Krawall Reference Manual

Generated by Doxygen 1.3.7

Sun Nov 14 17:23:33 2004

Contents

1 Krawall File Index

1.1 Krawall File List

Here is a list of all files with brief descriptions:

2 Krawall File Documentation

2.1 krawall.h File Reference

Defines

- #define KRAG_INIT_MONO 0
- #define KRAG_INIT_STEREO 1
- #define KRAP_MODE_LOOP 1
- #define KRAP_MODE_SONG 2
- #define KRAP_MODE_JINGLE 4
- #define KRAP_CB_FADE 1
- #define KRAP_CB_DONE 2
- #define KRAP_CB_MARK 3
- #define KRAP_CB_SONG 4
- #define KRAP_CB_JDONE 5
- #define KRAP_CB_LOOP 6
- #define KRAM_QM_NORMAL 0
- #define KRAM_QM_MARKED 1
- #define KRAM_QM_HQ 2
- #define KRAM_QM_RAMP 16
- #define KRAM_QM_RAMP_OFF 32
- #define KRAM_SP_LEFT -64
- #define KRAM_SP_CENTER 0
- #define KRAM_SP_RIGHT 64
- #define KRAM_MV_CHANNELS32 (5 << 16)
- #define KRAM_MV_CHANNELS16 (4 << 16)
- #define KRAM_MV_CHANNELS8 (3 << 16)

Functions

- void kragInit (int stereo)
  Init function.
- void kragReset ()
  Reset function.
- void krapPlay (const Module *m, int mode, int song)
  Start music.
- void krapStop ()
  Stop music.
- void krapStopJingle ()
  Stop a playing jingle.
- void krapCallback (void(*func)(int, int))
  Install callback.
- void krapPause (int sfx)
  Pause music.
- void krapUnpause ()
  Unpause music.
- int krapIsPaused ()
  Get Pause status.
- void krapSetMusicVol (unsigned int vol, int fade)
  Set music volume.
- unsigned int krapGetMusicVol ()
  Get music volume.
- ihandle krapInstPlay (const Instrument *ins, int note, ihandle old)
  Play an instrument as SFX.
- int krapInstRelease (ihandle i)
  Release a playing instrument.
- int krapInstStop (ihandle i)
  Stop a playing instrument.
- void krapInstProcess ()
  Process instrument-sfx envelopes.
- int krapInstHandleValid (ihandle i)
  Check if an instrument-handle is still valid.

- void krapSetChannelVol (unsigned int channel, unsigned int vol)
  Set a channel's volume.
- unsigned int krapGetChannelVol (unsigned int channel)
  Get a channel's volume.
- int kramWorker () LONG_CALL
  Worker procedure.
- int kramGetActiveChannels ()
  Get number of currently active channels.
- void kramQualityMode (int)
  Set quality mode.
- chandle kramPlay (const Sample *s, int sfx, chandle c) LONG_CALL
  Play a sample.
- chandle kramPlayExt (const Sample *s, int sfx, chandle c, unsigned int freq, unsigned int vol, int pan) LONG_CALL
  Play a sample Ext.
- int kramStop (chandle c) LONG_CALL
  Stop a channel.
- int kramSetFreq (chandle c, unsigned int freq) LONG_CALL
  Set frequency.
- int kramSetVol (chandle c, unsigned int vol) LONG_CALL
  Set volume.
- int kramSetPan (chandle c, int pan) LONG_CALL
  Set panning.
- int kramSetPos (chandle c, unsigned int pos) LONG_CALL
  Set Position.
- void kramSetSFXVol (unsigned int vol)
  Set SFX volume.
- unsigned int kramGetSFXVol ()
  Get SFX volume.
- void kramSetMasterVol (unsigned int vol)
  Set Master Clip volume.
- void kramStopSFXChannels ()
  Stops all active SFX channels.
- int kramHandleValid (chandle c)

Check whether a handle is still valid.

- unsigned int kramGetFreq (chandle c)
  Get frequency.
- unsigned int kramGetVol (chandle c)
  Get volume.
- int kramGetPan (chandle c)
  Get panning.
- unsigned int kramGetPos (chandle c)
  Get position.
- void kradInterrupt ()
  Directsound Interrupt.
- void kradActivate ()
  Activate Krawall.
- void kradDeactivate ()
  Deactivate Krawall.

Variables

- const Sample *const samples [ ]
- const Instrument *const instruments [ ]

2.1.1 Define Documentation

2.1.1.1 #define KRAG_INIT_MONO 0

2.1.1.2 #define KRAG_INIT_STEREO 1

2.1.1.3 #define KRAM_MV_CHANNELS16 (4 << 16)

2.1.1.4 #define KRAM_MV_CHANNELS32 (5 << 16)

2.1.1.5 #define KRAM_MV_CHANNELS8 (3 << 16)

2.1.1.6 #define KRAM_QM_HQ 2

2.1.1.7 #define KRAM_QM_MARKED 1

2.1.1.8 #define KRAM_QM_NORMAL 0

2.1.1.9 #define KRAM_QM_RAMP 16

2.1.1.10 #define KRAM_QM_RAMP_OFF 32

2.1.1.11 #define KRAM_SP_CENTER 0

2.1.1.12 #define KRAM_SP_LEFT -64

2.1.1.13 #define KRAM_SP_RIGHT 64

2.1.1.14 #define KRAP_CB_DONE 2

2.1.1.15 #define KRAP_CB_FADE 1

2.1.1.16 #define KRAP_CB_JDONE 5

2.1.1.17 #define KRAP_CB_LOOP 6

2.1.1.18 #define KRAP_CB_MARK 3

2.1.1.19 #define KRAP_CB_SONG 4

2.1.1.20 #define KRAP_MODE_JINGLE 4

2.1.1.21 #define KRAP_MODE_LOOP 1

2.1.1.22 #define KRAP_MODE_SONG 2

2.1.2 Function Documentation

2.1.2.1 void kradActivate ()

You only need to call this if you have called kradDeactivate().

2.1.2.2 void kradDeactivate ()

You might want to deactivate Krawall in order to write savegames and stuff like that. Calling this will stop all DMA-operations (and thus sound-output) until resumed by kradActivate().

2.1.2.3 void kradInterrupt ()

This function resets the DMA and must be tied to the Timer1-IRQ.

2.1.2.4 void kragInit (int *stereo*)

Call this function once at startup.

Parameters:

stereo Whether Krawall should operate stereo (KRAG_INIT_STEREO) or not (KRAG_INIT_-MONO)

2.1.2.5 void kragReset ()

This is only needed if you want to call kragInit() again, most likely because you want to switch from mono to stereo or vice versa. Calling this while there is sound being output will result in an audible (but harmless) hickup.

2.1.2.6 int kramGetActiveChannels ()

Returns number of currently active channels.

Returns:

Number of currently active channels

2.1.2.7 unsigned int kramGetFreq (chandle *c*)

Get frequency of a channel. THIS FUNCTION DOES NOT CHECK FOR HANDLE-VALIDITY!

Parameters:

c handle

Returns:

current frequency

2.1.2.8 int kramGetPan (chandle *c*)

Get panning of a channel. THIS FUNCTION DOES NOT CHECK FOR HANDLE-VALIDITY!

Parameters:

c handle

Returns:

current panning

2.1.2.9 unsigned int kramGetPos (chandle *c*)

Get position of a channel. THIS FUNCTION DOES NOT CHECK FOR HANDLE-VALIDITY!

Parameters:

c handle

Returns:

current position

2.1.2.10 unsigned int kramGetSFXVol ()

Returns volume as set with kramSetSFXVol().

See also:
kramSetSFXVol

Returns:
volume

2.1.2.11 unsigned int kramGetVol (chandle *c*)

Get volume of a channel. THIS FUNCTION DOES NOT CHECK FOR HANDLE-VALIDITY!

Parameters:
c handle

Returns:
current volume

2.1.2.12 int kramHandleValid (chandle *c*)

Checks if chandle is still a valid handle. A handle will get invalidated if for example a one-shot sample ends.

Parameters:
c handle to check

Returns:
true if valid, false if invalid

2.1.2.13 chandle kramPlay (const Sample * *s*, int *sfx*, chandle *c*)

Plays a sample with it's C2 (neutral) frequency.

Parameters:
s Pointer to sample

sfx Whether sample to play is an SFX

c Old handle, will be recycled if given

See also:
kramPlayExt()

Returns:
Channel handle

2.1.2.14 chandle kramPlayExt (const Sample * *s*, int *sfx*, chandle *c*, unsigned int *freq*, unsigned int *vol*, int *pan*)

Just like kramPlay, but all of the attribs can be specified.

Parameters:

s Pointer to sample

sfx Whether sample to play is an SFX

c Old handle, will be recycled if given

freq Frequency in hertz to play sample at

vol Volume to play sample with (0..64)

pan Panning to play sample with (-64..64)

See also:

kramPlay()

Returns:

Channel handle

2.1.2.15 void kramQualityMode (int)

This sets the quality mode of the mixing routines. KRAM_QM_NORMAL is the default, KRAM_QM_-MARKED only plays the marked samples (see docs) in HQ and KRAM_QM_HQ plays everything in HQ. The flag KRAM_QM_RAMP_OFF (must be OR'd) DISABLES stop-ramping. Until version 20040707 you had to enable it explicitly, now it must be disabled explicitly. Hence the flag KRAM_QM_RAMP has lost it's meaning. Especially looped samples that get stopped abruptly might cause pops. Stop-ramping removes these pops at the cost of a little more CPU.

2.1.2.16 int kramSetFreq (chandle *c*, unsigned int *freq*)

Sets frequency of an active channel. Note that if the channel has already stopped this call will not do anything and return false.

Parameters:

c Channel handle

freq Frequency in hertz

Returns:

true if successful

2.1.2.17 void kramSetMasterVol (unsigned int *vol*)

Sets the clipping curve's steepness. 128 is the default value, setting a neutral clipping curve. Values below 128 (down to 16) can be used to reduce distortion (volume) if the output is too high. Values above 128 will give you additional gain but also reduce the quality because information is lost, don't do this. Additionally you can OR the volume with one of the parameters KRAM_MV_CHANNELS32, KRAM_MV_-CHANNELS16 or KRAM_MV_CHANNELS8. KRAM_MV_CHANNELS32 is the default – specifying one of the other values will give you additional gain. However as the parameter says you should not use more than the amount of channels then - otherwise you might get unpredictable clicks/distortion.

Parameters:

vol 128 is default, everything below/above changes the clipping curve. OR with KRAM_MV_-CHANNELS16 or KRAM_MV_CHANNELS8 if appropriate

2.1.2.18 int kramSetPan (chandle *c*, int *pan*)

Sets the panning-position of an active channel. Note that if the channel has already stopped this call will not do anything and return false.

Parameters:

c Channel handle

pan Panning (-64..0..64), KRAM_SP_LEFT, KRAM_SP_RIGHT, KRAM_SP_CENTER

Returns:

true if successful

2.1.2.19 int kramSetPos (chandle *c*, unsigned int *pos*)

Sets the sample-position of an active channel. Note that if the channel has already stopped this call will not do anything and return false.

Parameters:

c Channel handle

pos Sample offset to set

Returns:

true if successful

2.1.2.20 void kramSetSFXVol (unsigned int *vol*)

Sets the volume of all active and future sfx.

Parameters:

vol Volume (0..128)

2.1.2.21 int kramSetVol (chandle *c*, unsigned int *vol*)

Sets volume of an active channel. Note that if the channel has already stopped this call will not do anything and return false.

Parameters:

c Channel handle

vol Volume (0..64)

Returns:

true if successful

2.1.2.22 int kramStop (chandle *c*)

Stops playback of a channel. Note that if the channel has already stopped this call will not do anything and return false.

Parameters:

c Channel handle

Returns:

true if successful

2.1.2.23 void kramStopSFXChannels ()

Stops all currently active SFX channels. Paused SFX channels are not stopped.

2.1.2.24 int kramWorker ()

This is where the actual work is done, you ∗MUST∗ call this once per frame after kraInit() to get sound

See also:
kragInit()

Returns:
True if actual work has been done

2.1.2.25 void krapCallback (void(∗ *func*)(int, int))

Installs a callback. The callback should return as quickly as possible. When the callback gets called the first numeric parameter describes the event, the second numeric parameter (if any) is the parameter to the event. The events are as following:

- KRAP_CB_FADE Destination volume has been reached
- KRAP_CB_DONE Module is done (also when KRAP_MODE_LOOP)
- KRAP_CB_MARK Mark-Effect Zxx (xx in param 2)
- KRAP_CB_SONG Song-boundary hit (+++-Marker)
- KRAP_CB_JDONE Jingle is done

See also:
krapSetMusicVol

2.1.2.26 unsigned int krapGetChannelVol (unsigned int *channel*)

This will get a channel's volume as either set by krapSetChannelVol() or S3M-effects Mxx and Nxx.

Parameters:
channel mod-channel (0..#channels of current mod)

Returns:
volume

See also:
krapSetChannelVol

2.1.2.27 unsigned int krapGetMusicVol ()

Returns volume as set by krapSetMusicVol()

See also:
krapSetMusicVol

Returns:
volume

2.1.2.28 int krapInstHandleValid (ihandle *i*)

This is similiar to kramHandleValid() but works for an instrument-handle. It will return false if an instrument has already stopped playing (one-shot sample).

Parameters:
i handle to check

Returns:
true if valid, false if invalid

See also:
kramHandleValid

2.1.2.29 ihandle krapInstPlay (const Instrument * *ins*, int *note*, ihandle *old*)

Plays an instrument as an SFX. If you use this, be sure to call krapInstProcess() periodically, this is where the envelopes get processed.

Parameters:
ins Pointer to instrument (instruments[])

note 0 (C-0) .. 95 (B-7)

old Old handle, will be recycled if given

See also:
kramPlay
krapInstRelease
krapInstStop
krapInstProcess

Returns:
Instrument handle, zero if no channel/instrument could be allocated

2.1.2.30 void krapInstProcess ()

If you use instruments for sfx you should call this periodically. Once a frame is quite good idea.

2.1.2.31 int krapInstRelease (ihandle *i*)

Releases a playing instrument if either still playing or in sustain-mode.

Parameters:
i Handle as returned by krapInstPlay()

Returns:
true if successful

See also:
krapInstPlay

2.1.2.32 int krapInstStop (ihandle *i*)

Will immediately stop a playing instrument

Parameters:
i Handle as returned by krapInstPlay()

Returns:
true if successful

See also:
krapInstPlay

2.1.2.33 int krapIsPaused ()

Returns whether playback is currently paused or not

Returns:
True if paused

See also:
krapPause
krapUnpause

2.1.2.34 void krapPause (int *sfx*)

Pauses all currently active channels. You still can play SFX's. The paused channels will be frozen until krapUnpause() gets called.

Parameters:
sfx If true pause sfx as well; if false pause music only

See also:
krapUnpause

2.1.2.35 void krapPlay (const Module * *m*, int *mode*, int *song*)

Parameters:
m Pointer to module

mode is one or more of:

- KRAP_MODE_LOOP Loop module
- KRAP_MODE_SONG Enable song-mode
- KRAP_MODE_JINGLE Play module as jingle

song Song of module to play

See also:
krapStop

2.1.2.36 void krapSetChannelVol (unsigned int *channel*, unsigned int *vol*)

This will set a channel's volume. S3M-effects Mxx and Nxx will override this value. krapPlay will reset all channel's volume to 64.

Parameters:

channel mod-channel (0..#channels of current mod)

vol volume (0..64)

See also:

krapGetChannelVol

2.1.2.37 void krapSetMusicVol (unsigned int *vol*, int *fade*)

You can either set the music volume immediately or fade slowly to the specified volume. The fadespeed depends on the speed of the currently active module. If module is paused then volume is always set immediately. If a callback is installed it will get triggered when fading is done. The volume given will directly scale the global volume set in the S3M.

Parameters:

vol Music volume (0..128)

fade If true fade, if false set immediately

See also:

kramSetSFXVol
krapCallback

2.1.2.38 void krapStop ()

Immediately stops playback of music.

See also:

krapPlay

2.1.2.39 void krapStopJingle ()

This will immediately stop a playing jingle and resume playback of the old song. Note that if no jingle is playing this function will do nothing. The callback will immediately get called with KRAP_CB_JDONE if set.

See also:

krapPlay
krapCallback

2.1.2.40 void krapUnpause ()

Reactivates all channels that have been paused with krapPause()

See also:

krapPause

2.1.3 Variable Documentation

2.1.3.1 const Instrument* const instruments[ ]

2.1.3.2 const Sample* const samples[ ]

Index