARAMETERS 3
00025 #define N_NATIVE_SETTINGS 1
00026
00027 #include "m_eng_linkowitz_riley.h"
00028
00029 #include "m_transformer.h"
00030
00031 int transformer_add_setting(m_transformer *trans, m_setting *setting);
00032 int transformer_add_parameter(m_transformer *trans, m_parameter *param);
00033
00034 m_parameter *transformer_get_parameter(m_transformer *trans, uint16_t ppid);
00035 m_setting   *transformer_get_setting  (m_transformer *trans, uint16_t sid );
00036
00037 int transformer_init_parameter_array(m_transformer *trans, int n);
00038 int transformer_init_setting_array(m_transformer *trans, int n);
00039
00040 int run_transformer(m_transformer *trans, float *dest, float *src);
00041
00042 void free_transformer(m_transformer *trans);
00043
00044 int clone_transformer(m_transformer **dest_ptr, m_transformer *src);
00045
00046 const char *transformer_type_to_string(uint16_t type);
00047
00048 int transformer_init_controls(m_transformer *trans);
00049
00050 #endif
```

4.75 m_eng_transformer_init.h File Reference

Functions

- int init_transformer (m_transformer *trans, uint16_t type)

4.75.1 Function Documentation

4.75.1.1 init_transformer()

```
int init_transformer (
            m_transformer * trans,
            uint16_t type)
```

4.76 m_eng_transformer_init.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_TRANSFORMER_INIT_H_
00002 #define M_ENG_TRANSFORMER_INIT_H_
00003
00004 int init_transformer(m_transformer *trans, uint16_t type);
00005
00006 #endif
```

4.77 m_eng_transformer_template.h File Reference

Data Structures

- struct m_transformer_str

Functions

- int init_transformer_str (m_transformer_str ∗str)
- int reconfigure_transformer (void ∗data_struct)
- int calc_transformer (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.77.1 Function Documentation

4.77.1.1 calc_transformer()

```
int calc_transformer (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.77.1.2 init_transformer_str()

```
int init_transformer_str (
            m_transformer_str * str)
```

4.77.1.3 reconfigure_transformer()

```
int reconfigure_transformer (
            void * data_struct)
```

4.78 m_eng_transformer_template.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_TRANSFORMER_H_
00002 #define M_ENG_TRANSFORMER_H_
00003
00004 typedef struct
00005 {
00006     m_parameter param;
00007
00008 } m_transformer_str;
00009
00010 int init_transformer_str(m_transformer_str *str);
00011 int reconfigure_transformer(void *data_struct);
00012 int calc_transformer(void *data_struct, float *dest, float *src, int n_samples);
00013
00014 #endif
```

4.79 m_eng_transition.h File Reference

Macros

- #define TRANSITION_MONOBLOCK_COS2 0
- #define TRANSITION_QUADBLOCK_COS2 1
- #define TRANSITION_OCTOBLOCK_COS2 2
- #define TRANSITION_TAIL 3
- #define TAIL_NEW_FADE_IN_SAMPLES 256
- #define TAIL_INPUT_FADE_SAMPLES 256

4.79.1 Macro Definition Documentation

4.79.1.1 TAIL_INPUT_FADE_SAMPLES

```
#define TAIL_INPUT_FADE_SAMPLES 256
```

4.79.1.2 TAIL_NEW_FADE_IN_SAMPLES

```
#define TAIL_NEW_FADE_IN_SAMPLES 256
```

4.79.1.3 TRANSITION_MONOBLOCK_COS2

```
#define TRANSITION_MONOBLOCK_COS2 0
```

4.79.1.4 TRANSITION_OCTOBLOCK_COS2

```
#define TRANSITION_OCTOBLOCK_COS2 2
```

4.79.1.5 TRANSITION_QUADBLOCK_COS2

```
#define TRANSITION_QUADBLOCK_COS2 1
```

4.79.1.6 TRANSITION_TAIL

```
#define TRANSITION_TAIL 3
```

4.80 m_eng_transition.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_TRANSITION_H_
00002 #define M_ENG_TRANSITION_H_
00003
00004 #define TRANSITION_MONOBLOCK_COS2   0
00005 #define TRANSITION_QUADBLOCK_COS2   1
00006 #define TRANSITION_OCTOBLOCK_COS2   2
00007 #define TRANSITION_TAIL             3
00008
00009 #define TAIL_NEW_FADE_IN_SAMPLES    256
00010 #define TAIL_INPUT_FADE_SAMPLES     256
00011
00012 #endif
```

4.81 m_eng_update.h File Reference

Macros

- #define m_eng_disable_software_interrupts()
- #define m_eng_enable_software_interrupts()

Functions

- void update_all ()
- int update_setup ()
- void update_stop ()
- void m_eng_software_isr ()

4.81.1 Macro Definition Documentation

4.81.1.1 m_eng_disable_software_interrupts

```
#define m_eng_disable_software_interrupts()
```

Value:

```
(NVIC_DISABLE_IRQ(IRQ_SOFTWARE))
```

4.81.1.2 m_eng_enable_software_interrupts

```
#define m_eng_enable_software_interrupts()
```

Value:
```
(NVIC_ENABLE_IRQ(IRQ_SOFTWARE))
```

4.81.2 Function Documentation

4.81.2.1 m_eng_software_isr()

```
void m_eng_software_isr ()
```

4.81.2.2 update_all()

```
void update_all ()
```

4.81.2.3 update_setup()

```
int update_setup ()
```

4.81.2.4 update_stop()

```
void update_stop ()
```

4.82 m_eng_update.h

Go to the documentation of this file.

```
00001 #ifndef M_UPDATE_H_
00002 #define M_UPDATE_H_
00003
00004 #ifndef M_SIMULATED
00005 #define m_eng_disable_software_interrupts()             (NVIC_DISABLE_IRQ(IRQ_SOFTWARE))
00006 #define m_eng_enable_software_interrupts()              (NVIC_ENABLE_IRQ(IRQ_SOFTWARE))
00007 #else
00008 #define m_eng_disable_software_interrupts()
00009 #define m_eng_enable_software_interrupts()
00010 #endif
00011
00012 void update_all();
00013
00014 int update_setup();
00015 void update_stop();
00016
00017 void m_eng_software_isr();
00018
00019 #endif
```

4.83 m_eng_useful_functions.h File Reference

Functions

- float identity_function (float x)
- float normalised_arctan (float x)
- float hard_clip (float x)
- float soft_fold (float x)
- float trig_transition_function (float x)
- int convert_block_int_to_float (float ∗dest, int16_t ∗src)
- int convert_block_float_to_int (int16_t ∗src, float ∗dest)

4.83.1 Function Documentation

4.83.1.1 convert_block_float_to_int()

```
int convert_block_float_to_int (
            int16_t * src,
            float * dest)
```

4.83.1.2 convert_block_int_to_float()

```
int convert_block_int_to_float (
            float * dest,
            int16_t * src)
```

4.83.1.3 hard_clip()

```
float hard_clip (
            float x)
```

4.83.1.4 identity_function()

```
float identity_function (
            float x)
```

4.83.1.5 normalised_arctan()

```
float normalised_arctan (
            float x)
```

4.83.1.6 soft_fold()

```
float soft_fold (
            float x)
```

4.83.1.7 trig_transition_function()

```
float trig_transition_function (
            float x)
```

4.84 m_eng_useful_functions.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_USEFUL_FUNCTIONS_H_
00002 #define M_ENG_USEFUL_FUNCTIONS_H_
00003
00004 float identity_function(float x);
00005 float normalised_arctan(float x);
00006 float hard_clip(float x);
00007 float soft_fold(float x);
00008 float trig_transition_function(float x);
00009
00010 int convert_block_int_to_float(float *dest, int16_t *src);
00011 int convert_block_float_to_int(int16_t *src, float *dest);
00012
00013 static inline int positive_mod(int x, int m)
00014 {
00015     int r = x % m;
00016     return r < 0 ? r + m : r;
00017 }
00018
00019 static inline int floor_divide(int x, int d)
00020 {
00021     // d > 0
00022     if (x >= 0) return x / d;
00023     return -(( -x + d - 1) / d);
00024 }
00025
00026 static inline int ring_index(int i, int n)
00027 {
00028     int r = i % n;
00029     return r < 0 ? r + n : r;
00030 }
00031
00032 #endif
```

4.85 m_eng_warbler.h File Reference

Data Structures

- struct m_eng_warbler_str

Functions

- int init_warbler_str (m_eng_warbler_str *str)
- int reconfigure_warbler (void *data_struct)
- int calc_warbler (void *data_struct, float *dest, float *src, int n_samples)

4.85.1 Function Documentation

4.85.1.1 calc_warbler()

```
int calc_warbler (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.85.1.2 init_warbler_str()

```
int init_warbler_str (
            m_eng_warbler_str * str)
```

4.85.1.3 reconfigure_warbler()

```
int reconfigure_warbler (
            void * data_struct)
```

4.86 m_eng_warbler.h

Go to the documentation of this file.

```
00001 #ifndef M_ENG_WARBLER_H_
00002 #define M_ENG_WARBLER_H_
00003
00004 typedef struct
00005 {
00006     m_parameter center;
00007     m_parameter width;
00008     m_parameter reactivity;
00009     m_parameter sensitivity;
00010     m_parameter max_rate;
00011     m_parameter min_rate;
00012
00013     float alpha;
00014     float e;
00015
00016     float t;
00017     float rate;
00018
00019     m_eng_band_pass_filter_str filter;
00020 } m_eng_warbler_str;
00021
00022 int init_warbler_str(m_eng_warbler_str *str);
00023 int reconfigure_warbler(void *data_struct);
00024 int calc_warbler(void *data_struct, float *dest, float *src, int n_samples);
00025
00026 #endif
```

4.87 m_eng_waveshaper.h File Reference

Data Structures

- struct m_eng_waveshaper_str

Macros

- #define WAVESHAPER_ENVELOPE_ATTACK expf(-7.0/8.0)
- #define WAVESHAPER_ENVELOPE_RELEASE expf(-7.0/2048.0)

Functions

- int init_waveshaper_str (m_eng_waveshaper_str ∗str)
- int calc_waveshaper (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.87.1 Macro Definition Documentation

4.87.1.1 WAVESHAPER_ENVELOPE_ATTACK

```
#define WAVESHAPER_ENVELOPE_ATTACK expf(-7.0/8.0)
```

4.87.1.2 WAVESHAPER_ENVELOPE_RELEASE

```
#define WAVESHAPER_ENVELOPE_RELEASE expf(-7.0/2048.0)
```

4.87.2 Function Documentation

4.87.2.1 calc_waveshaper()

```
int calc_waveshaper (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.87.2.2 init_waveshaper_str()

```
int init_waveshaper_str (
            m_eng_waveshaper_str * str)
```

4.88 m_eng_waveshaper.h

Go to the documentation of this file.

```
00001 #ifndef M_WAVESHAPER_H_
00002 #define M_WAVESHAPER_H_
00003
00004 #define WAVESHAPER_ENVELOPE_ATTACK  expf(-7.0/8.0)
00005 #define WAVESHAPER_ENVELOPE_RELEASE expf(-7.0/2048.0)
00006
00007 typedef struct
00008 {
00009     m_parameter coefficient;
00010
00011     float (*shape)(float x);
00012 } m_eng_waveshaper_str;
00013
00014 int init_waveshaper_str(m_eng_waveshaper_str *str);
00015 int calc_waveshaper(void *data_struct, float *dest, float *src, int n_samples);
00016
00017
00018 #endif
```

4.89 m_eng.cpp File Reference

```
#include <Wire.h>
#include "m_eng.h"
```

Macros

- #define LED_BLINK_MILLIS 50
- #define DEBUG_PRINT_MILLIS 1000
- #define PRINT_LOG
- #define LOG_PRINT_MILLIS 10
- #define PROFILER_PRINT_MILLIS 10000
- #define MEM_REPORT_MILLIS 1000
- #define SCHEDULED_MAINTAINANCE_MILLIS 100
- #define SGTL5000_CHECK_PERIOD 100
- #define SCHEDULED_MAINTAINANCE

Functions

- int main ()

4.89.1 Macro Definition Documentation

4.89.1.1 DEBUG_PRINT_MILLIS

```
#define DEBUG_PRINT_MILLIS 1000
```

4.89.1.2 LED_BLINK_MILLIS

```
#define LED_BLINK_MILLIS 50
```

4.89.1.3 LOG_PRINT_MILLIS

```
#define LOG_PRINT_MILLIS 10
```

4.89.1.4 MEM_REPORT_MILLIS

```
#define MEM_REPORT_MILLIS 1000
```

4.89.1.5 PRINT_LOG

```
#define PRINT_LOG
```

4.89.1.6 PROFILER_PRINT_MILLIS

```
#define PROFILER_PRINT_MILLIS 10000
```

4.89.1.7 SCHEDULED_MAINTAINANCE

```
#define SCHEDULED_MAINTAINANCE
```

4.89.1.8 SCHEDULED_MAINTAINANCE_MILLIS

```
#define SCHEDULED_MAINTAINANCE_MILLIS 100
```

4.89.1.9 SGTL5000_CHECK_PERIOD

```
#define SGTL5000_CHECK_PERIOD 100
```

4.89.2 Function Documentation

4.89.2.1 main()

```
int main ()
```

4.90 m_eng_adaptive_waveshaper.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_adaptive_waveshaper_str (m_eng_adaptive_waveshaper_str ∗str)
- int calc_adaptive_waveshaper (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.90.1 Function Documentation

4.90.1.1 calc_adaptive_waveshaper()

```
int calc_adaptive_waveshaper (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.90.1.2 init_adaptive_waveshaper_str()

```
int init_adaptive_waveshaper_str (
            m_eng_adaptive_waveshaper_str * str)
```

4.91 m_eng_audio_block.c File Reference

```
#include "m_eng.h"
```

4.92 m_eng_band_splitter.c File Reference

4.93 m_eng_biquad.c File Reference

```
#include "m_eng.h"
```

Functions

- int calc_biquad (void *data_struct, float *dest, float *src, int n_samples)
- int reconfigure_biquad (void *data_struct)
- int init_biquad_str (m_eng_biquad_str *str)

4.93.1 Function Documentation

4.93.1.1 calc_biquad()

```
int calc_biquad (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.93.1.2 init_biquad_str()

```
int init_biquad_str (
            m_eng_biquad_str * str)
```

4.93.1.3 reconfigure_biquad()

```
int reconfigure_biquad (
            void * data_struct)
```

4.94 m_eng_buffer_mixer_amp.c File Reference

```
#include "m_eng.h"
```

Functions

- int calc_buffer (void *data_struct, float *dest, float *src, int n_samples)
- int init_amplifier_str (m_eng_amplifier_str *str)
- int reconfigure_amplifier (void *data_struct)
- int calc_amplifier (void *data_struct, float *dest, float *src, int n_samples)

4.94.1 Function Documentation

4.94.1.1 calc_amplifier()

```
int calc_amplifier (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.94.1.2 calc_buffer()

```
int calc_buffer (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.94.1.3 init_amplifier_str()

```
int init_amplifier_str (
            m_eng_amplifier_str * str)
```

4.94.1.4 reconfigure_amplifier()

```
int reconfigure_amplifier (
            void * data_struct)
```

4.95 m_eng_comms.cpp File Reference

```
#include <cstdint>
#include "m_eng.h"
#include "signal.h"
#include <Wire.h>
#include "utility/imxrt_hw.h"
```

Macros

- #define PRINT_RESPONSE_BYTES

Enumerations

- enum comms_fsm_eng_state_t { IDLE , SENDING_STRING , RECIEVING_NEW_PARAM_NAM_ENG_LONG }

Functions

- int m_message_sanity_check (m_message msg, int len)
- void handle_esp32_message (m_message msg)
- void i2c_receive_isr (int n)
- void i2c_request_isr ()
- int init_esp32_link ()
- void esp32_message_check_handle ()

Variables

- m_response response
- m_response prev_response
- m_message received
- volatile uint8_t receive_buffer [M_MESSAGE_MAX_TRANSFER_LEN]
- uint8_t response_buffer [M_MESSAGE_MAX_TRANSFER_LEN]
- uint8_t wait_message [M_MESSAGE_MAX_TRANSFER_LEN]
- volatile unsigned int received_length
- unsigned int response_length
- volatile sig_atomic_t message_pending = 0
- volatile sig_atomic_t response_ready = 0
- char * string_out = NULL
- int string_out_pos
- char * string_in = NULL
- int string_in_pos
- comms_fsm_eng_state_t comms_fsm_eng_state = IDLE

4.95.1 Macro Definition Documentation

4.95.1.1 PRINT_RESPONSE_BYTES

```
#define PRINT_RESPONSE_BYTES
```

4.95.2 Enumeration Type Documentation

4.95.2.1 comms_fsm_eng_state_t

```
enum comms_fsm_eng_state_t
```

Enumerator

IDLE	
SENDING_STRING	
RECIEVING_NEW_PARAM_NAM_ENG_LONG	

4.95.3 Function Documentation

4.95.3.1 esp32_message_check_handle()

```
void esp32_message_check_handle ()
```

4.95.3.2 handle_esp32_message()

```
void handle_esp32_message (
            m_message msg)
```

4.95.3.3 i2c_receive_isr()

```
void i2c_receive_isr (
            int n)
```

4.95.3.4 i2c_request_isr()

```
void i2c_request_isr ()
```

4.95.3.5 init_esp32_link()

```
int init_esp32_link ()
```

4.95.3.6 m_message_sanity_check()

```
int m_message_sanity_check (
            m_message msg,
            int len)
```

4.95.4 Variable Documentation

4.95.4.1 comms_fsm_eng_state

```
comms_fsm_eng_state_t comms_fsm_eng_state = IDLE
```

4.95.4.2 message_pending

```
volatile sig_atomic_t message_pending = 0
```

4.95.4.3 prev_response

```
m_response prev_response
```

4.95.4.4 receive_buffer

```
volatile uint8_t receive_buffer[M_MESSAGE_MAX_TRANSFER_LEN]
```

4.95.4.5 received

```
m_message received
```

4.95.4.6 received_length

```
volatile unsigned int received_length
```

4.95.4.7 response

```
m_response response
```

4.95.4.8 response_buffer

```
uint8_t response_buffer[M_MESSAGE_MAX_TRANSFER_LEN]
```

4.95.4.9 response_length

```
unsigned int response_length
```

4.95.4.10 response_ready

```
volatile sig_atomic_t response_ready = 0
```

4.95.4.11 string_in

```
char* string_in = NULL
```

4.95.4.12 string_in_pos

```
int string_in_pos
```

4.95.4.13 string_out

```
char* string_out = NULL
```

4.95.4.14 string_out_pos

```
int string_out_pos
```

4.95.4.15 wait_message

```
uint8_t wait_message[M_MESSAGE_MAX_TRANSFER_LEN]
```

4.96 m_eng_compressor.c File Reference

```
#include "m_eng.h"
#include "m_eng_compressor.h"
```

Macros

- #define EPSILON 0.00000001

Functions

- int init_compressor_str (m_eng_compressor_str ∗str)
- int reconfigure_compressor (void ∗data_struct)
- int calc_compressor (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.96.1 Macro Definition Documentation

4.96.1.1 EPSILON

```
#define EPSILON 0.00000001
```

4.96.2 Function Documentation

4.96.2.1 calc_compressor()

```
int calc_compressor (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.96.2.2 init_compressor_str()

```
int init_compressor_str (
            m_eng_compressor_str * str)
```

4.96.2.3 reconfigure_compressor()

```
int reconfigure_compressor (
            void * data_struct)
```

4.97 m_eng_context.c File Reference

```
#include "m_eng.h"
```

Macros

- #define AVG_DURATION_UPDATE_COEF 0.99f

Functions

- int init_m_eng_context (m_eng_context *cxt)
- int m_eng_context_new_profile (m_eng_context *cxt)
- int cxt_profile_id_valid (m_eng_context *cxt, uint16_t pid)
- int cxt_transformer_id_valid (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int pcxt_profile_id_valid (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid)
- m_parameter * cxt_get_parameter_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid)
- m_parameter * cxt_get_front_parameter_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid)
- m_parameter * cxt_get_back_parameter_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid)
- int cxt_update_parameter_value_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t ppid, float new_value)
- m_setting * cxt_get_setting_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t sid)
- int cxt_update_setting_value_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid, uint16_t sid, uint16_t new_value)
- void m_eng_safe_reboot (m_eng_context *cxt)
- int reset_context (m_eng_context *cxt)
- int cxt_append_transformer_to_profile (m_eng_context *cxt, uint16_t pid, uint16_t type)
- int cxt_remove_transformer_from_profile (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_move_transformer (m_eng_context *cxt, uint16_t tid, uint16_t new_pos)
- int cxt_insert_transformer_to_profile (m_eng_context *cxt, uint16_t pid, uint16_t type, uint16_t pos)
- int cxt_prepend_transformer_to_profile (m_eng_context *cxt, uint16_t pid, uint16_t type)
- int cxt_get_n_profile_transformers (m_eng_context *cxt, uint16_t pid)
- int cxt_get_n_transformer_params (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_get_n_transformer_settings (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_get_transformer_type (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_get_tid_by_pos (m_eng_context *cxt, uint16_t pid, uint16_t pos)
- m_transformer * cxt_get_transformer_by_id (m_eng_context *cxt, uint16_t pid, uint16_t tid)
- int cxt_set_active_profile (m_eng_context *cxt, uint16_t pid)
- int cxt_switch_to_profile (m_eng_context *cxt, uint16_t pid)
- int cxt_process (m_eng_context *cxt)
- int cxt_run_scheduled_maintainance (m_eng_context *cxt)

4.97.1 Macro Definition Documentation

4.97.1.1 AVG_DURATION_UPDATE_COEF

```
#define AVG_DURATION_UPDATE_COEF 0.99f
```

4.97.2 Function Documentation

4.97.2.1 cxt_append_transformer_to_profile()

```
int cxt_append_transformer_to_profile (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t type)
```

4.97.2.2 cxt_get_back_parameter_by_id()

```
m_parameter * cxt_get_back_parameter_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid)
```

4.97.2.3 cxt_get_front_parameter_by_id()

```
m_parameter * cxt_get_front_parameter_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid)
```

4.97.2.4 cxt_get_n_profile_transformers()

```
int cxt_get_n_profile_transformers (
            m_eng_context * cxt,
            uint16_t pid)
```

4.97.2.5 cxt_get_n_transformer_params()

```
int cxt_get_n_transformer_params (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.97.2.6 cxt_get_n_transformer_settings()

```
int cxt_get_n_transformer_settings (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.97.2.7 cxt_get_parameter_by_id()

```
m_parameter * cxt_get_parameter_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid)
```

4.97.2.8 cxt_get_setting_by_id()

```
m_setting * cxt_get_setting_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t sid)
```

4.97.2.9 cxt_get_tid_by_pos()

```
int cxt_get_tid_by_pos (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t pos)
```

4.97.2.10 cxt_get_transformer_by_id()

```
m_transformer * cxt_get_transformer_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.97.2.11 cxt_get_transformer_type()

```
int cxt_get_transformer_type (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.97.2.12 cxt_insert_transformer_to_profile()

```
int cxt_insert_transformer_to_profile (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t type,
            uint16_t pos)
```

4.97.2.13 cxt_move_transformer()

```
int cxt_move_transformer (
            m_eng_context * cxt,
            uint16_t tid,
            uint16_t new_pos)
```

4.97.2.14 cxt_prepend_transformer_to_profile()

```
int cxt_prepend_transformer_to_profile (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t type)
```

4.97.2.15 cxt_process()

```
int cxt_process (
            m_eng_context * cxt)
```

4.97.2.16 cxt_profile_id_valid()

```
int cxt_profile_id_valid (
            m_eng_context * cxt,
            uint16_t pid)
```

4.97.2.17 cxt_remove_transformer_from_profile()

```
int cxt_remove_transformer_from_profile (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.97.2.18 cxt_run_scheduled_maintainance()

```
int cxt_run_scheduled_maintainance (
            m_eng_context * cxt)
```

4.97.2.19 cxt_set_active_profile()

```
int cxt_set_active_profile (
            m_eng_context * cxt,
            uint16_t pid)
```

4.97.2.20 cxt_switch_to_profile()

```
int cxt_switch_to_profile (
            m_eng_context * cxt,
            uint16_t pid)
```

4.97.2.21 cxt_transformer_id_valid()

```
int cxt_transformer_id_valid (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid)
```

4.97.2.22 cxt_update_parameter_value_by_id()

```
int cxt_update_parameter_value_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid,
            float new_value)
```

4.97.2.23 cxt_update_setting_value_by_id()

```
int cxt_update_setting_value_by_id (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t sid,
            uint16_t new_value)
```

4.97.2.24 init_m_eng_context()

```
int init_m_eng_context (
            m_eng_context * cxt)
```

4.97.2.25 m_eng_context_new_profile()

```
int m_eng_context_new_profile (
            m_eng_context * cxt)
```

4.97.2.26 m_eng_safe_reboot()

```
void m_eng_safe_reboot (
            m_eng_context * cxt)
```

4.97.2.27 pcxt_profile_id_valid()

```
int pcxt_profile_id_valid (
            m_eng_context * cxt,
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid)
```

4.97.2.28 reset_context()

```
int reset_context (
            m_eng_context * cxt)
```

4.98 m_eng_debugging.c File Reference

```
#include "m_eng.h"
```

Functions

- void print_binary (int v, int bits)
- void full_debug_print (m_eng_context *cxt)
- void print_context_info (m_eng_context *cxt, int depth)
- void print_profile_info (m_eng_profile *profile, int depth)
- void print_pipeline_info (m_pipeline *pipeline, int depth)
- void print_transformer_info (m_transformer *trans, int depth)

4.98.1 Function Documentation

4.98.1.1 full_debug_print()

```
void full_debug_print (
            m_eng_context * cxt)
```

4.98.1.2 print_binary()

```
void print_binary (
            int v,
            int bits)
```

4.98.1.3 print_context_info()

```
void print_context_info (
            m_eng_context * cxt,
            int depth)
```

4.98.1.4 print_pipeline_info()

```
void print_pipeline_info (
            m_pipeline * pipeline,
            int depth)
```

4.98.1.5 print_profile_info()

```
void print_profile_info (
            m_eng_profile * profile,
            int depth)
```

4.98.1.6 print_transformer_info()

```
void print_transformer_info (
            m_transformer * trans,
            int depth)
```

4.99 m_eng_delay.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_delay_str (m_eng_delay_str ∗str)
- int reconfigure_delay (void ∗data_struct)
- int calc_delay (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.99.1 Function Documentation

4.99.1.1 calc_delay()

```
int calc_delay (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.99.1.2 init_delay_str()

```
int init_delay_str (
            m_eng_delay_str * str)
```

4.99.1.3 reconfigure_delay()

```
int reconfigure_delay (
            void * data_struct)
```

4.100 m_eng_delay_buffer.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_delay_buffer (m_delay_buffer *buf)
- int m_delay_buffer_resize_samples (m_delay_buffer *buf, int n_samples)
- int m_delay_buffer_resize_seconds (m_delay_buffer *buf, float seconds)
- int m_delay_buffer_resize_milliseconds (m_delay_buffer *buf, float ms)
- int m_delay_buffer_tick (m_delay_buffer *buf, float new_sample)
- int m_delay_buffer_advance (m_delay_buffer *buf, float *new_samples, unsigned int n)
- int m_delay_buffer_get_delayed_sample (m_delay_buffer *buf, float *dest, unsigned int delay)
- float * m_delay_buffer_get_delayed_sample_ptr (m_delay_buffer *buf, unsigned int delay)
- int m_delay_buffer_get_fractional_delayed_sample (m_delay_buffer *buf, float *dest, float delay)

4.100.1 Function Documentation

4.100.1.1 init_delay_buffer()

```
int init_delay_buffer (
            m_delay_buffer * buf)
```

4.100.1.2 m_delay_buffer_advance()

```
int m_delay_buffer_advance (
            m_delay_buffer * buf,
            float * new_samples,
            unsigned int n)
```

4.100.1.3 m_delay_buffer_get_delayed_sample()

```
int m_delay_buffer_get_delayed_sample (
            m_delay_buffer * buf,
            float * dest,
            unsigned int delay)
```

4.100.1.4 m_delay_buffer_get_delayed_sample_ptr()

```
float * m_delay_buffer_get_delayed_sample_ptr (
            m_delay_buffer * buf,
            unsigned int delay)
```

4.100.1.5 m_delay_buffer_get_fractional_delayed_sample()

```
int m_delay_buffer_get_fractional_delayed_sample (
            m_delay_buffer * buf,
            float * dest,
            float delay)
```

4.100.1.6 m_delay_buffer_resize_milliseconds()

```
int m_delay_buffer_resize_milliseconds (
            m_delay_buffer * buf,
            float ms)
```

4.100.1.7 m_delay_buffer_resize_samples()

```
int m_delay_buffer_resize_samples (
            m_delay_buffer * buf,
            int n_samples)
```

4.100.1.8 m_delay_buffer_resize_seconds()

```
int m_delay_buffer_resize_seconds (
            m_delay_buffer * buf,
            float seconds)
```

4.100.1.9 m_delay_buffer_tick()

```
int m_delay_buffer_tick (
            m_delay_buffer * buf,
            float new_sample)
```

4.101 m_eng_dirty_octave.c File Reference

```
#include "m_eng.h"
```

Functions

- int reconfigure_dirty_octave (void *data_struct)
- int calc_dirty_octave (void *data_struct, float *dest, float *src, int n_samples)
- int init_dirty_octave_str (m_eng_dirty_octave_str *str)

4.101.1 Function Documentation

4.101.1.1 calc_dirty_octave()

```
int calc_dirty_octave (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.101.1.2 init_dirty_octave_str()

```
int init_dirty_octave_str (
            m_eng_dirty_octave_str * str)
```

4.101.1.3 reconfigure_dirty_octave()

```
int reconfigure_dirty_octave (
            void * data_struct)
```

4.102 m_eng_distortion.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_distortion_str (m_eng_distortion_str ∗str)
- int reconfigure_distortion (void ∗data_struct)
- int calc_distortion (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.102.1 Function Documentation

4.102.1.1 calc_distortion()

```
int calc_distortion (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.102.1.2 init_distortion_str()

```
int init_distortion_str (
            m_eng_distortion_str * str)
```

4.102.1.3 reconfigure_distortion()

```
int reconfigure_distortion (
            void * data_struct)
```

4.103 m_eng_envelope.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_envelope_str (m_eng_envelope_str *str)
- int reconfigure_envelope (void *data_struct)
- int calc_envelope (void *data_struct, float *dest, float *src, int n_samples)

4.103.1 Function Documentation

4.103.1.1 calc_envelope()

```
int calc_envelope (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.103.1.2 init_envelope_str()

```
int init_envelope_str (
            m_eng_envelope_str * str)
```

4.103.1.3 reconfigure_envelope()

```
int reconfigure_envelope (
            void * data_struct)
```

4.104 m_eng_equaliser.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_3_band_eq_str (m_eng_3_band_eq_str ∗str)
- int reconfigure_3_band_eq (void ∗data_struct)
- int calc_3_band_eq (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.104.1 Function Documentation

4.104.1.1 calc_3_band_eq()

```
int calc_3_band_eq (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.104.1.2 init_3_band_eq_str()

```
int init_3_band_eq_str (
            m_eng_3_band_eq_str * str)
```

4.104.1.3 reconfigure_3_band_eq()

```
int reconfigure_3_band_eq (
            void * data_struct)
```

4.105 m_eng_flanger.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_flanger_str (m_eng_flanger_str ∗str)
- int reconfigure_flanger (void ∗data_struct)
- int calc_flanger (void ∗data_struct, float ∗dest, float ∗src, int n_samples)
- int free_flanger_struct (void ∗data_struct)

4.105.1 Function Documentation

4.105.1.1 calc_flanger()

```
int calc_flanger (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.105.1.2 free_flanger_struct()

```
int free_flanger_struct (
            void * data_struct)
```

4.105.1.3 init_flanger_str()

```
int init_flanger_str (
            m_eng_flanger_str * str)
```

4.105.1.4 reconfigure_flanger()

```
int reconfigure_flanger (
            void * data_struct)
```

4.106 m_eng_globals.c File Reference

```
#include "m_eng.h"
```

Functions

- double cycles_to_seconds (uint64_t cycles)
- uint64_t current_cycle ()

Variables

- m_eng_context global_cxt
- uint16_t cpu_cycles_total = 0
- uint16_t cpu_cycles_total_max = 0
- uint16_t memory_used = 0
- uint16_t memory_used_max = 0
- int update_scheduled = 0
- uint32_t trace_depth = 0
- uint32_t cycles_upper = 0

4.106.1 Function Documentation

4.106.1.1 current_cycle()

```
uint64_t current_cycle ()  [inline]
```

4.106.1.2 cycles_to_seconds()

```
double cycles_to_seconds (
            uint64_t cycles)
```

4.106.2 Variable Documentation

4.106.2.1 cpu_cycles_total

```
uint16_t cpu_cycles_total = 0
```

4.106.2.2 cpu_cycles_total_max

```
uint16_t cpu_cycles_total_max = 0
```

4.106.2.3 cycles_upper

```
uint32_t cycles_upper = 0
```

4.106.2.4 global_cxt

```
m_eng_context global_cxt
```

4.106.2.5 memory_used

```
uint16_t memory_used = 0
```

4.106.2.6 memory_used_max

```
uint16_t memory_used_max = 0
```

4.106.2.7 trace_depth

```
uint32_t trace_depth = 0
```

4.106.2.8 update_scheduled

```
int update_scheduled = 0
```

4.107 m_eng_i2s_dma.cpp File Reference

```
#include <DMAChannel.h>
#include "utility/imxrt_hw.h"
#include "m_eng.h"
```

Functions

- DMAMEM __attribute__ ((aligned(32))) static uint32_t i2s_rx_buffer[AUDIO_BLOCK_SAMPLES]
- DMAChannel i2s_in_dma (false)
- DMAChannel i2s_out_dma (false)
- void configure_i2s_dma ()
- void init_i2s_dma ()
- void m_eng_i2s_input_isr ()
- void m_eng_i2s_output_isr ()
- void i2s_in_transmit (raw_sample_t *block, unsigned char index)
- void i2s_input_update ()
- void i2s_output_update ()
- void i2s_output_transmit_mono_int (raw_sample_t *block)
- void i2s_output_transmit_mono_float (float *block)

Variables

- raw_sample_t * i2s_in_block_left = NULL
- raw_sample_t * i2s_in_block_right = NULL
- uint16_t i2s_in_block_offset = 0
- bool i2s_in_update_responsibility = false
- raw_sample_t * i2s_out_block_left_1st = NULL
- raw_sample_t * i2s_out_block_right_1st = NULL
- raw_sample_t * i2s_out_block_left_2nd = NULL
- raw_sample_t * i2s_out_block_right_2nd = NULL
- uint16_t i2s_out_block_left_offset = 0
- uint16_t i2s_out_block_right_offset = 0
- bool i2s_out_update_responsibility = false
- raw_sample_t i2s_input_blocks [2][AUDIO_BLOCK_SAMPLES]
- raw_sample_t i2s_output_blocks [2][AUDIO_BLOCK_SAMPLES]

4.107.1 Function Documentation

4.107.1.1 __attribute__()

```
DMAMEM __attribute__ (
             (aligned(32)) )
```

4.107.1.2 configure_i2s_dma()

```
void configure_i2s_dma ()
```

4.107.1.3 i2s_in_dma()

```
DMAChannel i2s_in_dma (
             false )
```

4.107.1.4 i2s_in_transmit()

```
void i2s_in_transmit (
            raw_sample_t * block,
            unsigned char index)
```

4.107.1.5 i2s_input_update()

```
void i2s_input_update ()
```

4.107.1.6 i2s_out_dma()

```
DMAChannel i2s_out_dma (
            false )
```

4.107.1.7 i2s_output_transmit_mono_float()

```
void i2s_output_transmit_mono_float (
            float * block)
```

4.107.1.8 i2s_output_transmit_mono_int()

```
void i2s_output_transmit_mono_int (
            raw_sample_t * block)
```

4.107.1.9 i2s_output_update()

```
void i2s_output_update ()
```

4.107.1.10 init_i2s_dma()

```
void init_i2s_dma ()
```

4.107.1.11 m_eng_i2s_input_isr()

```
void m_eng_i2s_input_isr ()
```

4.107.1.12 m_eng_i2s_output_isr()

```
void m_eng_i2s_output_isr ()
```

4.107.2 Variable Documentation

4.107.2.1 i2s_in_block_left

```
raw_sample_t* i2s_in_block_left = NULL
```

4.107.2.2 i2s_in_block_offset

```
uint16_t i2s_in_block_offset = 0
```

4.107.2.3 i2s_in_block_right

```
raw_sample_t* i2s_in_block_right = NULL
```

4.107.2.4 i2s_in_update_responsibility

```
bool i2s_in_update_responsibility = false
```

4.107.2.5 i2s_input_blocks

```
raw_sample_t i2s_input_blocks[2][AUDIO_BLOCK_SAMPLES]
```

4.107.2.6 i2s_out_block_left_1st

```
raw_sample_t* i2s_out_block_left_1st = NULL
```

4.107.2.7 i2s_out_block_left_2nd

```
raw_sample_t* i2s_out_block_left_2nd = NULL
```

4.107.2.8 i2s_out_block_left_offset

```
uint16_t i2s_out_block_left_offset = 0
```

4.107.2.9 i2s_out_block_right_1st

```
raw_sample_t* i2s_out_block_right_1st = NULL
```

4.107.2.10 i2s_out_block_right_2nd

```
raw_sample_t* i2s_out_block_right_2nd = NULL
```

4.107.2.11 i2s_out_block_right_offset

```
uint16_t i2s_out_block_right_offset = 0
```

4.107.2.12 i2s_out_update_responsibility

```
bool i2s_out_update_responsibility = false
```

4.107.2.13 i2s_output_blocks

```
raw_sample_t i2s_output_blocks[2][AUDIO_BLOCK_SAMPLES]
```

4.108 m_eng_linkowitz_riley.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_lr_low_pass_filter_str (m_lr_low_pass_filter_str ∗str)
- int reconfigure_lr_low_pass_filter (void ∗data_struct)
- int calc_lr_low_pass_filter (void ∗data_struct, float ∗dest, float ∗src, int n_samples)
- int init_lr_high_pass_filter_str (m_lr_high_pass_filter_str ∗str)
- int reconfigure_lr_high_pass_filter (void ∗data_struct)
- int calc_lr_high_pass_filter (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.108.1 Function Documentation

4.108.1.1 calc_lr_high_pass_filter()

```
int calc_lr_high_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.108.1.2 calc_lr_low_pass_filter()

```
int calc_lr_low_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.108.1.3 init_lr_high_pass_filter_str()

```
int init_lr_high_pass_filter_str (
            m_lr_high_pass_filter_str * str)
```

4.108.1.4 init_lr_low_pass_filter_str()

```
int init_lr_low_pass_filter_str (
            m_lr_low_pass_filter_str * str)
```

4.108.1.5 reconfigure_lr_high_pass_filter()

```
int reconfigure_lr_high_pass_filter (
            void * data_struct)
```

4.108.1.6 reconfigure_lr_low_pass_filter()

```
int reconfigure_lr_low_pass_filter (
            void * data_struct)
```

4.109 m_eng_logging.cpp File Reference

```
#include <Arduino.h>
#include "m_eng.h"
```

Macros

- #define M_ENG_LOG_ENTRIES_N 64
- #define MESSAGE_BEGIN_COL 43
- #define LOG_ENTRIES_PRINT_BUF_LEN 4096
- #define M_ENG_PROFILER_ARRAY_N 256
- #define M_ENG_PROFILER_RA_CYCLES_ALPHA 0.9

Functions

- int format_log_entry (int index, char *buf, int max_len, int max_indent)
- void m_eng_trace_log_begin (const char *fname, const char *line, const char *function, int local_trace_↩ depth, uint64_t cycle)
- void m_eng_trace_log_return (const char *fname, const char *line, const char *function, int local_trace_↩ depth, uint64_t cycle)
- void m_eng_log_error_code (const char *fname, const char *line, const char *function, int error_code, uint64_t cycle)
- void m_eng_log_return_err (const char *fname, const char *line, const char *function, int error_code, uint64_t cycle)
- void m_eng_log_return_ptr (const char *fname, const char *line, const char *function, void *ptr, uint64_t cycle)
- void m_eng_log_return_int (const char *fname, const char *line, const char *function, int val, uint64_t cycle)
- void m_eng_log_return_ (const char *fname, const char *line, const char *function, uint64_t cycle)
- void m_eng_log_message (const char *fname, const char *line, const char *function, uint64_t cycle, const char *fmt,...)
- void m_eng_print_flush_log ()
- void m_eng_init_profiler ()
- void m_eng_profiler_log_entry (const char *function_name)
- void m_eng_profiler_log_return (const char *function_name, uint64_t cycle)
- void m_eng_profiler_sort ()
- void m_eng_profiler_print ()

4.109.1 Macro Definition Documentation

4.109.1.1 LOG_ENTRIES_PRINT_BUF_LEN

```
#define LOG_ENTRIES_PRINT_BUF_LEN 4096
```

4.109.1.2 M_ENG_LOG_ENTRIES_N

```
#define M_ENG_LOG_ENTRIES_N 64
```

4.109.1.3 M_ENG_PROFILER_ARRAY_N

```
#define M_ENG_PROFILER_ARRAY_N 256
```

4.109.1.4 M_ENG_PROFILER_RA_CYCLES_ALPHA

```
#define M_ENG_PROFILER_RA_CYCLES_ALPHA 0.9
```

4.109.1.5 MESSAGE_BEGIN_COL

```
#define MESSAGE_BEGIN_COL 43
```

4.109.2 Function Documentation

4.109.2.1 format_log_entry()

```
int format_log_entry (
            int index,
            char * buf,
            int max_len,
            int max_indent)
```

4.109.2.2 m_eng_init_profiler()

```
void m_eng_init_profiler ()
```

4.109.2.3 m_eng_log_error_code()

```
void m_eng_log_error_code (
            const char * fname,
            const char * line,
            const char * function,
            int error_code,
            uint64_t cycle)
```

4.109.2.4 m_eng_log_message()

```
void m_eng_log_message (
            const char * fname,
            const char * line,
            const char * function,
            uint64_t cycle,
            const char * fmt,
             ...)
```

4.109.2.5 m_eng_log_return_()

```
void m_eng_log_return_ (
            const char * fname,
            const char * line,
            const char * function,
            uint64_t cycle)
```

4.109.2.6 m_eng_log_return_err()

```
void m_eng_log_return_err (
            const char * fname,
            const char * line,
            const char * function,
            int error_code,
            uint64_t cycle)
```

4.109.2.7 m_eng_log_return_int()

```
void m_eng_log_return_int (
            const char * fname,
            const char * line,
            const char * function,
            int val,
            uint64_t cycle)
```

4.109.2.8 m_eng_log_return_ptr()

```
void m_eng_log_return_ptr (
            const char * fname,
            const char * line,
            const char * function,
            void * ptr,
            uint64_t cycle)
```

4.109.2.9 m_eng_print_flush_log()

```
void m_eng_print_flush_log ()
```

4.109.2.10 m_eng_profiler_log_entry()

```
void m_eng_profiler_log_entry (
            const char * function_name)
```

4.109.2.11 m_eng_profiler_log_return()

```
void m_eng_profiler_log_return (
            const char * function_name,
            uint64_t cycle)
```

4.109.2.12 m_eng_profiler_print()

```
void m_eng_profiler_print ()
```

4.109.2.13 m_eng_profiler_sort()

```
void m_eng_profiler_sort ()
```

4.109.2.14 m_eng_trace_log_begin()

```
void m_eng_trace_log_begin (
            const char * fname,
            const char * line,
            const char * function,
            int local_trace_depth,
            uint64_t cycle)
```

4.109.2.15 m_eng_trace_log_return()

```
void m_eng_trace_log_return (
            const char * fname,
            const char * line,
            const char * function,
            int local_trace_depth,
            uint64_t cycle)
```

4.110 m_eng_low_end_compressor.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_low_end_compressor_str (m_eng_low_end_compressor_str *str)
- int reconfigure_low_end_compressor (void *data_struct)
- int calc_low_end_compressor (void *data_struct, float *dest, float *src, int n_samples)

4.110.1 Function Documentation

4.110.1.1 calc_low_end_compressor()

```
int calc_low_end_compressor (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.110.1.2 init_low_end_compressor_str()

```
int init_low_end_compressor_str (
            m_eng_low_end_compressor_str * str)
```

4.110.1.3 reconfigure_low_end_compressor()

```
int reconfigure_low_end_compressor (
            void * data_struct)
```

4.111 m_eng_mempool.c File Reference

```
#include "m_eng.h"
```

Macros

- #define BUFFER_QUEUE_STATIC
- #define MEMPOOL_MALLOC_TRIES 6

Functions

- FLASHMEM void init_mem_pools ()
- float * allocate_buffer ()
- void release_buffer (float *buffer)
- void print_mempool_info ()

Variables

- int head = 0
- int tail = MEM_SIZE - 1
- int mem_pools_initialised = 0
- float buffer_pool [M_BUFFER_POOL_SIZE][AUDIO_BLOCK_SAMPLES]
- float * buffer_buffer [M_BUFFER_POOL_SIZE]
- int buffer_head = 0
- int buffer_tail = M_BUFFER_POOL_SIZE - 1
- float zero_buffer [AUDIO_BLOCK_SAMPLES]
- float sink_buffer [AUDIO_BLOCK_SAMPLES]

4.111.1 Macro Definition Documentation

4.111.1.1 BUFFER_QUEUE_STATIC

```
#define BUFFER_QUEUE_STATIC
```

4.111.1.2 MEMPOOL_MALLOC_TRIES

```
#define MEMPOOL_MALLOC_TRIES 6
```

4.111.2 Function Documentation

4.111.2.1 allocate_buffer()

```
float * allocate_buffer ()
```

4.111.2.2 init_mem_pools()

```
FLASHMEM void init_mem_pools ()
```

4.111.2.3 print_mempool_info()

```
void print_mempool_info ()
```

4.111.2.4 release_buffer()

```
void release_buffer (
            float * buffer)
```

4.111.3 Variable Documentation

4.111.3.1 buffer_buffer

```
float* buffer_buffer[M_BUFFER_POOL_SIZE]
```

4.111.3.2 buffer_head

```
int buffer_head = 0
```

4.111.3.3 buffer_pool

```
float buffer_pool[M_BUFFER_POOL_SIZE][AUDIO_BLOCK_SAMPLES]
```

4.111.3.4 buffer_tail

```
int buffer_tail = M_BUFFER_POOL_SIZE - 1
```

4.111.3.5 head

```
int head = 0
```

4.111.3.6 mem_pools_initialised

```
int mem_pools_initialised = 0
```

4.111.3.7 sink_buffer

```
float sink_buffer[AUDIO_BLOCK_SAMPLES]
```

4.111.3.8 tail

```
int tail = MEM_SIZE - 1
```

4.111.3.9 zero_buffer

```
float zero_buffer[AUDIO_BLOCK_SAMPLES]
```

4.112 m_eng_noise_suppressor.c File Reference

```
#include "m_eng.h"
```

Functions

- int reconfigure_noise_suppressor (void *data_struct)
- int calc_noise_suppressor (void *data_struct, float *dest, float *src, int n_samples)
- int init_noise_suppressor_str (m_eng_noise_suppressor_str *str)

4.112.1 Function Documentation

4.112.1.1 calc_noise_suppressor()

```
int calc_noise_suppressor (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.112.1.2 init_noise_suppressor_str()

```
int init_noise_suppressor_str (
            m_eng_noise_suppressor_str * str)
```

4.112.1.3 reconfigure_noise_suppressor()

```
int reconfigure_noise_suppressor (
            void * data_struct)
```

4.113 m_eng_parameter.c File Reference

```
#include "m_eng.h"
```

Functions

- void init_parameter (m_parameter *param, float initial, float min, float max, float max_jump, int scale)
- int init_setting (m_setting *setting, int16_t initial)
- int update_setting (m_setting *setting, uint16_t new_value)

4.113.1 Function Documentation

4.113.1.1 init_parameter()

```
void init_parameter (
            m_parameter * param,
            float initial,
            float min,
            float max,
            float max_jump,
            int scale)
```

4.113.1.2 init_setting()

```
int init_setting (
            m_setting * setting,
            int16_t initial)
```

4.113.1.3 update_setting()

```
int update_setting (
            m_setting * setting,
            uint16_t new_value)
```

4.114 m_eng_pass_filter.c File Reference

```
#include "m_eng.h"
```

Functions

- int reconfigure_low_pass_filter (void *data_struct)
- int calc_low_pass_filter (void *data_struct, float *dest, float *src, int n_samples)
- int init_low_pass_filter_str (m_eng_low_pass_filter_str *str)
- int reconfigure_high_pass_filter (void *data_struct)
- int calc_high_pass_filter (void *data_struct, float *dest, float *src, int n_samples)
- int init_high_pass_filter_str (m_eng_high_pass_filter_str *str)
- int reconfigure_band_pass_filter (void *data_struct)
- int calc_band_pass_filter (void *data_struct, float *dest, float *src, int n_samples)
- int init_band_pass_filter_str (m_eng_band_pass_filter_str *str)

4.114.1 Function Documentation

4.114.1.1 calc_band_pass_filter()

```
int calc_band_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.114.1.2 calc_high_pass_filter()

```
int calc_high_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.114.1.3 calc_low_pass_filter()

```
int calc_low_pass_filter (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.114.1.4 init_band_pass_filter_str()

```
int init_band_pass_filter_str (
            m_eng_band_pass_filter_str * str)
```

4.114.1.5 init_high_pass_filter_str()

```
int init_high_pass_filter_str (
            m_eng_high_pass_filter_str * str)
```

4.114.1.6 init_low_pass_filter_str()

```
int init_low_pass_filter_str (
            m_eng_low_pass_filter_str * str)
```

4.114.1.7 reconfigure_band_pass_filter()

```
int reconfigure_band_pass_filter (
            void * data_struct)
```

4.114.1.8 reconfigure_high_pass_filter()

```
int reconfigure_high_pass_filter (
            void * data_struct)
```

4.114.1.9 reconfigure_low_pass_filter()

```
int reconfigure_low_pass_filter (
            void * data_struct)
```

4.115 m_eng_percussifier.c File Reference

```
#include "m_eng.h"
```

Functions

- int reconfigure_percussifier (void *data_struct)
- int calc_percussifier (void *data_struct, float *dest, float *src, int n_samples)
- int init_percussifier_str (m_eng_percussifier_str *str)

4.115.1 Function Documentation

4.115.1.1 calc_percussifier()

```
int calc_percussifier (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.115.1.2 init_percussifier_str()

```
int init_percussifier_str (
            m_eng_percussifier_str * str)
```

4.115.1.3 reconfigure_percussifier()

```
int reconfigure_percussifier (
            void * data_struct)
```

4.116 m_eng_pipeline.c File Reference

```
#include "m_eng.h"
```

Macros

- #define TRANSFORMERS_MALLOC_CHUNK_SIZE 8

Functions

- int init_pipeline (m_pipeline *pipeline)
- int compute_pipeline (m_pipeline *pipeline, float *dest, float *src)
- int pipeline_expand_transformer_array (m_pipeline *pipeline)
- int pipeline_expand_transformer_array_to (m_pipeline *pipeline, int n)
- int pipeline_update_transition_policy (m_pipeline *pipeline)
- int pipeline_print_transformer_array (m_pipeline *pipeline)
- int pipeline_append_transformer (m_pipeline *pipeline, m_transformer *trans)
- int pipeline_remove_transformer (m_pipeline *pipeline, uint16_t tid)
- int pipeline_insert_transformer (m_pipeline *pipeline, m_transformer *trans, int pos)
- int pipeline_prepend_transformer (m_pipeline *pipeline, m_transformer *trans)
- int pipeline_append_transformer_type (m_pipeline *pipeline, uint16_t type)
- int pipeline_insert_transformer_type (m_pipeline *pipeline, uint16_t type, uint16_t pos)
- int pipeline_prepend_transformer_type (m_pipeline *pipeline, uint16_t type)
- int pipeline_get_transformer_position (m_pipeline *pipeline, uint16_t id)
- int pipeline_move_transformer (m_pipeline *pipeline, uint16_t id, int new_pos)
- m_transformer * pipeline_get_transformer_by_id (m_pipeline *pipeline, uint16_t id)
- int pipeline_swap_transformers (m_pipeline *pipeline, uint16_t id1, uint16_t id2)
- int pipeline_valid (m_pipeline *pipeline)
- int pipeline_compare (m_pipeline *pipeline_a, m_pipeline *pipeline_b)
- int gut_pipeline (m_pipeline *pipeline)
- int clone_pipeline (m_pipeline **dest_ptr, m_pipeline *src)
- int pipeline_clone_transformer_into_position (m_pipeline *pipeline, m_transformer *trans, int pos)
- int pipeline_change_transformer_setting (m_pipeline *pipeline, uint16_t tid, uint16_t sid, int16_t new_val)

4.116.1 Macro Definition Documentation

4.116.1.1 TRANSFORMERS_MALLOC_CHUNK_SIZE

```
#define TRANSFORMERS_MALLOC_CHUNK_SIZE 8
```

4.116.2 Function Documentation

4.116.2.1 clone_pipeline()

```
int clone_pipeline (
            m_pipeline ** dest_ptr,
            m_pipeline * src)
```

4.116.2.2 compute_pipeline()

```
int compute_pipeline (
            m_pipeline * pipeline,
            float * dest,
            float * src)
```

4.116.2.3 gut_pipeline()

```
int gut_pipeline (
            m_pipeline * pipeline)
```

4.116.2.4 init_pipeline()

```
int init_pipeline (
            m_pipeline * pipeline)
```

4.116.2.5 pipeline_append_transformer()

```
int pipeline_append_transformer (
            m_pipeline * pipeline,
            m_transformer * trans)
```

4.116.2.6 pipeline_append_transformer_type()

```
int pipeline_append_transformer_type (
            m_pipeline * pipeline,
            uint16_t type)
```

4.116.2.7 pipeline_change_transformer_setting()

```
int pipeline_change_transformer_setting (
            m_pipeline * pipeline,
            uint16_t tid,
            uint16_t sid,
            int16_t new_val)
```

4.116.2.8 pipeline_clone_transformer_into_position()

```
int pipeline_clone_transformer_into_position (
            m_pipeline * pipeline,
            m_transformer * trans,
            int pos)
```

4.116.2.9 pipeline_compare()

```
int pipeline_compare (
            m_pipeline * pipeline_a,
            m_pipeline * pipeline_b)
```

4.116.2.10 pipeline_expand_transformer_array()

```
int pipeline_expand_transformer_array (
            m_pipeline * pipeline)
```

4.116.2.11 pipeline_expand_transformer_array_to()

```
int pipeline_expand_transformer_array_to (
            m_pipeline * pipeline,
            int n)
```

4.116.2.12 pipeline_get_transformer_by_id()

```
m_transformer * pipeline_get_transformer_by_id (
            m_pipeline * pipeline,
            uint16_t id)
```

4.116.2.13 pipeline_get_transformer_position()

```
int pipeline_get_transformer_position (
            m_pipeline * pipeline,
            uint16_t id)
```

4.116.2.14 pipeline_insert_transformer()

```
int pipeline_insert_transformer (
            m_pipeline * pipeline,
            m_transformer * trans,
            int pos)
```

4.116.2.15 pipeline_insert_transformer_type()

```
int pipeline_insert_transformer_type (
            m_pipeline * pipeline,
            uint16_t type,
            uint16_t pos)
```

4.116.2.16 pipeline_move_transformer()

```
int pipeline_move_transformer (
            m_pipeline * pipeline,
            uint16_t id,
            int new_pos)
```

4.116.2.17 pipeline_prepend_transformer()

```
int pipeline_prepend_transformer (
            m_pipeline * pipeline,
            m_transformer * trans)
```

4.116.2.18 pipeline_prepend_transformer_type()

```
int pipeline_prepend_transformer_type (
            m_pipeline * pipeline,
            uint16_t type)
```

4.116.2.19 pipeline_print_transformer_array()

```
int pipeline_print_transformer_array (
            m_pipeline * pipeline)
```

4.116.2.20 pipeline_remove_transformer()

```
int pipeline_remove_transformer (
            m_pipeline * pipeline,
            uint16_t tid)
```

4.116.2.21 pipeline_swap_transformers()

```
int pipeline_swap_transformers (
            m_pipeline * pipeline,
            uint16_t id1,
            uint16_t id2)
```

4.116.2.22 pipeline_update_transition_policy()

```
int pipeline_update_transition_policy (
            m_pipeline * pipeline)
```

4.116.2.23 pipeline_valid()

```
int pipeline_valid (
            m_pipeline * pipeline)
```

4.117 m_eng_pipeline_mod.c File Reference

```
#include "m_eng.h"
```

Functions

- IMPLEMENT_LINKED_LIST (m_pipeline_mod)
- m_pipeline_mod create_pipeline_mod_append_transformer (uint16_t type)
- m_pipeline_mod create_pipeline_mod_move_transformer (uint16_t tid, uint16_t position)
- m_pipeline_mod create_pipeline_mod_remove_transformer (uint16_t tid)
- m_pipeline_mod create_pipeline_mod_change_transformer_setting (uint16_t tid, uint16_t setting_id, int16↩ _t new_value)
- const char * pipeline_mod_type_string (int type)
- int apply_pipeline_mod (m_pipeline *pipeline, m_pipeline_mod mod, int *err_code)

4.117.1 Function Documentation

4.117.1.1 apply_pipeline_mod()

```
int apply_pipeline_mod (
            m_pipeline * pipeline,
            m_pipeline_mod mod,
            int * err_code)
```

4.117.1.2 create_pipeline_mod_append_transformer()

```
m_pipeline_mod create_pipeline_mod_append_transformer (
            uint16_t type)
```

4.117.1.3 create_pipeline_mod_change_transformer_setting()

```
m_pipeline_mod create_pipeline_mod_change_transformer_setting (
            uint16_t tid,
            uint16_t setting_id,
            int16_t new_value)
```

4.117.1.4 create_pipeline_mod_move_transformer()

```
m_pipeline_mod create_pipeline_mod_move_transformer (
            uint16_t tid,
            uint16_t position)
```

4.117.1.5 create_pipeline_mod_remove_transformer()

```
m_pipeline_mod create_pipeline_mod_remove_transformer (
            uint16_t tid)
```

4.117.1.6 IMPLEMENT_LINKED_LIST()

```
IMPLEMENT_LINKED_LIST (
            m_pipeline_mod )
```

4.117.1.7 pipeline_mod_type_string()

```
const char * pipeline_mod_type_string (
            int type)
```

4.118 m_eng_printf.cpp File Reference

```
#include "m_eng.h"
```

Macros

- #define SPACING 7
- #define SPACING 7

Functions

- void m_printf (const char *fmt,...)
- void m_voice_printf (int who, const char *fmt,...)
- void m_mute_voice (int who)
- void m_unmute_voice (int who)
- void serial_print_blocks (int n,...)
- void pretty_print_block (int16_t *data, const char *start)
- void pretty_print_block_float (float *data, const char *start)

Variables

- const char * voice_colour_table [64]

4.118.1 Macro Definition Documentation

4.118.1.1 SPACING [1/2]

```
#define SPACING 7
```

4.118.1.2 SPACING [2/2]

```
#define SPACING 7
```

4.118.2 Function Documentation

4.118.2.1 m_mute_voice()

```
void m_mute_voice (
            int who)
```

4.118.2.2 m_printf()

```
void m_printf (
            const char * fmt,
             ...)
```

4.118.2.3 m_unmute_voice()

```
void m_unmute_voice (
            int who)
```

4.118.2.4 m_voice_printf()

```
void m_voice_printf (
            int who,
            const char * fmt,
             ...)
```

4.118.2.5 pretty_print_block()

```
void pretty_print_block (
            int16_t * data,
            const char * start)
```

4.118.2.6 pretty_print_block_float()

```
void pretty_print_block_float (
            float * data,
            const char * start)
```

4.118.2.7 serial_print_blocks()

```
void serial_print_blocks (
            int n,
             ...)
```

4.118.3 Variable Documentation

4.118.3.1 voice_colour_table

```
const char* voice_colour_table[64]
```

Initial value:

```
= {"\e[31m",
    "\e[34m", "\e[35m", "\e[36m", "\e[33m", "\e[34m", "\e[35m", "\e[36m", "\e[33m",
    "\e[34m", "\e[35m", "\e[36m", "\e[33m", "\e[34m", "\e[35m", "\e[36m", "\e[33m",
    "\e[34m", "\e[35m", "\e[36m", "\e[33m", "\e[34m", "\e[35m", "\e[36m", "\e[33m",
    "\e[34m", "\e[35m", "\e[36m", "\e[33m", "\e[34m", "\e[35m", "\e[36m", "\e[33m",
    "\e[34m", "\e[35m", "\e[36m", "\e[33m", "\e[34m", "\e[35m", "\e[36m", "\e[33m",
    "\e[34m", "\e[35m", "\e[36m", "\e[33m", "\e[34m", "\e[35m", "\e[36m", "\e[33m",
    "\e[34m", "\e[35m", "\e[36m", "\e[33m", "\e[34m", "\e[35m", "\e[36m", "\e[33m",
    "\e[34m", "\e[35m", "\e[36m", "\e[33m", "\e[34m", "\e[35m", "\e[36m"
}
```

4.119 m_eng_profile.c File Reference

```
#include "m_eng.h"
```

Functions

- int nullify_profile (m_eng_profile *profile)
- int init_profile (m_eng_profile *profile)
- int profile_print_job_list (m_eng_profile *profile)
- int profile_print_ujob_list (m_eng_profile *profile)
- int profile_apply_pipeline_mod (m_eng_profile *profile, m_pipeline_mod mod)

  Applies the given modification to the given profile.
- int profile_update (m_eng_profile *profile)
- int profile_process (m_eng_profile *profile, float *dest, float *src)
- int profile_trigger_pipeline_swap (m_eng_profile *profile)
- int profile_regenerate_back_pipeline (m_eng_profile *profile)
- int profile_scheduled_maintainance (m_eng_profile *profile)

4.119.1 Function Documentation

4.119.1.1 init_profile()

```
int init_profile (
            m_eng_profile * profile)
```

4.119.1.2 nullify_profile()

```
int nullify_profile (
            m_eng_profile * profile)
```

4.119.1.3 profile_apply_pipeline_mod()

```
int profile_apply_pipeline_mod (
            m_eng_profile * profile,
            m_pipeline_mod mod)
```

Applies the given modification to the given profile.

If the current profile is active, to prevent audio artifacts, pipeline modifications ("mods", encoded by m_pipeline_mod) are applied in the background - to an inactive copy of the profile's pipeline (the "back pipeline"), and the outputs from the front and back pipeline are smoothly cross-faded between, after which it can be safely applied to the (now back, previously front) pipeline, keeping the two in sync.

To achieve this, profile_apply_pipeline_mod calls apply_pipeline_mod to apply the mod to the back pipeline, calls profile_trigger_pipeline_swap to trigger the pipeline swap, and saves the mod in a linked list (m_eng_profile::jobs). After the swap is complete, profile_update will retrieve the job from the list and apply it to the back (previously front) pipeline.

If, however, when profile_apply_pipeline_mod is called, a pipeline swap is already in progress, the mod is *not* applied, but rather stored in a second linked list, (m_eng_profile::blocked_jobs), whereupon profile_update will apply it (after the swap is finished and any jobs waiting in (m_eng_profile::jobs) have been applied) by calling profile_apply_pipeline_mod.

Parameters

profile	The profile to be modified
mod	A struct describing the modification to apply

4.119.1.4 profile_print_job_list()

```
int profile_print_job_list (
            m_eng_profile * profile)
```

4.119.1.5 profile_print_ujob_list()

```
int profile_print_ujob_list (
            m_eng_profile * profile)
```

4.119.1.6 profile_process()

```
int profile_process (
            m_eng_profile * profile,
            float * dest,
            float * src)
```

4.119.1.7 profile_regenerate_back_pipeline()

```
int profile_regenerate_back_pipeline (
            m_eng_profile * profile)
```

4.119.1.8 profile_scheduled_maintainance()

```
int profile_scheduled_maintainance (
            m_eng_profile * profile)
```

4.119.1.9 profile_trigger_pipeline_swap()

```
int profile_trigger_pipeline_swap (
            m_eng_profile * profile)
```

4.119.1.10 profile_update()

```
int profile_update (
            m_eng_profile * profile)
```

4.120 m_eng_sgtl5000.cpp File Reference

```
#include <Arduino.h>
#include <Wire.h>
#include "m_eng.h"
#include "m_eng_sgtl5000_defs.h"
```

Functions

- int sgtl5000_volume (float n)
- int sgtl5000_mute_headphone ()
- int sgtl5000_unmute_headphone ()
- int sgtl5000_mute_line_out ()
- int sgtl5000_unmute_line_out ()
- void sgtl5000_set_address (uint8_t level)
- int sgtl5000_start ()
- int sgtl5000_enable ()
- unsigned int sgtl5000_read_reg (unsigned int reg)
- int sgtl5000_write_reg (unsigned int reg, unsigned int val)
- unsigned int sgtl5000_modify_reg (unsigned int reg, unsigned int val, unsigned int i_mask)
- int sgtl5000_volum_eng_integer (unsigned int n)
- int sgtl5000_line_in_level (uint8_t n)
- unsigned short sgtl5000_line_out_level (uint8_t n)
- unsigned short sgtl5000_adc_high_pass_filter_enable ()
- unsigned short sgtl5000_adc_high_pass_filter_freeze ()
- unsigned short sgtl5000_adc_high_pass_filter_disable ()
- void sgtl5000_kill_automation ()
- unsigned char calc_vol (float n, unsigned char range)
- unsigned short sgtl5000_dap_audio_eq_band (uint8_t band_num, float n)
- unsigned short sgtl5000_eq_select (uint8_t n)
- void sgtl5000_automate (uint8_t dap, uint8_t eq)

4.120.1 Function Documentation

4.120.1.1 calc_vol()

```
unsigned char calc_vol (
            float n,
            unsigned char range)
```

4.120.1.2 sgtl5000_adc_high_pass_filter_disable()

```
unsigned short sgtl5000_adc_high_pass_filter_disable ()
```

4.120.1.3 sgtl5000_adc_high_pass_filter_enable()

```
unsigned short sgtl5000_adc_high_pass_filter_enable ()
```

4.120.1.4 sgtl5000_adc_high_pass_filter_freeze()

```
unsigned short sgtl5000_adc_high_pass_filter_freeze ()
```

4.120.1.5 sgtl5000_automate()

```
void sgtl5000_automate (
            uint8_t dap,
            uint8_t eq)
```

4.120.1.6 sgtl5000_dap_audio_eq_band()

```
unsigned short sgtl5000_dap_audio_eq_band (
            uint8_t band_num,
            float n)
```

4.120.1.7 sgtl5000_enable()

```
int sgtl5000_enable ()
```

4.120.1.8 sgtl5000_eq_select()

```
unsigned short sgtl5000_eq_select (
            uint8_t n)
```

4.120.1.9 sgtl5000_kill_automation()

```
void sgtl5000_kill_automation ()
```

4.120.1.10 sgtl5000_line_in_level()

```
int sgtl5000_line_in_level (
            uint8_t n)
```

4.120.1.11 sgtl5000_line_out_level()

```
unsigned short sgtl5000_line_out_level (
            uint8_t n)
```

4.120.1.12 sgtl5000_modify_reg()

```
unsigned int sgtl5000_modify_reg (
            unsigned int reg,
            unsigned int val,
            unsigned int i_mask)
```

4.120.1.13 sgtl5000_mute_headphone()

```
int sgtl5000_mute_headphone ()
```

4.120.1.14 sgtl5000_mute_line_out()

```
int sgtl5000_mute_line_out ()
```

4.120.1.15 sgtl5000_read_reg()

```
unsigned int sgtl5000_read_reg (
            unsigned int reg)
```

4.120.1.16 sgtl5000_set_address()

```
void sgtl5000_set_address (
            uint8_t level)
```

4.120.1.17 sgtl5000_start()

```
int sgtl5000_start ()
```

4.120.1.18 sgtl5000_unmute_headphone()

```
int sgtl5000_unmute_headphone ()
```

4.120.1.19 sgtl5000_unmute_line_out()

```
int sgtl5000_unmute_line_out ()
```

4.120.1.20 sgtl5000_volum_eng_integer()

```
int sgtl5000_volum_eng_integer (
            unsigned int n)
```

4.120.1.21 sgtl5000_volume()

```
int sgtl5000_volume (
            float n)
```

4.120.1.22 sgtl5000_write_reg()

```
int sgtl5000_write_reg (
            unsigned int reg,
            unsigned int val)
```

4.121 m_eng_simple_distortion.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_simple_distortion_str (m_eng_simple_distortion_str ∗str)
- int reconfigure_simple_distortion (void ∗data_struct)
- int calc_simple_distortion (void ∗data_struct, float ∗dest, float ∗src, int n_samples)

4.121.1 Function Documentation

4.121.1.1 calc_simple_distortion()

```
int calc_simple_distortion (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.121.1.2 init_simple_distortion_str()

```
int init_simple_distortion_str (
            m_eng_simple_distortion_str * str)
```

4.121.1.3 reconfigure_simple_distortion()

```
int reconfigure_simple_distortion (
            void * data_struct)
```

4.122 m_eng_transformer.c File Reference

```
#include "m_eng.h"
```

Macros

- #define MAX_BLOCK_DIVIDER 8

Functions

- int transformer_init_parameter_array (m_transformer *trans, int n)
- int transformer_init_setting_array (m_transformer *trans, int n)
- int transformer_init_controls (m_transformer *trans)
- void run_bypass (float **dest, float **src, int n_inputs, int n_valid_inputs, int n_outputs)
- int run_transformer (m_transformer *trans, float *dest, float *src)
- int transformer_add_setting (m_transformer *trans, m_setting *setting)
- int transformer_add_parameter (m_transformer *trans, m_parameter *param)
- m_parameter * transformer_get_parameter (m_transformer *trans, uint16_t ppid)
- m_setting * transformer_get_setting (m_transformer *trans, uint16_t sid)
- void free_transformer (m_transformer *trans)
- int clone_transformer (m_transformer **dest_ptr, m_transformer *src)

4.122.1 Macro Definition Documentation

4.122.1.1 MAX_BLOCK_DIVIDER

```
#define MAX_BLOCK_DIVIDER 8
```

4.122.2 Function Documentation

4.122.2.1 clone_transformer()

```
int clone_transformer (
            m_transformer ** dest_ptr,
            m_transformer * src)
```

4.122.2.2 free_transformer()

```
void free_transformer (
            m_transformer * trans)
```

4.122.2.3 run_bypass()

```
void run_bypass (
            float ** dest,
            float ** src,
            int n_inputs,
            int n_valid_inputs,
            int n_outputs)
```

4.122.2.4 run_transformer()

```
int run_transformer (
            m_transformer * trans,
            float * dest,
            float * src)
```

4.122.2.5 transformer_add_parameter()

```
int transformer_add_parameter (
            m_transformer * trans,
            m_parameter * param)
```

4.122.2.6 transformer_add_setting()

```
int transformer_add_setting (
            m_transformer * trans,
            m_setting * setting)
```

4.122.2.7 transformer_get_parameter()

```
m_parameter * transformer_get_parameter (
            m_transformer * trans,
            uint16_t ppid)
```

4.122.2.8 transformer_get_setting()

```
m_setting * transformer_get_setting (
            m_transformer * trans,
            uint16_t sid)
```

4.122.2.9 transformer_init_controls()

```
int transformer_init_controls (
            m_transformer * trans)
```

4.122.2.10 transformer_init_parameter_array()

```
int transformer_init_parameter_array (
            m_transformer * trans,
            int n)
```

4.122.2.11 transformer_init_setting_array()

```
int transformer_init_setting_array (
            m_transformer * trans,
            int n)
```

4.123 m_eng_transformer_init.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_3_band_eq (m_transformer *trans)
- int init_amplifier (m_transformer *trans)
- int init_band_pass_filter (m_transformer *trans)
- int init_compressor (m_transformer *trans)
- int init_delay (m_transformer *trans)
- int init_dirty_octave (m_transformer *trans)
- int init_distortion (m_transformer *trans)
- int init_envelope (m_transformer *trans)
- int init_flanger (m_transformer *trans)
- int init_high_pass_filter (m_transformer *trans)
- int init_low_end_compressor (m_transformer *trans)
- int init_low_pass_filter (m_transformer *trans)
- int init_noise_suppressor (m_transformer *trans)
- int init_percussifier (m_transformer *trans)
- int init_warbler (m_transformer *trans)
- int init_transformer (m_transformer *trans, uint16_t type)

4.123.1 Function Documentation

4.123.1.1 init_3_band_eq()

```
int init_3_band_eq (
            m_transformer * trans)
```

4.123.1.2 init_amplifier()

```
int init_amplifier (
            m_transformer * trans)
```

4.123.1.3 init_band_pass_filter()

```
int init_band_pass_filter (
            m_transformer * trans)
```

4.123.1.4 init_compressor()

```
int init_compressor (
            m_transformer * trans)
```

4.123.1.5 init_delay()

```
int init_delay (
            m_transformer * trans)
```

4.123.1.6 init_dirty_octave()

```
int init_dirty_octave (
            m_transformer * trans)
```

4.123.1.7 init_distortion()

```
int init_distortion (
            m_transformer * trans)
```

4.123.1.8 init_envelope()

```
int init_envelope (
            m_transformer * trans)
```

4.123.1.9 init_flanger()

```
int init_flanger (
            m_transformer * trans)
```

4.123.1.10 init_high_pass_filter()

```
int init_high_pass_filter (
            m_transformer * trans)
```

4.123.1.11 init_low_end_compressor()

```
int init_low_end_compressor (
            m_transformer * trans)
```

4.123.1.12 init_low_pass_filter()

```
int init_low_pass_filter (
            m_transformer * trans)
```

4.123.1.13 init_noise_suppressor()

```
int init_noise_suppressor (
            m_transformer * trans)
```

4.123.1.14 init_percussifier()

```
int init_percussifier (
            m_transformer * trans)
```

4.123.1.15 init_transformer()

```
int init_transformer (
            m_transformer * trans,
            uint16_t type)
```

4.123.1.16 init_warbler()

```
int init_warbler (
            m_transformer * trans)
```

4.124 m_eng_transformer_template.c File Reference

4.125 m_eng_update.c File Reference

```
#include "m_eng.h"
```

Functions

- void update_all ()
- int update_setup ()
- void update_stop ()
- void m_eng_software_isr ()

4.125.1 Function Documentation

4.125.1.1 m_eng_software_isr()

```
void m_eng_software_isr ()
```

4.125.1.2 update_all()

```
void update_all ()
```

4.125.1.3 update_setup()

```
int update_setup ()
```

4.125.1.4 update_stop()

```
void update_stop ()
```

4.126 m_eng_useful_functions.c File Reference

```
#include "m_eng.h"
```

Macros

- #define DENORMAL_THRESHOLD 1e-30f
- #define FLOAT_TO_INT16_MAX (32767.0f / 32768.0f)
- #define SCALE_FACTOR 32768.0f
- #define MAX_INT 32768.0

Functions

- float identity_function (float x)
- float normalised_arctan (float x)
- float hard_clip (float x)
- float soft_fold (float x)
- float trig_transition_function (float x)
- int convert_block_int_to_float (float ∗dest, int16_t ∗src)
- int convert_block_float_to_int (int16_t ∗dest, float ∗src)

4.126.1 Macro Definition Documentation

4.126.1.1 DENORMAL_THRESHOLD

```
#define DENORMAL_THRESHOLD 1e-30f
```

4.126.1.2 FLOAT_TO_INT16_MAX

```
#define FLOAT_TO_INT16_MAX (32767.0f / 32768.0f)
```

4.126.1.3 MAX_INT

```
#define MAX_INT 32768.0
```

4.126.1.4 SCALE_FACTOR

```
#define SCALE_FACTOR 32768.0f
```

4.126.2 Function Documentation

4.126.2.1 convert_block_float_to_int()

```
int convert_block_float_to_int (
            int16_t * dest,
            float * src)
```

4.126.2.2 convert_block_int_to_float()

```
int convert_block_int_to_float (
            float * dest,
            int16_t * src)
```

4.126.2.3 hard_clip()

```
float hard_clip (
            float x)
```

4.126.2.4 identity_function()

```
float identity_function (
            float x)
```

4.126.2.5 normalised_arctan()

```
float normalised_arctan (
            float x)
```

4.126.2.6 soft_fold()

```
float soft_fold (
            float x)
```

4.126.2.7 trig_transition_function()

```
float trig_transition_function (
            float x)
```

4.127 m_eng_warbler.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_warbler_str (m_eng_warbler_str *str)
- int reconfigure_warbler (void *data_struct)
- int calc_warbler (void *data_struct, float *dest, float *src, int n_samples)

4.127.1 Function Documentation

4.127.1.1 calc_warbler()

```
int calc_warbler (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.127.1.2 init_warbler_str()

```
int init_warbler_str (
            m_eng_warbler_str * str)
```

4.127.1.3 reconfigure_warbler()

```
int reconfigure_warbler (
            void * data_struct)
```

4.128 m_eng_waveshaper.c File Reference

```
#include "m_eng.h"
```

Functions

- int init_waveshaper_str (m_eng_waveshaper_str *str)
- int calc_waveshaper (void *data_struct, float *dest, float *src, int n_samples)

4.128.1 Function Documentation

4.128.1.1 calc_waveshaper()

```
int calc_waveshaper (
            void * data_struct,
            float * dest,
            float * src,
            int n_samples)
```

4.128.1.2 init_waveshaper_str()

```
int init_waveshaper_str (
            m_eng_waveshaper_str * str)
```

4.129 m_alloc.c File Reference

Functions

- void * m_alloc (size_t size)
- void * m_realloc (void *ptr, size_t size)
- char * m_strndup (const char *str, size_t n)
- void m_free (void *ptr)
- void print_memory_report ()

4.129.1 Function Documentation

4.129.1.1 m_alloc()

```
void * m_alloc (
            size_t size)
```

4.129.1.2 m_free()

```
void m_free (
            void * ptr)
```

4.129.1.3 m_realloc()

```
void * m_realloc (
            void * ptr,
            size_t size)
```

4.129.1.4 m_strndup()

```
char * m_strndup (
            const char * str,
            size_t n)
```

4.129.1.5 print_memory_report()

```
void print_memory_report ()
```

4.130 m_alloc.h File Reference

Functions

- void * m_alloc (size_t size)
- void * m_realloc (void *ptr, size_t size)
- char * m_strndup (const char *str, size_t n)
- void m_free (void *ptr)
- void * m_int_lv_malloc (size_t size)
- void m_int_lv_free (void *ptr)
- void print_memory_report ()

4.130.1 Function Documentation

4.130.1.1 m_alloc()

```
void * m_alloc (
            size_t size)
```

4.130.1.2 m_free()

```
void m_free (
            void * ptr)
```

4.130.1.3 m_int_lv_free()

```
void m_int_lv_free (
            void * ptr)
```

4.130.1.4 m_int_lv_malloc()

```
void * m_int_lv_malloc (
            size_t size)
```

4.130.1.5 m_realloc()

```
void * m_realloc (
            void * ptr,
            size_t size)
```

4.130.1.6 m_strndup()

```
char * m_strndup (
            const char * str,
            size_t n)
```

4.130.1.7 print_memory_report()

```
void print_memory_report ()
```

4.131 m_alloc.h

Go to the documentation of this file.

```
00001 #ifndef M_MALLOC_WRAPPER_H_
00002 #define M_MALLOC_WRAPPER_H_
00003
00004 void *m_alloc(size_t size);
00005 void *m_realloc(void *ptr, size_t size);
00006 char *m_strndup(const char *str, size_t n);
00007 void m_free(void *ptr);
00008
00009 void *m_int_lv_malloc(size_t size);
00010 void m_int_lv_free(void *ptr);
00011
00012 void print_memory_report();
00013
00014 #endif
```

4.132 m_comms.c File Reference

```
#include "m_eng.h"
#include "m_comms.h"
#include "m_error_codes.h"
```

Functions

- uint8_t crc_8 (const uint8_t *data, int len)
- int et_message_data_length (m_message msg)
- int et_message_default_retries (uint16_t type)
- int encode_m_message (uint8_t *buf, m_message msg)
- m_message decode_m_message (uint8_t *bytes, unsigned int len)
- m_message create_m_message_nodata (uint16_t type)
- m_message create_m_message (uint16_t type, const char *fmt,...)
- int valid_m_message_type (uint8_t type)
- const char * m_message_code_to_string (uint16_t code)
- const char * m_response_code_to_string (uint16_t code)
- int te_message_data_length (m_response msg)
- int encode_m_response (uint8_t *buf, m_response msg)
- m_response decode_m_response (uint8_t *bytes, unsigned int len)
- int valid_m_response_type (uint8_t type)
- m_response create_m_response (uint16_t type, const char *fmt,...)
- m_response create_m_response_nodata (uint16_t type)
- m_response create_m_response_ok ()
- m_response create_m_response_parameter_value (uint16_t pid, uint16_t tid, uint16_t ppid, float val)
- m_response create_m_response_error (uint16_t error_code)
- m_response create_m_response_profile_id (uint16_t pid)
- m_response create_m_response_transformer_id (uint16_t pid, uint16_t tid)

4.132.1 Function Documentation

4.132.1.1 crc_8()

```
uint8_t crc_8 (
            const uint8_t * data,
            int len)
```

4.132.1.2 create_m_message()

```
m_message create_m_message (
            uint16_t type,
            const char * fmt,
             ...)
```

4.132.1.3 create_m_message_nodata()

```
m_message create_m_message_nodata (
            uint16_t type)
```

4.132.1.4 create_m_response()

```
m_response create_m_response (
            uint16_t type,
            const char * fmt,
             ...)
```

4.132.1.5 create_m_response_error()

```
m_response create_m_response_error (
            uint16_t error_code)
```

4.132.1.6 create_m_response_nodata()

```
m_response create_m_response_nodata (
            uint16_t type)
```

4.132.1.7 create_m_response_ok()

```
m_response create_m_response_ok ()
```

4.132.1.8 create_m_response_parameter_value()

```
m_response create_m_response_parameter_value (
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid,
            float val)
```

4.132.1.9 create_m_response_profile_id()

```
m_response create_m_response_profile_id (
            uint16_t pid)
```

4.132.1.10 create_m_response_transformer_id()

```
m_response create_m_response_transformer_id (
            uint16_t pid,
            uint16_t tid)
```

4.132.1.11 decode_m_message()

```
m_message decode_m_message (
            uint8_t * bytes,
            unsigned int len)
```

4.132.1.12 decode_m_response()

```
m_response decode_m_response (
            uint8_t * bytes,
            unsigned int len)
```

4.132.1.13 encode_m_message()

```
int encode_m_message (
            uint8_t * buf,
            m_message msg)
```

4.132.1.14 encode_m_response()

```
int encode_m_response (
            uint8_t * buf,
            m_response msg)
```

4.132.1.15 et_message_data_length()

```
int et_message_data_length (
            m_message msg)
```

4.132.1.16 et_message_default_retries()

```
int et_message_default_retries (
            uint16_t type)
```

4.132.1.17 m_message_code_to_string()

```
const char * m_message_code_to_string (
            uint16_t code)
```

4.132.1.18 m_response_code_to_string()

```
const char * m_response_code_to_string (
            uint16_t code)
```

4.132.1.19 te_message_data_length()

```
int te_message_data_length (
            m_response msg)
```

4.132.1.20 valid_m_message_type()

```
int valid_m_message_type (
            uint8_t type)
```

4.132.1.21 valid_m_response_type()

```
int valid_m_response_type (
            uint8_t type)
```

4.133 m_comms.h File Reference

Data Structures

- struct m_response
- struct m_message

Macros

- #define TEENSY_ADDR 0x08
- #define M_MESSAGE_NO_MESSAGE 255
- #define M_MESSAGE_CRC_FAIL 254
- #define M_MESSAGE_INVALID 0
- #define M_MESSAGE_HI 1
- #define M_MESSAGE_RESET 2
- #define M_MESSAGE_REBOOT 3
- #define M_MESSAGE_CREATE_PROFILE 4
- #define M_MESSAGE_APPEND_TRANSFORMER 5
- #define M_MESSAGE_MOVE_TRANSFORMER 6
- #define M_MESSAGE_REMOVE_TRANSFORMER 7
- #define M_MESSAGE_GET_N_PROFILES 8
- #define M_MESSAGE_GET_N_TRANSFORMERS 9
- #define M_MESSAGE_GET_TRANSFORMER_ID 10
- #define M_MESSAGE_GET_TRANSFORMER_TYPE 11
- #define M_MESSAGE_GET_N_PARAMETERS 12
- #define M_MESSAGE_GET_PARAM_VALUE 13
- #define M_MESSAGE_SET_PARAM_VALUE 14

- #define M_MESSAGE_GET_N_SETTINGS 15
- #define M_MESSAGE_GET_SETTING_VALUE 16
- #define M_MESSAGE_SET_SETTING_VALUE 17
- #define M_MESSAGE_STRING_CONTINUE 18
- #define M_MESSAGE_STRING_CONTINUING 19
- #define M_MESSAGE_SWITCH_PROFILE 20
- #define M_MESSAGE_DELETE_PROFILE 21
- #define M_MESSAGE_REPEAT_MESSAGE 22
- #define M_MESSAGE_ENTER_TUNER_MODE 23
- #define M_MESSAGE_EXIT_TUNER_MODE 24
- #define M_MESSAGE_TYPE_MAX M_MESSAGE_EXIT_TUNER_MODE
- #define MESSAGE_LEN_VARIABLE -2
- #define M_MESSAGE_MAX_DATA_LEN 16
- #define M_MESSAGE_MAX_TRANSFER_LEN (M_RESPONSE_MAX_DATA_LEN + 2)
- #define M_RESPONSE_MAX_DATA_LEN 16
- #define M_RESPONSE_MAX_TRANSFER_LEN (M_RESPONSE_MAX_DATA_LEN + 2)
- #define M_RESPONSE_NO_MESSAGE 255
- #define M_RESPONSE_CRC_FAIL 254
- #define M_RESPONSE_INVALID 0
- #define M_RESPONSE_WAIT 1
- #define M_RESPONSE_HI 2
- #define M_RESPONSE_BAD_MESSAGE 3
- #define M_RESPONSE_BAD_REQUEST 4
- #define M_RESPONSE_TRY_AGAIN 5
- #define M_RESPONSE_OK 6
- #define M_RESPONSE_ERROR 7
- #define M_RESPONSE_PROFILE_ID 8
- #define M_RESPONSE_TRANSFORMER_ID 10
- #define M_RESPONSE_N_PROFILES 11
- #define M_RESPONSE_N_TRANSFORMERS 12
- #define M_RESPONSE_TRANSFORMER_TYPE 13
- #define M_RESPONSE_N_PARAMETERS 14
- #define M_RESPONSE_PARAM_VALUE 15
- #define M_RESPONSE_N_SETTINGS 16
- #define M_RESPONSE_SETTING_VALUE 17
- #define M_RESPONSE_STRING_CONTINUING 18
- #define M_RESPONSE_START_OVER 19
- #define M_RESPONSE_SWITCHING_PROFILE 20
- #define M_RESPONSE_DELETED_PROFILE 21
- #define M_RESPONSE_REPEAT_MESSAGE 22
- #define M_RESPONSE_TYPE_MAX M_RESPONSE_REPEAT_MESSAGE

Functions

- m_message create_m_message_nodata (uint16_t type)
- m_message create_m_message (uint16_t type, const char *fmt,...)
- m_response create_m_response_nodata (uint16_t type)
- m_response create_m_response (uint16_t type, const char *fmt,...)
- m_response create_m_response_ok ()
- m_response create_m_response_error (uint16_t error_code)
- m_response create_m_response_profile_id (uint16_t pid)
- m_response create_m_response_transformer_id (uint16_t pid, uint16_t tid)
- m_response create_m_response_parameter_value (uint16_t pid, uint16_t tid, uint16_t ppid, float value)
- int et_message_data_length (m_message msg)

- int valid_m_message_type (uint8_t type)
- int encode_m_message (uint8_t *buf, m_message msg)
- m_message decode_m_message (uint8_t *bytes, unsigned int len)
- int te_message_data_length (m_response msg)
- int valid_m_response_type (uint8_t type)
- int encode_m_response (uint8_t *buf, m_response msg)
- m_response decode_m_response (uint8_t *bytes, unsigned int len)
- const char * m_message_code_to_string (uint16_t code)
- const char * m_response_code_to_string (uint16_t code)

4.133.1 Macro Definition Documentation

4.133.1.1 M_MESSAGE_APPEND_TRANSFORMER

```
#define M_MESSAGE_APPEND_TRANSFORMER 5
```

4.133.1.2 M_MESSAGE_CRC_FAIL

```
#define M_MESSAGE_CRC_FAIL 254
```

4.133.1.3 M_MESSAGE_CREATE_PROFILE

```
#define M_MESSAGE_CREATE_PROFILE 4
```

4.133.1.4 M_MESSAGE_DELETE_PROFILE

```
#define M_MESSAGE_DELETE_PROFILE 21
```

4.133.1.5 M_MESSAGE_ENTER_TUNER_MODE

```
#define M_MESSAGE_ENTER_TUNER_MODE 23
```

4.133.1.6 M_MESSAGE_EXIT_TUNER_MODE

```
#define M_MESSAGE_EXIT_TUNER_MODE 24
```

4.133.1.7 M_MESSAGE_GET_N_PARAMETERS

```
#define M_MESSAGE_GET_N_PARAMETERS 12
```

4.133.1.8 M_MESSAGE_GET_N_PROFILES

```
#define M_MESSAGE_GET_N_PROFILES 8
```

4.133.1.9 M_MESSAGE_GET_N_SETTINGS

```
#define M_MESSAGE_GET_N_SETTINGS 15
```

4.133.1.10 M_MESSAGE_GET_N_TRANSFORMERS

```
#define M_MESSAGE_GET_N_TRANSFORMERS 9
```

4.133.1.11 M_MESSAGE_GET_PARAM_VALUE

```
#define M_MESSAGE_GET_PARAM_VALUE 13
```

4.133.1.12 M_MESSAGE_GET_SETTING_VALUE

```
#define M_MESSAGE_GET_SETTING_VALUE 16
```

4.133.1.13 M_MESSAGE_GET_TRANSFORMER_ID

```
#define M_MESSAGE_GET_TRANSFORMER_ID 10
```

4.133.1.14 M_MESSAGE_GET_TRANSFORMER_TYPE

```
#define M_MESSAGE_GET_TRANSFORMER_TYPE 11
```

4.133.1.15 M_MESSAGE_HI

```
#define M_MESSAGE_HI 1
```

4.133.1.16 M_MESSAGE_INVALID

```
#define M_MESSAGE_INVALID 0
```

4.133.1.17 M_MESSAGE_MAX_DATA_LEN

```
#define M_MESSAGE_MAX_DATA_LEN 16
```

4.133.1.18 M_MESSAGE_MAX_TRANSFER_LEN

```
#define M_MESSAGE_MAX_TRANSFER_LEN (M_RESPONSE_MAX_DATA_LEN + 2)
```

4.133.1.19 M_MESSAGE_MOVE_TRANSFORMER

```
#define M_MESSAGE_MOVE_TRANSFORMER 6
```

4.133.1.20 M_MESSAGE_NO_MESSAGE

```
#define M_MESSAGE_NO_MESSAGE 255
```

4.133.1.21 M_MESSAGE_REBOOT

```
#define M_MESSAGE_REBOOT 3
```

4.133.1.22 M_MESSAGE_REMOVE_TRANSFORMER

```
#define M_MESSAGE_REMOVE_TRANSFORMER 7
```

4.133.1.23 M_MESSAGE_REPEAT_MESSAGE

```
#define M_MESSAGE_REPEAT_MESSAGE 22
```

4.133.1.24 M_MESSAGE_RESET

```
#define M_MESSAGE_RESET 2
```

4.133.1.25 M_MESSAGE_SET_PARAM_VALUE

```
#define M_MESSAGE_SET_PARAM_VALUE 14
```

4.133.1.26 M_MESSAGE_SET_SETTING_VALUE

```
#define M_MESSAGE_SET_SETTING_VALUE 17
```

4.133.1.27 M_MESSAGE_STRING_CONTINUE

```
#define M_MESSAGE_STRING_CONTINUE 18
```

4.133.1.28 M_MESSAGE_STRING_CONTINUING

```
#define M_MESSAGE_STRING_CONTINUING 19
```

4.133.1.29 M_MESSAGE_SWITCH_PROFILE

```
#define M_MESSAGE_SWITCH_PROFILE 20
```

4.133.1.30 M_MESSAGE_TYPE_MAX

```
#define M_MESSAGE_TYPE_MAX M_MESSAGE_EXIT_TUNER_MODE
```

4.133.1.31 M_RESPONSE_BAD_MESSAGE

```
#define M_RESPONSE_BAD_MESSAGE 3
```

4.133.1.32 M_RESPONSE_BAD_REQUEST

```
#define M_RESPONSE_BAD_REQUEST 4
```

4.133.1.33 M_RESPONSE_CRC_FAIL

```
#define M_RESPONSE_CRC_FAIL 254
```

4.133.1.34 M_RESPONSE_DELETED_PROFILE

```
#define M_RESPONSE_DELETED_PROFILE 21
```

4.133.1.35 M_RESPONSE_ERROR

```
#define M_RESPONSE_ERROR 7
```

4.133.1.36 M_RESPONSE_HI

```
#define M_RESPONSE_HI 2
```

4.133.1.37 M_RESPONSE_INVALID

```
#define M_RESPONSE_INVALID 0
```

4.133.1.38 M_RESPONSE_MAX_DATA_LEN

```
#define M_RESPONSE_MAX_DATA_LEN 16
```

4.133.1.39 M_RESPONSE_MAX_TRANSFER_LEN

```
#define M_RESPONSE_MAX_TRANSFER_LEN (M_RESPONSE_MAX_DATA_LEN + 2)
```

4.133.1.40 M_RESPONSE_N_PARAMETERS

```
#define M_RESPONSE_N_PARAMETERS 14
```

4.133.1.41 M_RESPONSE_N_PROFILES

```
#define M_RESPONSE_N_PROFILES 11
```

4.133.1.42 M_RESPONSE_N_SETTINGS

```
#define M_RESPONSE_N_SETTINGS 16
```

4.133.1.43 M_RESPONSE_N_TRANSFORMERS

```
#define M_RESPONSE_N_TRANSFORMERS 12
```

4.133.1.44 M_RESPONSE_NO_MESSAGE

```
#define M_RESPONSE_NO_MESSAGE 255
```

4.133.1.45 M_RESPONSE_OK

```
#define M_RESPONSE_OK 6
```

4.133.1.46 M_RESPONSE_PARAM_VALUE

```
#define M_RESPONSE_PARAM_VALUE 15
```

4.133.1.47 M_RESPONSE_PROFILE_ID

```
#define M_RESPONSE_PROFILE_ID 8
```

4.133.1.48 M_RESPONSE_REPEAT_MESSAGE

```
#define M_RESPONSE_REPEAT_MESSAGE 22
```

4.133.1.49 M_RESPONSE_SETTING_VALUE

```
#define M_RESPONSE_SETTING_VALUE 17
```

4.133.1.50 M_RESPONSE_START_OVER

```
#define M_RESPONSE_START_OVER 19
```

4.133.1.51 M_RESPONSE_STRING_CONTINUING

```
#define M_RESPONSE_STRING_CONTINUING 18
```

4.133.1.52 M_RESPONSE_SWITCHING_PROFILE

```
#define M_RESPONSE_SWITCHING_PROFILE 20
```

4.133.1.53 M_RESPONSE_TRANSFORMER_ID

```
#define M_RESPONSE_TRANSFORMER_ID 10
```

4.133.1.54 M_RESPONSE_TRANSFORMER_TYPE

```
#define M_RESPONSE_TRANSFORMER_TYPE 13
```

4.133.1.55 M_RESPONSE_TRY_AGAIN

```
#define M_RESPONSE_TRY_AGAIN 5
```

4.133.1.56 M_RESPONSE_TYPE_MAX

```
#define M_RESPONSE_TYPE_MAX M_RESPONSE_REPEAT_MESSAGE
```

4.133.1.57 M_RESPONSE_WAIT

```
#define M_RESPONSE_WAIT 1
```

4.133.1.58 MESSAGE_LEN_VARIABLE

```
#define MESSAGE_LEN_VARIABLE -2
```

4.133.1.59 TEENSY_ADDR

```
#define TEENSY_ADDR 0x08
```

4.133.2 Function Documentation

4.133.2.1 create_m_message()

```
m_message create_m_message (
            uint16_t type,
            const char * fmt,
             ...)
```

4.133.2.2 create_m_message_nodata()

```
m_message create_m_message_nodata (
            uint16_t type)
```

4.133.2.3 create_m_response()

```
m_response create_m_response (
            uint16_t type,
            const char * fmt,
             ...)
```

4.133.2.4 create_m_response_error()

```
m_response create_m_response_error (
            uint16_t error_code)
```

4.133.2.5 create_m_response_nodata()

```
m_response create_m_response_nodata (
            uint16_t type)
```

4.133.2.6 create_m_response_ok()

```
m_response create_m_response_ok ()
```

4.133.2.7 create_m_response_parameter_value()

```
m_response create_m_response_parameter_value (
            uint16_t pid,
            uint16_t tid,
            uint16_t ppid,
            float value)
```

4.133.2.8 create_m_response_profile_id()

```
m_response create_m_response_profile_id (
            uint16_t pid)
```

4.133.2.9 create_m_response_transformer_id()

```
m_response create_m_response_transformer_id (
            uint16_t pid,
            uint16_t tid)
```

4.133.2.10 decode_m_message()

```
m_message decode_m_message (
            uint8_t * bytes,
            unsigned int len)
```

4.133.2.11 decode_m_response()

```
m_response decode_m_response (
            uint8_t * bytes,
            unsigned int len)
```

4.133.2.12 encode_m_message()

```
int encode_m_message (
            uint8_t * buf,
            m_message msg)
```

4.133.2.13 encode_m_response()

```
int encode_m_response (
            uint8_t * buf,
            m_response msg)
```

4.133.2.14 et_message_data_length()

```
int et_message_data_length (
            m_message msg)
```

4.133.2.15 m_message_code_to_string()

```
const char * m_message_code_to_string (
            uint16_t code)
```

4.133.2.16 m_response_code_to_string()

```
const char * m_response_code_to_string (
            uint16_t code)
```

4.133.2.17 te_message_data_length()

```
int te_message_data_length (
            m_response msg)
```

4.133.2.18 valid_m_message_type()

```
int valid_m_message_type (
            uint8_t type)
```

4.133.2.19 valid_m_response_type()

```
int valid_m_response_type (
            uint8_t type)
```

4.134 m_comms.h

Go to the documentation of this file.

```
00001 #ifndef M_COMMS_H_
00002 #define M_COMMS_H_
00003
00004 #ifndef TEENSY_ADDR
00005 #define TEENSY_ADDR 0x08
00006 #endif
00007
00008 /* Messages from ESP32 to Teensy */
00009
00010 #define M_MESSAGE_NO_MESSAGE               255
00011 #define M_MESSAGE_CRC_FAIL                 254
00012 #define M_MESSAGE_INVALID                  0
00013 #define M_MESSAGE_HI                       1
00014 #define M_MESSAGE_RESET                    2
00015 #define M_MESSAGE_REBOOT                   3
00016 #define M_MESSAGE_CREATE_PROFILE           4
00017 #define M_MESSAGE_APPEND_TRANSFORMER       5
00018 #define M_MESSAGE_MOVE_TRANSFORMER         6
00019 #define M_MESSAGE_REMOVE_TRANSFORMER       7
00020 #define M_MESSAGE_GET_N_PROFILES           8
00021 #define M_MESSAGE_GET_N_TRANSFORMERS       9
00022 #define M_MESSAGE_GET_TRANSFORMER_ID       10
00023 #define M_MESSAGE_GET_TRANSFORMER_TYPE     11
00024 #define M_MESSAGE_GET_N_PARAMETERS         12
00025 #define M_MESSAGE_GET_PARAM_VALUE          13
00026 #define M_MESSAGE_SET_PARAM_VALUE          14
00027 #define M_MESSAGE_GET_N_SETTINGS           15
00028 #define M_MESSAGE_GET_SETTING_VALUE        16
00029 #define M_MESSAGE_SET_SETTING_VALUE        17
00030 #define M_MESSAGE_STRING_CONTINUE          18
00031 #define M_MESSAGE_STRING_CONTINUING        19
00032 #define M_MESSAGE_SWITCH_PROFILE           20
00033 #define M_MESSAGE_DELETE_PROFILE           21
00034 #define M_MESSAGE_REPEAT_MESSAGE           22
00035 #define M_MESSAGE_ENTER_TUNER_MODE         23
00036 #define M_MESSAGE_EXIT_TUNER_MODE          24
00037
00038 #define M_MESSAGE_TYPE_MAX M_MESSAGE_EXIT_TUNER_MODE
00039
00040 #define MESSAGE_LEN_VARIABLE               -2
```

```
00041 #define M_MESSAGE_MAX_DATA_LEN 16
00042 #define M_MESSAGE_MAX_TRANSFER_LEN (M_RESPONSE_MAX_DATA_LEN + 2)
00043
00044 #define M_RESPONSE_MAX_DATA_LEN 16
00045 #define M_RESPONSE_MAX_TRANSFER_LEN (M_RESPONSE_MAX_DATA_LEN + 2)
00046
00047 typedef struct
00048 {
00049     uint8_t type;
00050     uint8_t data[M_RESPONSE_MAX_DATA_LEN];
00051     void *extra;
00052 } m_response;
00053
00054 typedef struct m_message
00055 {
00056     uint8_t type;
00057     uint8_t data[M_RESPONSE_MAX_DATA_LEN];
00058     void (*callback)(struct m_message msg, m_response response);
00059     void *cb_arg;
00060     int retries;
00061 } m_message;
00062
00063 m_message create_m_message_nodata(uint16_t type);
00064 m_message create_m_message(uint16_t type, const char *fmt, ...);
00065
00066 /* Messages from Teensy to ESP32 */
00067
00068 #define M_RESPONSE_NO_MESSAGE           255
00069 #define M_RESPONSE_CRC_FAIL             254
00070 #define M_RESPONSE_INVALID              0
00071 #define M_RESPONSE_WAIT                 1
00072 #define M_RESPONSE_HI                   2
00073 #define M_RESPONSE_BAD_MESSAGE          3
00074 #define M_RESPONSE_BAD_REQUEST          4
00075 #define M_RESPONSE_TRY_AGAIN            5
00076 #define M_RESPONSE_OK                   6
00077 #define M_RESPONSE_ERROR                7
00078 #define M_RESPONSE_PROFILE_ID           8
00079 #define M_RESPONSE_TRANSFORMER_ID       10
00080 #define M_RESPONSE_N_PROFILES           11
00081 #define M_RESPONSE_N_TRANSFORMERS       12
00082 #define M_RESPONSE_TRANSFORMER_TYPE     13
00083 #define M_RESPONSE_N_PARAMETERS         14
00084 #define M_RESPONSE_PARAM_VALUE          15
00085 #define M_RESPONSE_N_SETTINGS           16
00086 #define M_RESPONSE_SETTING_VALUE        17
00087 #define M_RESPONSE_STRING_CONTINUING    18
00088 #define M_RESPONSE_START_OVER           19
00089 #define M_RESPONSE_SWITCHING_PROFILE    20
00090 #define M_RESPONSE_DELETED_PROFILE      21
00091 #define M_RESPONSE_REPEAT_MESSAGE       22
00092
00093 #define M_RESPONSE_TYPE_MAX M_RESPONSE_REPEAT_MESSAGE
00094
00095 m_response create_m_response_nodata(uint16_t type);
00096 m_response create_m_response(uint16_t type, const char *fmt, ...);
00097 m_response create_m_response_ok();
00098
00099 m_response create_m_response_error(uint16_t error_code);
00100 m_response create_m_response_profile_id(uint16_t pid);
00101 m_response create_m_response_transformer_id(uint16_t pid, uint16_t tid);
00102 m_response create_m_response_parameter_value(uint16_t pid, uint16_t tid, uint16_t ppid, float value);
00103
00104 int et_message_data_length(m_message msg);
00105 int valid_m_message_type(uint8_t type);
00106
00107 int encode_m_message(uint8_t *buf, m_message msg);
00108 m_message decode_m_message(uint8_t *bytes, unsigned int len);
00109
00110 int te_message_data_length(m_response msg);
00111
00112 int valid_m_response_type(uint8_t type);
00113 int encode_m_response(uint8_t *buf, m_response msg);
00114 m_response decode_m_response(uint8_t *bytes, unsigned int len);
00115
00116 const char *m_message_code_to_string(uint16_t code);
00117 const char *m_response_code_to_string(uint16_t code);
00118
00119 #endif
```

4.135 m_error_codes.c File Reference

```
#include "m_error_codes.h"
```

Functions

- const char * m_error_code_to_string (int error_code)

4.135.1 Function Documentation

4.135.1.1 m_error_code_to_string()

```
const char * m_error_code_to_string (
            int error_code)
```

4.136 m_error_codes.h File Reference

Macros

- #define NO_ERROR 0
- #define ERR_NULL_PTR 1
- #define ERR_BAD_ARGS 2
- #define ERR_SGTL5000_WRITE_FAIL 3
- #define ERR_ALLOC_FAIL 5
- #define ERR_PIPELINE_NULL 4
- #define ERR_PIPELINE_FULL 6
- #define ERR_POSITION_ILLEGAL 7
- #define ERR_POSITION_OCCUPIED 8
- #define ERR_TRANSFORMER_MALFORMED 9
- #define ERR_ARRAY_MALFORMED 10
- #define ERR_POT_LINK_MALFORMED 11
- #define ERR_SWITCH_LINK_MALFORMED 12
- #define ERR_MUTEX_UNAVAILABLE 13
- #define ERR_FIXED_ARRAY_FULL 14
- #define ERR_BUSTED_MSG 15
- #define ERR_BAD_REQUEST 16
- #define ERR_QUEUE_SEND_FAILED 17
- #define ERR_QUEUE_FULL 18
- #define ERR_LOOP_DETECTED 19
- #define ERR_NODE_PRIVATE 20
- #define ERR_PIPELINE_BUSTED 21
- #define ERR_INVALID_MESSAGE 22
- #define ERR_VALUE_OUT_OF_BOUNDS 23
- #define ERR_INVALID_PARAMETER_ID 24
- #define ERR_INVALID_SETTING_ID 25
- #define ERR_INVALID_TRANSFORMER_ID 26
- #define ERR_INVALID_PROFILE_ID 27
- #define ERR_INCONSISTENT_BACK_PIPELINE 28

- #define ERR_SPI_INIT_FAIL 29
- #define ERR_SD_INIT_FAIL 30
- #define ERR_SD_MOUNT_FAIL 31
- #define ERR_FOPEN_FAIL 32
- #define ERR_UNFINISHED_WRITE 33
- #define ERR_MANGLED_FILE 34
- #define ERR_I2C_FAIL 35
- #define ERR_NO_RESPONSE 36
- #define ERR_COMMS_FAIL 37
- #define ERR_UNKNOWN_ERR 4999
- #define ERR_UNIMPLEMENTED 5000

Functions

- const char * m_error_code_to_string (int error_code)

4.136.1 Macro Definition Documentation

4.136.1.1 ERR_ALLOC_FAIL

```
#define ERR_ALLOC_FAIL 5
```

4.136.1.2 ERR_ARRAY_MALFORMED

```
#define ERR_ARRAY_MALFORMED 10
```

4.136.1.3 ERR_BAD_ARGS

```
#define ERR_BAD_ARGS 2
```

4.136.1.4 ERR_BAD_REQUEST

```
#define ERR_BAD_REQUEST 16
```

4.136.1.5 ERR_BUSTED_MSG

```
#define ERR_BUSTED_MSG 15
```

4.136.1.6 ERR_COMMS_FAIL

```
#define ERR_COMMS_FAIL 37
```

4.136.1.7 ERR_FIXED_ARRAY_FULL

```
#define ERR_FIXED_ARRAY_FULL 14
```

4.136.1.8 ERR_FOPEN_FAIL

```
#define ERR_FOPEN_FAIL 32
```

4.136.1.9 ERR_I2C_FAIL

```
#define ERR_I2C_FAIL 35
```

4.136.1.10 ERR_INCONSISTENT_BACK_PIPELINE

```
#define ERR_INCONSISTENT_BACK_PIPELINE 28
```

4.136.1.11 ERR_INVALID_MESSAGE

```
#define ERR_INVALID_MESSAGE 22
```

4.136.1.12 ERR_INVALID_PARAMETER_ID

```
#define ERR_INVALID_PARAMETER_ID 24
```

4.136.1.13 ERR_INVALID_PROFILE_ID

```
#define ERR_INVALID_PROFILE_ID 27
```

4.136.1.14 ERR_INVALID_SETTING_ID

```
#define ERR_INVALID_SETTING_ID 25
```

4.136.1.15 ERR_INVALID_TRANSFORMER_ID

```
#define ERR_INVALID_TRANSFORMER_ID 26
```

4.136.1.16 ERR_LOOP_DETECTED

```
#define ERR_LOOP_DETECTED 19
```

4.136.1.17 ERR_MANGLED_FILE

```
#define ERR_MANGLED_FILE 34
```

4.136.1.18 ERR_MUTEX_UNAVAILABLE

```
#define ERR_MUTEX_UNAVAILABLE 13
```

4.136.1.19 ERR_NO_RESPONSE

```
#define ERR_NO_RESPONSE 36
```

4.136.1.20 ERR_NODE_PRIVATE

```
#define ERR_NODE_PRIVATE 20
```

4.136.1.21 ERR_NULL_PTR

```
#define ERR_NULL_PTR 1
```

4.136.1.22 ERR_PIPELINE_BUSTED

```
#define ERR_PIPELINE_BUSTED 21
```

4.136.1.23 ERR_PIPELINE_FULL

```
#define ERR_PIPELINE_FULL 6
```

4.136.1.24 ERR_PIPELINE_NULL

```
#define ERR_PIPELINE_NULL 4
```

4.136.1.25 ERR_POSITION_ILLEGAL

```
#define ERR_POSITION_ILLEGAL 7
```

4.136.1.26 ERR_POSITION_OCCUPIED

```
#define ERR_POSITION_OCCUPIED 8
```

4.136.1.27 ERR_POT_LINK_MALFORMED

```
#define ERR_POT_LINK_MALFORMED 11
```

4.136.1.28 ERR_QUEUE_FULL

```
#define ERR_QUEUE_FULL 18
```

4.136.1.29 ERR_QUEUE_SEND_FAILED

```
#define ERR_QUEUE_SEND_FAILED 17
```

4.136.1.30 ERR_SD_INIT_FAIL

```
#define ERR_SD_INIT_FAIL 30
```

4.136.1.31 ERR_SD_MOUNT_FAIL

```
#define ERR_SD_MOUNT_FAIL 31
```

4.136.1.32 ERR_SGTL5000_WRITE_FAIL

```
#define ERR_SGTL5000_WRITE_FAIL 3
```

4.136.1.33 ERR_SPI_INIT_FAIL

```
#define ERR_SPI_INIT_FAIL 29
```

4.136.1.34 ERR_SWITCH_LINK_MALFORMED

```
#define ERR_SWITCH_LINK_MALFORMED 12
```

4.136.1.35 ERR_TRANSFORMER_MALFORMED

```
#define ERR_TRANSFORMER_MALFORMED 9
```

4.136.1.36 ERR_UNFINISHED_WRITE

```
#define ERR_UNFINISHED_WRITE 33
```

4.136.1.37 ERR_UNIMPLEMENTED

```
#define ERR_UNIMPLEMENTED 5000
```

4.136.1.38 ERR_UNKNOWN_ERR

```
#define ERR_UNKNOWN_ERR 4999
```

4.136.1.39 ERR_VALUE_OUT_OF_BOUNDS

```
#define ERR_VALUE_OUT_OF_BOUNDS 23
```

4.136.1.40 NO_ERROR

```
#define NO_ERROR 0
```

4.136.2 Function Documentation

4.136.2.1 m_error_code_to_string()

```
const char * m_error_code_to_string (
            int error_code)
```

4.137 m_error_codes.h

Go to the documentation of this file.

```
00001 #ifndef M_ERROR_CODES_H_
00002 #define M_ERROR_CODES_H_
00003
00004 #define NO_ERROR                        0
00005 #define ERR_NULL_PTR                    1
00006 #define ERR_BAD_ARGS                    2
00007 #define ERR_SGTL5000_WRITE_FAIL         3
00008
00009 #define ERR_ALLOC_FAIL                  5
00010
00011 #define ERR_PIPELINE_NULL               4
00012 #define ERR_PIPELINE_FULL               6
00013 #define ERR_POSITION_ILLEGAL            7
00014 #define ERR_POSITION_OCCUPIED           8
00015
00016 #define ERR_TRANSFORMER_MALFORMED       9
00017 #define ERR_ARRAY_MALFORMED             10
00018
00019 #define ERR_POT_LINK_MALFORMED          11
00020 #define ERR_SWITCH_LINK_MALFORMED       12
00021
00022 #define ERR_MUTEX_UNAVAILABLE           13
00023
00024 #define ERR_FIXED_ARRAY_FULL            14
00025
00026 #define ERR_BUSTED_MSG                  15
00027 #define ERR_BAD_REQUEST                 16
00028
00029 #define ERR_QUEUE_SEND_FAILED           17
00030 #define ERR_QUEUE_FULL                  18
00031 #define ERR_LOOP_DETECTED               19
00032
```

```
00033 #define ERR_NODE_PRIVATE                20
00034 #define ERR_PIPELINE_BUSTED             21
00035
00036 #define ERR_INVALID_MESSAGE             22
00037
00038 #define ERR_VALUE_OUT_OF_BOUNDS         23
00039
00040 #define ERR_INVALID_PARAMETER_ID        24
00041 #define ERR_INVALID_SETTING_ID          25
00042 #define ERR_INVALID_TRANSFORMER_ID      26
00043 #define ERR_INVALID_PROFILE_ID          27
00044
00045 #define ERR_INCONSISTENT_BACK_PIPELINE  28
00046
00047 #define ERR_SPI_INIT_FAIL               29
00048 #define ERR_SD_INIT_FAIL                30
00049 #define ERR_SD_MOUNT_FAIL               31
00050 #define ERR_FOPEN_FAIL                  32
00051 #define ERR_UNFINISHED_WRITE            33
00052 #define ERR_MANGLED_FILE                34
00053
00054 #define ERR_I2C_FAIL                    35
00055 #define ERR_NO_RESPONSE                 36
00056 #define ERR_COMMS_FAIL                  37
00057
00058 #define ERR_UNKNOWN_ERR                 4999
00059 #define ERR_UNIMPLEMENTED               5000
00060
00061 const char *m_error_code_to_string(int error_code);
00062
00063 #endif
```

4.138 m_linked_list.h File Reference

Macros

- #define LL_FREE free
- #define LL_MALLOC malloc
- #define DECLARE_LINKED_LIST(X)
- #define IMPLEMENT_LINKED_LIST(X)
- #define DECLARE_LINKED_PTR_LIST(X)
- #define IMPLEMENT_LINKED_PTR_LIST(X)

4.138.1 Macro Definition Documentation

4.138.1.1 DECLARE_LINKED_LIST

```
#define DECLARE_LINKED_LIST(
              X)
```

Value:

```
struct X##_ll;
      \
typedef struct X##_ll {
      \
    X data;
      \
    struct X##_ll *next;
      \
} X##_ll;
      \

X##_ll *X##_ll_new(X x);
      \
void free_##X##_ll(X##_ll *list);
      \
X##_ll *X##_ll_tail(X##_ll *list);
      \
```

```
X##_ll *X##_ll_append(X##_ll *list, X x);
      \
X##_ll *X##_ll_append_return_tail(X##_ll **list, X x);
      \
X##_ll *X##_ll_remove_next(X##_ll *list);
      \
void destructor_free_##X##_ll(X##_ll *list, void (*destructor)(X x));
      \
void X##_ll_map(X##_ll *list, X (*fmap)(X x));
      \
X##_ll *X##_ll_cmp_search(X##_ll *list, int (*cmp_function)(X, X), X x);
      \
X##_ll *X##_ll_destructor_free_and_remove_matching(X##_ll *list, int (*cmp_function)(X, X), X x, void
      (*destructor)(X));
```

4.138.1.2 DECLARE_LINKED_PTR_LIST

```
#define DECLARE_LINKED_PTR_LIST(
              X)
```

Value:

```
struct X##_pll;                                                                                    \
typedef struct X##_pll {
      \
    X *data;
      \
    struct X##_pll *next;
      \
} X##_pll;
      \

X##_pll *X##_pll_new(X *value);
      \
void free_##X##_pll(X##_pll *list);
      \
X##_pll *X##_pll_tail(X##_pll *list);
      \
X##_pll *X##_pll_append(X##_pll *list, X *value);
      \
X##_pll *X##_pll_append_return_tail(X##_pll **list, X *x);
      \
X##_pll *X##_pll_remove_next(X##_pll *list);
      \
void X##_pll_free_all(X##_pll *list);
      \
void destructor_free_##X##_pll(X##_pll *list, void (*destructor)(X *x));
      \
X##_pll *X##_pll_cmp_search(X##_pll *list, int (*cmp_function)(const X*, const X*), const X *x);
      \
X##_pll *X##_pll_destructor_free_and_remove_matching(X##_pll *list, int (*cmp_function)(X*, X*), X *x, void
      (*destructor)(X*));
```

4.138.1.3 IMPLEMENT_LINKED_LIST

```
#define IMPLEMENT_LINKED_LIST(
              X)
```

4.138.1.4 IMPLEMENT_LINKED_PTR_LIST

```
#define IMPLEMENT_LINKED_PTR_LIST(
              X)
```

4.138.1.5 LL_FREE

```
#define LL_FREE free
```

4.138.1.6 LL_MALLOC

```
#define LL_MALLOC malloc
```

4.139 m_linked_list.h

Go to the documentation of this file.

```
00001 #ifndef LINKED_LIST_H
00002 #define LINKED_LIST_H
00003
00004 #ifndef LL_FREE
00005 #define LL_FREE free
00006 #endif
00007
00008 #ifndef LL_MALLOC
00009 #define LL_MALLOC malloc
00010 #endif
00011
00012 #define DECLARE_LINKED_LIST(X) struct X##_ll;
      \
00013 typedef struct X##_ll {
      \
00014     X data;
      \
00015     struct X##_ll *next;
      \
00016 } X##_ll;
      \
00017
      \
00018 X##_ll *X##_ll_new(X x);
      \
00019 void free_##X##_ll(X##_ll *list);
      \
00020 X##_ll *X##_ll_tail(X##_ll *list);
      \
00021 X##_ll *X##_ll_append(X##_ll *list, X x);
      \
00022 X##_ll *X##_ll_append_return_tail(X##_ll **list, X x);
      \
00023 X##_ll *X##_ll_remove_next(X##_ll *list);
      \
00024 void destructor_free_##X##_ll(X##_ll *list, void (*destructor)(X x));
      \
00025 void X##_ll_map(X##_ll *list, X (*fmap)(X x));
      \
00026 X##_ll *X##_ll_cmp_search(X##_ll *list, int (*cmp_function)(X, X), X x);
      \
00027 X##_ll *X##_ll_destructor_free_and_remove_matching(X##_ll *list, int (*cmp_function)(X, X), X x, void
      (*destructor)(X));
00028
00029 #define IMPLEMENT_LINKED_LIST(X)
      \
00030 X##_ll *X##_ll_new(X x)
      \
00031 {
      \
00032     X##_ll *result = (X##_ll*)LL_MALLOC(sizeof(X##_ll));
      \
00033
      \
00034     if (result == NULL)
      \
00035         return NULL;
      \
00036
      \
00037     result->data = x;
      \
00038     result->next = NULL;
      \
00039
      \
00040     return result;
      \
00041 }
      \
00042
      \
```

```
00043 X##_ll *X##_ll_tail(X##_ll *list)
    \
00044 {
    \
00045     if (list == NULL)
    \
00046         return NULL;
    \
00047 
    \
00048     X##_ll *current = list;
    \
00049 
    \
00050     while (current->next != NULL)
    \
00051         current = current->next;
    \
00052 
    \
00053     return current;
    \
00054 }
    \
00055 
    \
00056 void free_##X##_ll(X##_ll *list)
    \
00057 {
    \
00058     if (list == NULL)
    \
00059         return;
    \
00060     X##_ll *current = list;
    \
00061     X##_ll *next = NULL;
    \
00062     while (current != NULL) {
    \
00063         next = current->next;
    \
00064         LL_FREE(current);
    \
00065         current = next;
    \
00066     }
    \
00067 }
    \
00068 
    \
00069 void destructor_free_##X##_ll(X##_ll *list, void (*destructor)(X x))
    \
00070 {
    \
00071     if (list == NULL)
    \
00072         return;
    \
00073 
    \
00074     X##_ll *current = list;
    \
00075     X##_ll *next = NULL;
    \
00076 
    \
00077     while (current != NULL) {
    \
00078         next = current->next;
    \
00079         destructor(current->data);
    \
00080         LL_FREE(current);
    \
00081         current = next;
    \
00082     }
    \
00083 }
    \
00084 
    \
00085 X##_ll *X##_ll_append(X##_ll *list, X x)
    \
00086 {
```

```
\
00087     X##_ll *next = X##_ll_new(x);
\
00088     if (list == NULL) return next;
\
00089
\
00090     X##_ll *current = list;
\
00091
\
00092     while (current->next != NULL)
\
00093         current = current->next;
\
00094
\
00095     current->next = next;
\
00096
\
00097     return list;
\
00098 }
\
00099
\
00100 X##_ll *X##_ll_append_return_tail(X##_ll **list, X x)
\
00101 {
\
00102     if (list == NULL)
\
00103         return NULL;
\
00104
\
00105     X##_ll *next = X##_ll_new(x);
\
00106     if (*list == NULL) {
\
00107         *list = next;
\
00108         return next;
\
00109     }
\
00110
\
00111     X##_ll *current = *list;
\
00112
\
00113     while (current->next != NULL)
\
00114         current = current->next;
\
00115
\
00116     current->next = next;
\
00117
\
00118     return next;
\
00119 }
\
00120
\
00121 X##_ll *X##_ll_remove_next(X##_ll *list)
\
00122 {
\
00123     if (list == NULL)
\
00124         return NULL;
\
00125
\
00126     X##_ll *next = list->next;
\
00127
\
00128     if (next == NULL)
\
00129         return NULL;
\
```

```
00130
    \
00131     list->next = list->next->next;
    \
00132
    \
00133     return next;
    \
00134 }
    \
00135
    \
00136 void X##_ll_map(X##_ll *list, X (*fmap)(X x))
    \
00137 {
    \
00138     if (list == NULL)
    \
00139         return;
    \
00140
    \
00141     list->data = fmap(list->data);
    \
00142     X##_ll_map(list->next, fmap);
    \
00143 }
    \
00144
    \
00145 X##_ll *X##_ll_cmp_search(X##_ll *list, int (*cmp_function)(X, X), X x)
    \
00146 {
    \
00147     if (list == NULL)
    \
00148         return NULL;
    \
00149
    \
00150     X##_ll *current = list;
    \
00151
    \
00152     while (current) {
    \
00153         if (cmp_function(current->data, x) == 0)
    \
00154             return current;
    \
00155
    \
00156         current = current->next;
    \
00157     }
    \
00158
    \
00159     return NULL;
    \
00160 }
    \
00161
    \
00162 X##_ll *X##_ll_destructor_free_and_remove_matching(X##_ll *list, int (*cmp_function)(X, X), X x, void
    (*destructor)(X))     \
00163 {
    \
00164     if (list == NULL)
    \
00165         return NULL;
    \
00166
    \
00167     X##_ll *current = list;
    \
00168     X##_ll *prev = NULL;
    \
00169     X##_ll *next = NULL;
    \
00170
    \
00171     while (current) {
    \
00172         next = current->next;
    \
00173         if (cmp_function(current->data, x) == 0) {
```

```
    \
00174             if (current == list)
    \
00175                 list = next;
    \
00176             destructor(current->data);
    \
00177             LL_FREE(current);
    \
00178             if (prev)
    \
00179                 prev->next = next;
    \
00180         } else {
    \
00181             prev = current;
    \
00182         }
    \
00183         current = next;
    \
00184     }
    \
00185 
    \
00186     return list;
    \
00187 }
00188 
00189 #define DECLARE_LINKED_PTR_LIST(X) struct X##_pll;
    \
00190 typedef struct X##_pll {
    \
00191     X *data;
    \
00192     struct X##_pll *next;
    \
00193 } X##_pll;
    \
00194 
    \
00195 X##_pll *X##_pll_new(X *value);
    \
00196 void free_##X##_pll(X##_pll *list);
    \
00197 X##_pll *X##_pll_tail(X##_pll *list);
    \
00198 X##_pll *X##_pll_append(X##_pll *list, X *value);
    \
00199 X##_pll *X##_pll_append_return_tail(X##_pll **list, X *x);
    \
00200 X##_pll *X##_pll_remove_next(X##_pll *list);
    \
00201 void X##_pll_free_all(X##_pll *list);
    \
00202 void destructor_free_##X##_pll(X##_pll *list, void (*destructor)(X *x));
    \
00203 X##_pll *X##_pll_cmp_search(X##_pll *list, int (*cmp_function)(const X*, const X*), const X *x);
    \
00204 X##_pll *X##_pll_destructor_free_and_remove_matching(X##_pll *list, int (*cmp_function)(X*, X*), X *x,
    void (*destructor)(X*));
00205 
00206 #define IMPLEMENT_LINKED_PTR_LIST(X)
    \
00207 
    \
00208 X##_pll *X##_pll_new(X *value)
    \
00209 {
    \
00210     X##_pll *result = (X##_pll*)LL_MALLOC(sizeof(X##_pll));
    \
00211 
    \
00212     if (result == NULL)
    \
00213         return NULL;
    \
00214 
    \
00215     result->data = value;
    \
00216     result->next = NULL;
    \
00217 
    \
00218     return result;
```

```
    \
00219 }
    \
00220
    \
00221 X##_pll *X##_pll_tail(X##_pll *list)
    \
00222 {
    \
00223     if (list == NULL)
    \
00224         return NULL;
    \
00225
    \
00226     X##_pll *current = list;
    \
00227
    \
00228     while (current->next != NULL)
    \
00229         current = current->next;
    \
00230
    \
00231     return current;
    \
00232 }
    \
00233
    \
00234 X##_pll *X##_pll_append(X##_pll *list, X *x)
    \
00235 {
    \
00236     X##_pll *next = X##_pll_new(x);
    \
00237     if (list == NULL) return next;
    \
00238
    \
00239     X##_pll *current = list;
    \
00240
    \
00241     while (current->next != NULL)
    \
00242         current = current->next;
    \
00243
    \
00244     current->next = next;
    \
00245
    \
00246     return list;
    \
00247 }
    \
00248
    \
00249 X##_pll *X##_pll_append_return_tail(X##_pll **list, X *x)
    \
00250 {
    \
00251     if (list == NULL)
    \
00252         return NULL;
    \
00253
    \
00254     X##_pll *next = X##_pll_new(x);
    \
00255     if (*list == NULL) {
    \
00256         *list = next;
    \
00257         return next;
    \
00258     }
    \
00259
    \
00260     X##_pll *current = *list;
    \
00261
    \
```

```
00262     while (current->next != NULL)
\
00263         current = current->next;
\
00264
\
00265     current->next = next;
\
00266
\
00267     return next;
\
00268 }
\
00269
\
00270
\
00271 void free_##X##_pll(X##_pll *list)
\
00272 {
\
00273     if (list == NULL)
\
00274         return;
\
00275     X##_pll *current = list;
\
00276     X##_pll *next = NULL;
\
00277     while (current != NULL) {
\
00278         next = current->next;
\
00279         LL_FREE(current->data);
\
00280         LL_FREE(current);
\
00281         current = next;
\
00282     }
\
00283 }
\
00284
\
00285 void destructor_free_##X##_pll(X##_pll *list, void (*destructor)(X *x))
\
00286 {
\
00287     if (list == NULL)
\
00288         return;
\
00289
\
00290     X##_pll *current = list;
\
00291     X##_pll *next = NULL;
\
00292     while (current != NULL) {
\
00293         next = current->next;
\
00294         destructor(current->data);
\
00295         LL_FREE(current);
\
00296         current = next;
\
00297     }
\
00298 }
\
00299
\
00300 X##_pll *X##_pll_remove_next(X##_pll *list)
\
00301 {
\
00302     if (list == NULL)
\
00303         return NULL;
\
00304
\
00305     X##_pll *next = list->next;
```

```
\
00306
\
00307     if (next == NULL)
\
00308         return NULL;
\
00309
\
00310     list->next = next->next;
\
00311
\
00312     return next;
\
00313 }
\
00314
\
00315 void X##_pll_map(X##_pll *list, X *(*fmap)(X *x))
\
00316 {
\
00317     if (list == NULL)
\
00318         return;
\
00319
\
00320     list->data = fmap(list->data);
\
00321     X##_pll_map(list->next, fmap);
\
00322 }
\
00323
\
00324 X##_pll *X##_pll_cmp_search(X##_pll *list, int (*cmp_function)(const X*, const X*), const X *x)
\
00325 {
\
00326     if (list == NULL)
\
00327         return NULL;
\
00328
\
00329     X##_pll *current = list;
\
00330
\
00331     while (current) {
\
00332         if (cmp_function(current->data, x) == 0)
\
00333             return current;
\
00334
\
00335         current = current->next;
\
00336     }
\
00337
\
00338     return NULL;
\
00339 }
\
00340
\
00341
\
00342 X##_pll *X##_pll_destructor_free_and_remove_matching(X##_pll *list, int (*cmp_function)(X*, X*), X *x,
    void (*destructor)(X*))\
00343 {
\
00344     if (list == NULL)
\
00345         return NULL;
\
00346
\
00347     X##_pll *current = list;
\
00348     X##_pll *next = NULL;
\
```

```
00349     X##_pll *prev = NULL;
\
00350
\
00351     while (current) {
\
00352         next = current->next;
\
00353         if (cmp_function(current->data, x) == 0) {
\
00354             if (current == list)
\
00355                 list = next;
\
00356             destructor(current->data);
\
00357             LL_FREE(current);
\
00358             if (prev)
\
00359                 prev->next = next;
\
00360         } else {
\
00361             prev = current;
\
00362         }
\
00363         current = next;
\
00364     }
\
00365
\
00366     return list;
\
00367 }
00368
00369 #endif
```

4.140 m_parameter.h File Reference

Data Structures

- struct m_parameter_id
- struct m_parameter
- struct m_setting_id
- struct m_setting

Macros

- #define PARAMETER_SCALE_LINEAR 0
- #define PARAMETER_SCALE_LOGARITHMIC 1
- #define TRANSFORMER_SETTING_ENUM 0
- #define TRANSFORMER_SETTING_BOOL 1
- #define TRANSFORMER_SETTING_INT 2
- #define TRANSFORMER_SETTING_PAGE_SETTINGS 0
- #define TRANSFORMER_SETTING_PAGE_MAIN 1

Functions

- DECLARE_LINKED_PTR_LIST (m_parameter)
- DECLARE_LINKED_PTR_LIST (m_setting)

4.140.1 Macro Definition Documentation

4.140.1.1 PARAMETER_SCALE_LINEAR

```
#define PARAMETER_SCALE_LINEAR 0
```

4.140.1.2 PARAMETER_SCALE_LOGARITHMIC

```
#define PARAMETER_SCALE_LOGARITHMIC 1
```

4.140.1.3 TRANSFORMER_SETTING_BOOL

```
#define TRANSFORMER_SETTING_BOOL 1
```

4.140.1.4 TRANSFORMER_SETTING_ENUM

```
#define TRANSFORMER_SETTING_ENUM 0
```

4.140.1.5 TRANSFORMER_SETTING_INT

```
#define TRANSFORMER_SETTING_INT 2
```

4.140.1.6 TRANSFORMER_SETTING_PAGE_MAIN

```
#define TRANSFORMER_SETTING_PAGE_MAIN 1
```

4.140.1.7 TRANSFORMER_SETTING_PAGE_SETTINGS

```
#define TRANSFORMER_SETTING_PAGE_SETTINGS 0
```

4.140.2 Function Documentation

4.140.2.1 DECLARE_LINKED_PTR_LIST() [1/2]

```
DECLARE_LINKED_PTR_LIST (
            m_parameter )
```

4.140.2.2 DECLARE_LINKED_PTR_LIST() [2/2]

```
DECLARE_LINKED_PTR_LIST (
            m_setting )
```

4.141 m_parameter.h

Go to the documentation of this file.

```
00001 #ifndef M_PARAMETER_H_
00002 #define M_PARAMETER_H_
00003
00004 #define PARAMETER_SCALE_LINEAR      0
00005 #define PARAMETER_SCALE_LOGARITHMIC 1
00006
00007 typedef struct m_parameter_id
00008 {
00009     uint16_t profile_id;
00010     uint16_t transformer_id;
00011     uint16_t parameter_id;
00012 } m_parameter_id;
00013
00014 typedef struct m_parameter
00015 {
00016     float value;
00017     float min;
00018     float max;
00019
00020     int scale;
00021
00022     int updated;
00023     float old_value;
00024     float new_value;
00025
00026     #if defined(M_ENGINE)
00027
00028     float max_jump;
00029
00030     #elif defined(M_INTERFACE)
00031
00032     m_parameter_id id;
00033
00034     float factor;
00035
00036     int widget_type;
00037     const char *name;
00038     const char *units;
00039
00040     int group;
00041
00042     #endif
00043 } m_parameter;
00044
00045 #if defined(M_INTERFACE)
00046
00047 typedef struct m_setting_option
00048 {
00049     uint16_t value;
00050     const char *name;
00051 } m_setting_option;
00052
00053 #endif
00054
00055 typedef struct m_setting_id
00056 {
00057     uint16_t profile_id;
00058     uint16_t transformer_id;
00059     uint16_t setting_id;
00060 } m_setting_id;
00061
00062 #define TRANSFORMER_SETTING_ENUM    0
00063 #define TRANSFORMER_SETTING_BOOL    1
00064 #define TRANSFORMER_SETTING_INT     2
00065
00066 #define TRANSFORMER_SETTING_PAGE_SETTINGS 0
00067 #define TRANSFORMER_SETTING_PAGE_MAIN     1
00068
00069 typedef struct m_setting
00070 {
00071     int16_t value;
00072
00073     int updated;
00074     int16_t old_value;
00075     int16_t new_value;
00076
00077     #if defined(M_ENGINE)
00078
00079
00080     #elif defined(M_INTERFACE)
00081
00082     int type;
```

```
00083     int page;
00084
00085     m_setting_id id;
00086
00087     uint16_t min;
00088     uint16_t max;
00089
00090     int n_options;
00091     m_setting_option *options;
00092
00093     int widget_type;
00094     const char *name;
00095     const char *units;
00096
00097     int group;
00098
00099     #endif
00100 } m_setting;
00101
00102 DECLARE_LINKED_PTR_LIST(m_parameter);
00103 DECLARE_LINKED_PTR_LIST(m_setting);
00104
00105 #endif
```

4.142 m_pipeline.h File Reference

Data Structures

- struct m_pipeline

4.143 m_pipeline.h

Go to the documentation of this file.

```
00001 #ifndef M_PIPELINE_H_
00002 #define M_PIPELINE_H_
00003
00004 typedef struct
00005 {
00006     #if defined(M_ENGINE)
00007
00008     int n_transformers;
00009     int transformer_array_length;
00010     m_transformer **transformers;
00011
00012     uint16_t next_id;
00013
00014     int transition_policy;
00015
00016     #elif defined(M_INTERFACE)
00017
00018     m_transformer_pll *transformers;
00019
00020     #endif
00021 } m_pipeline;
00022
00023 #endif
```

4.144 m_profile.h File Reference

Data Structures

- struct m_profile

4.145 m_profile.h

Go to the documentation of this file.

```
00001 #ifndef M_PROFILE_H_
00002 #define M_PROFILE_H_
00003
00004 #ifdef M_INTERFACE
00005     struct m_int_glide_button_pll;
00006     struct m_int_menu_item_pll;
00007 #endif
00008
00009 #ifdef M_ENGINE
00010     struct m_pipeline_mod_ll;
00011 #endif
00012
00013 typedef struct m_profile
00014 {
00015     int active;
00016
00017     #if defined(M_ENGINE)
00018     m_pipeline *front_pipeline;
00019     m_pipeline *back_pipeline;
00020
00021     int pipelines_swapping;
00022     int pipeline_swap_progress;
00023     int pipeline_swap_samples;
00024     int pipeline_swap_type;
00025     int back_pipeline_warmed_up;
00026
00027     struct m_pipeline_mod_ll *jobs;
00028     struct m_pipeline_mod_ll *ujobs;
00029
00030     int transition_policy;
00031
00032     float *prev_block;
00033
00034     m_transformer output_amp;
00035
00036     #elif defined(M_INTERFACE)
00037
00038     char *name;
00039     uint16_t id;
00040     m_pipeline pipeline;
00041
00042     struct m_ui_page *view_page;
00043
00044     struct m_int_menu_item_pll *listings;
00045     struct m_int_glide_button_pll *gbs;
00046
00047     m_parameter volume;
00048
00049     char *fname;
00050
00051     int default_profile;
00052     int unsaved_changes;
00053
00054     #endif
00055 } m_profile;
00056
00057 #endif
```

4.146 m_status.h File Reference

Macros

- #define M_STATUS_OK 0
- #define M_STATUS_BOOTING 0b0001
- #define M_STATUS_FRESH_BOOT 0b0010

4.146.1 Macro Definition Documentation

4.146.1.1 M_STATUS_BOOTING

```
#define M_STATUS_BOOTING 0b0001
```

4.146.1.2 M_STATUS_FRESH_BOOT

```
#define M_STATUS_FRESH_BOOT 0b0010
```

4.146.1.3 M_STATUS_OK

```
#define M_STATUS_OK 0
```

4.147 m_status.h

Go to the documentation of this file.

```
00001 #ifndef M_STATUS_H_
00002 #define M_STATUS_H_
00003
00004 #define M_STATUS_OK              0
00005 #define M_STATUS_BOOTING    0b0001
00006 #define M_STATUS_FRESH_BOOT 0b0010
00007
00008 #endif
```

4.148 m_transformer.h File Reference

Data Structures

- struct m_transformer

Functions

- DECLARE_LINKED_PTR_LIST (m_transformer)

4.148.1 Function Documentation

4.148.1.1 DECLARE_LINKED_PTR_LIST()

```
DECLARE_LINKED_PTR_LIST (
            m_transformer )
```

4.149 m_transformer.h

Go to the documentation of this file.

```
00001 #ifndef M_TRANSFORMER_H_
00002 #define M_TRANSFORMER_H_
00003
00004 #ifdef M_INTERFACE
00005     struct m_ui_page;
00006 #endif
00007
00008 #ifdef M_ENGINE
00009     #include "m_eng_linkowitz_riley.h"
00010 #endif
00011
00012 typedef struct m_transformer
00013 {
00014     uint16_t type;
00015     uint16_t id;
00016
00017     m_parameter wet_mix;
00018
00019     m_setting band_mode;
00020     m_parameter band_lp_cutoff;
00021     m_parameter band_hp_cutoff;
00022     m_parameter band_center;
00023     m_parameter band_width;
00024
00025     #if defined(M_ENGINE)
00026
00027     int n_settings;
00028     int setting_array_size;
00029
00030     m_setting **settings;
00031
00032     int n_parameters;
00033     int parameter_array_size;
00034     m_parameter **parameters;
00035
00036     struct m_lr_low_pass_filter_str  input_lpf;
00037     struct m_lr_low_pass_filter_str  input_lpf_complement;
00038     struct m_lr_high_pass_filter_str input_hpf;
00039     struct m_lr_high_pass_filter_str input_hpf_band;
00040     struct m_lr_high_pass_filter_str input_hpf_complement;
00041
00042     void *data_struct;
00043     int (*compute_transformer)(void *data_struct, float *dest, float *src, int n_samples);
00044     int (*compute_transformer_nl)(void *data_struct, float **dest, float **src, int n_samples);
00045
00046     int (*reconfigure)(void *data_struct);
00047     int (*clone_struct)(void *dest, void *src);
00048     int (*free_struct)(void *data_struct);
00049
00050     size_t struct_size;
00051     int transition_policy;
00052
00053     #elif defined(M_INTERFACE)
00054
00055     int position;
00056
00057     m_parameter_pll *parameters;
00058     m_setting_pll *settings;
00059
00060     struct m_profile *profile;
00061     struct m_ui_page *view_page;
00062
00063     #endif
00064 } m_transformer;
00065
00066 DECLARE_LINKED_PTR_LIST(m_transformer);
00067
00068 #endif
```

4.150 m_transformer_enum.c File Reference

```
#include <stdint.h>
#include "m_transformer_enum.h"
```

Functions

- const char * transformer_type_to_string (uint16_t type)
- int transformer_type_valid (uint16_t type)

4.150.1 Function Documentation

4.150.1.1 transformer_type_to_string()

```
const char * transformer_type_to_string (
            uint16_t type)
```

4.150.1.2 transformer_type_valid()

```
int transformer_type_valid (
            uint16_t type)
```

4.151 m_transformer_enum.h File Reference

Macros

- #define TRANSFORMER_MODE_FULL_SPECTRUM 0
- #define TRANSFORMER_MODE_UPPER_SPECTRUM 1
- #define TRANSFORMER_MODE_LOWER_SPECTRUM 2
- #define TRANSFORMER_MODE_BAND 3
- #define TRANSFORMER_WET_MIX_PID 0xFFFF
- #define TRANSFORMER_BAND_LP_CUTOFF_PID 0xFFFE
- #define TRANSFORMER_BAND_HP_CUTOFF_PID 0xFFFD
- #define TRANSFORMER_BAND_MODE_SID 0xFFFF
- #define TRANSFORMER_3_BAND_EQ 0
- #define TRANSFORMER_AMPLIFIER 1
- #define TRANSFORMER_BAND_PASS_FILTER 2
- #define TRANSFORMER_COMPRESSOR 3
- #define TRANSFORMER_DELAY 4
- #define TRANSFORMER_DIRTY_OCTAVE 5
- #define TRANSFORMER_DISTORTION 6
- #define TRANSFORMER_ENVELOPE 7
- #define TRANSFORMER_FLANGER 8
- #define TRANSFORMER_HIGH_PASS_FILTER 9
- #define TRANSFORMER_LOW_END_COMPRESSOR 10
- #define TRANSFORMER_LOW_PASS_FILTER 11
- #define TRANSFORMER_NOISE_SUPPRESSOR 12
- #define TRANSFORMER_PERCUSSIFIER 13
- #define TRANSFORMER_WARBLER 14
- #define DISTORTION_SOFT_FOLD 0
- #define DISTORTION_ARCTAN 1
- #define DISTORTION_TANH 2
- #define DISTORTION_CLIP 3

Enumerations

- enum biquad_type {
  low_pass = 0 , high_pass = 1 , band_pass = 2 , notch = 3 ,
  peaking_band_eq = 4 , low_shelf = 5 , high_shelf = 6 }

Functions

- const char * transformer_type_to_string (uint16_t type)
- int transformer_type_valid (uint16_t type)

4.151.1 Macro Definition Documentation

4.151.1.1 DISTORTION_ARCTAN

```
#define DISTORTION_ARCTAN 1
```

4.151.1.2 DISTORTION_CLIP

```
#define DISTORTION_CLIP 3
```

4.151.1.3 DISTORTION_SOFT_FOLD

```
#define DISTORTION_SOFT_FOLD 0
```

4.151.1.4 DISTORTION_TANH

```
#define DISTORTION_TANH 2
```

4.151.1.5 TRANSFORMER_3_BAND_EQ

```
#define TRANSFORMER_3_BAND_EQ 0
```

4.151.1.6 TRANSFORMER_AMPLIFIER

```
#define TRANSFORMER_AMPLIFIER 1
```

4.151.1.7 TRANSFORMER_BAND_HP_CUTOFF_PID

```
#define TRANSFORMER_BAND_HP_CUTOFF_PID 0xFFFD
```

4.151.1.8 TRANSFORMER_BAND_LP_CUTOFF_PID

```
#define TRANSFORMER_BAND_LP_CUTOFF_PID 0xFFFE
```

4.151.1.9 TRANSFORMER_BAND_MODE_SID

```
#define TRANSFORMER_BAND_MODE_SID 0xFFFF
```

4.151.1.10 TRANSFORMER_BAND_PASS_FILTER

```
#define TRANSFORMER_BAND_PASS_FILTER 2
```

4.151.1.11 TRANSFORMER_COMPRESSOR

```
#define TRANSFORMER_COMPRESSOR 3
```

4.151.1.12 TRANSFORMER_DELAY

```
#define TRANSFORMER_DELAY 4
```

4.151.1.13 TRANSFORMER_DIRTY_OCTAVE

```
#define TRANSFORMER_DIRTY_OCTAVE 5
```

4.151.1.14 TRANSFORMER_DISTORTION

```
#define TRANSFORMER_DISTORTION 6
```

4.151.1.15 TRANSFORMER_ENVELOPE

```
#define TRANSFORMER_ENVELOPE 7
```

4.151.1.16 TRANSFORMER_FLANGER

```
#define TRANSFORMER_FLANGER 8
```

4.151.1.17 TRANSFORMER_HIGH_PASS_FILTER

```
#define TRANSFORMER_HIGH_PASS_FILTER 9
```

4.151.1.18 TRANSFORMER_LOW_END_COMPRESSOR

```
#define TRANSFORMER_LOW_END_COMPRESSOR 10
```

4.151.1.19 TRANSFORMER_LOW_PASS_FILTER

```
#define TRANSFORMER_LOW_PASS_FILTER 11
```

4.151.1.20 TRANSFORMER_MODE_BAND

```
#define TRANSFORMER_MODE_BAND 3
```

4.151.1.21 TRANSFORMER_MODE_FULL_SPECTRUM

```
#define TRANSFORMER_MODE_FULL_SPECTRUM 0
```

4.151.1.22 TRANSFORMER_MODE_LOWER_SPECTRUM

```
#define TRANSFORMER_MODE_LOWER_SPECTRUM 2
```

4.151.1.23 TRANSFORMER_MODE_UPPER_SPECTRUM

```
#define TRANSFORMER_MODE_UPPER_SPECTRUM 1
```

4.151.1.24 TRANSFORMER_NOISE_SUPPRESSOR

```
#define TRANSFORMER_NOISE_SUPPRESSOR 12
```

4.151.1.25 TRANSFORMER_PERCUSSIFIER

```
#define TRANSFORMER_PERCUSSIFIER 13
```

4.151.1.26 TRANSFORMER_WARBLER

```
#define TRANSFORMER_WARBLER 14
```

4.151.1.27 TRANSFORMER_WET_MIX_PID

```
#define TRANSFORMER_WET_MIX_PID 0xFFFF
```

4.151.2 Enumeration Type Documentation

4.151.2.1 biquad_type

```
enum biquad_type
```

Enumerator

low_pass	
high_pass	
band_pass	
notch	
peaking_band_eq	
low_shelf	
high_shelf	

4.151.3 Function Documentation

4.151.3.1 transformer_type_to_string()

```
const char * transformer_type_to_string (
             uint16_t type)
```

4.151.3.2 transformer_type_valid()

```
int transformer_type_valid (
             uint16_t type)
```

4.152 m_transformer_enum.h

Go to the documentation of this file.

```
00001 // Code generated from config/transformer/*.yaml by codegen.py
00002 #ifndef M_TRANSFORMER_ENUM_H_
00003 #define M_TRANSFORMER_ENUM_H_
00004
00005 #define TRANSFORMER_MODE_FULL_SPECTRUM  0
00006 #define TRANSFORMER_MODE_UPPER_SPECTRUM 1
00007 #define TRANSFORMER_MODE_LOWER_SPECTRUM 2
00008 #define TRANSFORMER_MODE_BAND           3
00009
00010 #define TRANSFORMER_WET_MIX_PID 0xFFFF
00011
00012 #define TRANSFORMER_BAND_LP_CUTOFF_PID 0xFFFE
00013 #define TRANSFORMER_BAND_HP_CUTOFF_PID 0xFFFD
00014
00015 #define TRANSFORMER_BAND_MODE_SID 0xFFFF
00016
00017 #define TRANSFORMER_3_BAND_EQ          0
00018 #define TRANSFORMER_AMPLIFIER          1
00019 #define TRANSFORMER_BAND_PASS_FILTER   2
00020 #define TRANSFORMER_COMPRESSOR         3
00021 #define TRANSFORMER_DELAY              4
00022 #define TRANSFORMER_DIRTY_OCTAVE       5
00023 #define TRANSFORMER_DISTORTION         6
00024 #define TRANSFORMER_ENVELOPE           7
00025 #define TRANSFORMER_FLANGER            8
00026 #define TRANSFORMER_HIGH_PASS_FILTER   9
00027 #define TRANSFORMER_LOW_END_COMPRESSOR 10
00028 #define TRANSFORMER_LOW_PASS_FILTER    11
00029 #define TRANSFORMER_NOISE_SUPPRESSOR   12
00030 #define TRANSFORMER_PERCUSSIFIER       13
00031 #define TRANSFORMER_WARBLER            14
00032
00033 typedef enum
00034 {
00035     low_pass        = 0,
00036     high_pass       = 1,
```

```
00037     band_pass       = 2,
00038     notch           = 3,
00039     peaking_band_eq = 4,
00040     low_shelf       = 5,
00041     high_shelf        = 6
00042 } biquad_type;
00043
00044 #define DISTORTION_SOFT_FOLD    0
00045 #define DISTORTION_ARCTAN       1
00046 #define DISTORTION_TANH         2
00047 #define DISTORTION_CLIP         3
00048
00049 const char *transformer_type_to_string(uint16_t type);
00050 int transformer_type_valid(uint16_t type);
00051
00052 #endif
```

Index