The Kwave Handbook

Thomas Eschenbacher

The Kwave Handbook

Contents

1 Introduction **14**
1.1 Kwave Resources . . . 15
1.2 Kwave Revision History . . . 15

2 Basics about digital audio **16**
2.1 The analogue world . . . 16
2.2 Digitalization . . . 17
2.3 Sample Encoding . . . 18
2.4 Sample Formats . . . 18

3 Using Kwave **20**
3.1 User Interface Mode . . . 20
3.2 Command Line . . . 22
3.2.1 List of Files or Commands . . . 22
3.2.2 GUI Type . . . 22
3.2.3 Qt Toolkit options . . . 22
3.2.4 Startup Options . . . 23
3.3 Opening and Saving files . . . 23
3.3.1 Supported File formats . . . 23
3.3.2 Converting to and from .wav . . . 24
3.3.3 Format of ASCII files . . . 24
3.4 Creating a New File . . . 25
3.5 Recording . . . 26
3.6 Playback . . . 27
3.7 File Properties . . . 28
3.8 Zooming and navigating . . . 29
3.8.1 Zooming in and out . . . 29
3.8.2 Scrolling left and right . . . 30
3.8.3 Using the overview . . . 30
3.8.4 Vertical zoom . . . 30
3.9 How to select . . . 31
3.9.1 Selecting channels . . . 31
3.9.2 Selecting samples . . . 31
3.10 Clipboard . . . 32
3.11 Drag and Drop . . . 32

4 Automization and Scripting with Kwave 33
4.1 General Syntax 33
4.2 Using the Command Line 33
4.3 Kwave Script Files 34
4.3.1 General Structure 34
4.3.2 Comments and Empty Lines 34
4.3.3 Termination 35
4.3.4 Labels 35
4.4 Command Reference 35
4.5 a 35
4.5.1 about_kde 35
4.5.1.1 Syntax: **about_kde**() 35
4.5.2 add_track 35
4.5.2.1 Syntax: **add_track**() 35
4.5.2.2 See also 35
4.6 c 36
4.6.1 clipboard_flush 36
4.6.1.1 Syntax: **clipboard_flush**() 36
4.6.2 close 36
4.6.2.1 Syntax: **close**() 36
4.6.2.2 See also 36
4.6.3 continue 36
4.6.3.1 Syntax: **continue**() 36
4.6.3.2 See also 36
4.6.4 copy 36
4.6.4.1 Syntax: **copy**() 36
4.6.4.2 See also 36
4.6.5 crop 36
4.6.5.1 Syntax: **crop**() 36
4.6.6 cut 37
4.6.6.1 Syntax: **cut**() 37
4.7 d 37
4.7.1 delayed 37
4.7.1.1 Syntax: **delayed**(*milliseconds*, *command*) 37
4.7.1.2 Parameters 37
4.7.1.3 See also 37
4.7.2 delete 37
4.7.2.1 Syntax: **delete**() 37
4.7.3 delete_track 37
4.7.3.1 Syntax: **delete_track**(*index*) 37
4.7.3.2 Parameters 37
4.7.4 dump_metadata 38

4.7.4.1 Syntax: **dump_metadata**() . . . 38
4.8 e . . . 38
4.8.1 expandtolabel . . . 38
4.8.1.1 Syntax: **expandtolabel**() . . . 38
4.9 f . . . 38
4.9.1 fileinfo . . . 38
4.9.1.1 Syntax: **fileinfo**(*index*) . . . 38
4.9.1.2 Parameters . . . 38
4.9.2 forward . . . 38
4.9.2.1 Syntax: **forward**() . . . 38
4.9.2.2 See also . . . 38
4.10 g . . . 39
4.10.1 goto . . . 39
4.10.1.1 Syntax: **goto**(*pos*) . . . 39
4.10.1.2 Parameters . . . 39
4.11 i . . . 39
4.11.1 insert_at . . . 39
4.11.1.1 Syntax: **insert_at**(*pos*) . . . 39
4.11.1.2 Parameters . . . 39
4.11.1.3 See also . . . 39
4.11.2 insert_track . . . 39
4.11.2.1 Syntax: **insert_track**(*index*) . . . 39
4.11.2.2 Parameters . . . 39
4.11.2.3 See also . . . 40
4.12 l . . . 40
4.12.1 label:add . . . 40
4.12.1.1 Syntax: **label:add**(*pos*[,*text*]) . . . 40
4.12.1.2 Parameters . . . 40
4.12.2 label:delete . . . 40
4.12.2.1 Syntax: **label:delete**(*index*) . . . 40
4.12.2.2 Parameters . . . 40
4.12.3 label:edit . . . 41
4.12.3.1 Syntax: **label:edit**(*index*) . . . 41
4.12.3.2 Parameters . . . 41
4.12.4 loadbatch . . . 41
4.12.4.1 Syntax: **loadbatch**(*filename*) . . . 41
4.12.4.2 Parameters . . . 41
4.12.5 loop . . . 42
4.12.5.1 Syntax: **loop**() . . . 42
4.12.5.2 See also . . . 42
4.13 m . . . 42
4.13.1 menu . . . 42
4.13.1.1 Syntax: **menu** (*command*, *path*, [*hotkey*], [*id*]) . . . 42

4.13.1.2 Parameters . . . 42
4.13.1.3 Sub Commands . . . 43
4.13.2 msgbox . . . 45
4.13.2.1 Syntax: **msgbox**(*text*) . . . 45
4.13.2.2 Parameters . . . 45
4.14 n . . . 45
4.14.1 newsignal . . . 45
4.14.1.1 Syntax: **newsignal**(*samples*, *rate*, *bits*, *tracks*) . . . 45
4.14.1.2 Parameters . . . 45
4.14.2 next . . . 45
4.14.2.1 Syntax: **next**() . . . 45
4.14.2.2 See also . . . 45
4.15 o . . . 46
4.15.1 open . . . 46
4.15.1.1 Syntax: **open**([*filename*]) . . . 46
4.15.1.2 Parameters . . . 46
4.15.1.3 See also . . . 46
4.15.2 openrecent . . . 46
4.15.2.1 Syntax: **openrecent**(*filename*) . . . 46
4.15.2.2 Parameters . . . 46
4.15.2.3 See also . . . 46
4.16 p . . . 46
4.16.1 paste . . . 46
4.16.1.1 Syntax: **paste**() . . . 46
4.16.1.2 See also . . . 47
4.16.2 pause . . . 47
4.16.2.1 Syntax: **continue**() . . . 47
4.16.2.2 See also . . . 47
4.16.3 playback_start . . . 47
4.16.3.1 Syntax: **playback_start**() . . . 47
4.16.4 plugin . . . 47
4.16.4.1 Syntax: **plugin**(*name*, [*parameter* ...]) . . . 47
4.16.4.2 Parameters . . . 47
4.16.4.3 See also . . . 47
4.16.5 plugin:execute . . . 48
4.16.5.1 Syntax: **plugin:execute**(*name*, [*parameter* ...]) . . . 48
4.16.5.2 Parameters . . . 48
4.16.6 plugin:setup . . . 48
4.16.6.1 Syntax: **plugin:setup**(*name*, [*parameter* ...]) . . . 48
4.16.6.2 Parameters . . . 48
4.16.7 prev . . . 48
4.16.7.1 Syntax: **prev**() . . . 48

4.16.7.2 See also . . . 48
4.17 q . . . 49
4.17.1 quit . . . 49
4.17.1.1 Syntax: **quit**() . . . 49
4.17.1.2 See also . . . 49
4.18 r . . . 49
4.18.1 redo . . . 49
4.18.1.1 Syntax: **redo**() . . . 49
4.18.1.2 See also . . . 49
4.18.2 redo_all . . . 49
4.18.2.1 Syntax: **redo_all**() . . . 49
4.18.2.2 See also . . . 49
4.18.3 reenable_dna . . . 49
4.18.3.1 Syntax: **reenable_dna**() . . . 49
4.18.4 reset_toolbars . . . 49
4.18.4.1 Syntax: **reset_toolbars**() . . . 49
4.18.5 revert . . . 50
4.18.5.1 Syntax: **revert**() . . . 50
4.18.6 rewind . . . 50
4.18.6.1 Syntax: **rewind**() . . . 50
4.18.6.2 See also . . . 50
4.19 s . . . 50
4.19.1 save . . . 50
4.19.1.1 Syntax: **save**() . . . 50
4.19.1.2 See also . . . 50
4.19.2 saveas . . . 50
4.19.2.1 Syntax: **saveas**([*filename*]) . . . 50
4.19.2.2 Parameters . . . 50
4.19.3 saveselect . . . 51
4.19.3.1 Syntax: **saveselect**() . . . 51
4.19.3.2 See also . . . 51
4.19.4 select_gui_type . . . 51
4.19.4.1 Syntax: **select_gui_type**(*mode*) . . . 51
4.19.4.2 Parameters . . . 51
4.19.5 select_track:all . . . 51
4.19.5.1 Syntax: **select_track:all**() . . . 51
4.19.5.2 See also . . . 51
4.19.6 select_track:invert . . . 51
4.19.6.1 Syntax: **select_track:all**() . . . 51
4.19.6.2 See also . . . 51
4.19.7 select_track:none . . . 52
4.19.7.1 Syntax: **select_track:none**() . . . 52

4.19.7.2 See also . . . 52
4.19.8 select_track:off . . . 52
4.19.8.1 Syntax: **select_track:off**(*index*) . . . 52
4.19.8.2 Parameters . . . 52
4.19.9 select_track:on . . . 52
4.19.9.1 Syntax: **select_track:on**(*index*) . . . 52
4.19.9.2 Parameters . . . 52
4.19.10 select_track:toggle . . . 52
4.19.10.1 Syntax: **select_track:toggle**(*index*) . . . 52
4.19.10.2 Parameters . . . 52
4.19.11 selectall . . . 53
4.19.11.1 Syntax: **selectall**() . . . 53
4.19.12 selectnext . . . 53
4.19.12.1 Syntax: **selectnext**() . . . 53
4.19.12.2 See also . . . 53
4.19.13 selectnextlabels . . . 53
4.19.13.1 Syntax: **selectnextlabels**() . . . 53
4.19.13.2 See also . . . 53
4.19.14 selectnone . . . 53
4.19.14.1 Syntax: **selectnone**() . . . 53
4.19.15 selectprev . . . 53
4.19.15.1 Syntax: **selectprev**() . . . 53
4.19.15.2 See also . . . 54
4.19.16 selectprevlabels . . . 54
4.19.16.1 Syntax: **selectprevlabels**() . . . 54
4.19.16.2 See also . . . 54
4.19.17 selecttoleft . . . 54
4.19.17.1 Syntax: **selecttoleft**() . . . 54
4.19.17.2 See also . . . 54
4.19.18 selecttoright . . . 54
4.19.18.1 Syntax: **selecttoright**() . . . 54
4.19.18.2 See also . . . 54
4.19.19 selectvisible . . . 54
4.19.19.1 Syntax: **selectvisible**() . . . 54
4.19.20 start . . . 55
4.19.20.1 Syntax: **start**() . . . 55
4.19.20.2 See also . . . 55
4.19.21 stop . . . 55
4.19.21.1 Syntax: **stop**() . . . 55
4.19.21.2 See also . . . 55
4.19.22 sync . . . 55
4.19.22.1 Syntax: **sync**() . . . 55

4.19.22.2 See also . . . 55
4.20 u . . . 55
4.20.1 undo . . . 55
4.20.1.1 Syntax: **undo**() . . . 55
4.20.1.2 See also . . . 55
4.20.2 undo_all . . . 55
4.20.2.1 Syntax: **undo_all**() . . . 55
4.20.2.2 See also . . . 56
4.21 v . . . 56
4.21.1 view:scroll_end . . . 56
4.21.1.1 Syntax: **view:scroll_end**() . . . 56
4.21.1.2 See also . . . 56
4.21.2 view:scroll_left . . . 56
4.21.2.1 Syntax: **view:scroll_left**() . . . 56
4.21.2.2 See also . . . 56
4.21.3 view:scroll_next . . . 56
4.21.3.1 Syntax: **view:scroll_next**() . . . 56
4.21.3.2 See also . . . 56
4.21.4 view:scroll_next_label . . . 56
4.21.4.1 Syntax: **view:scroll_next_label**() . . . 56
4.21.4.2 See also . . . 56
4.21.5 view:scroll_prev . . . 57
4.21.5.1 Syntax: **view:scroll_prev**() . . . 57
4.21.5.2 See also . . . 57
4.21.6 view:scroll_prev_label . . . 57
4.21.6.1 Syntax: **view:scroll_prev_label**() . . . 57
4.21.6.2 See also . . . 57
4.21.7 view:scroll_right . . . 57
4.21.7.1 Syntax: **view:scroll_right**() . . . 57
4.21.7.2 See also . . . 57
4.21.8 view:scroll_start . . . 57
4.21.8.1 Syntax: **view:scroll_start**() . . . 57
4.21.8.2 See also . . . 57
4.21.9 view:zoom_all . . . 57
4.21.9.1 Syntax: **view:zoom_all**() . . . 57
4.21.10 view:zoom_in . . . 58
4.21.10.1 Syntax: **view:zoom_in**([*position*]) . . . 58
4.21.10.2 Parameters . . . 58
4.21.10.3 See also . . . 58
4.21.11 view:zoom_normal . . . 58
4.21.11.1 Syntax: **view:zoom_normal**() . . . 58
4.21.12 view:zoom_out . . . 58

4.21.12.1 Syntax: **view:zoom_out([*position*])** . . . 58
4.21.12.2 Parameters . . . 58
4.21.12.3 See also . . . 58
4.21.13 view:zoom_selection . . . 59
4.21.13.1 Syntax: **view:zoom_selection**() . . . 59
4.22 w . . . 59
4.22.1 window:activate . . . 59
4.22.1.1 Syntax: **window:activate**(*title*]) . . . 59
4.22.1.2 Parameters . . . 59
4.22.2 window:cascade . . . 59
4.22.2.1 Syntax: **window:cascade**() . . . 59
4.22.3 window:click . . . 59
4.22.3.1 Syntax: **window:click**(*class*, *x*, *y*) . . . 59
4.22.3.2 Parameters . . . 59
4.22.4 window:close . . . 60
4.22.4.1 Syntax: **window:close**(*class*) . . . 60
4.22.4.2 Parameters . . . 60
4.22.5 window:minimize . . . 60
4.22.5.1 Syntax: **window:minimize** . . . 60
4.22.6 window:mousemove . . . 60
4.22.6.1 Syntax: **window:resize**(*class*, *x*, *y*) . . . 60
4.22.6.2 Parameters . . . 60
4.22.7 window:next_sub . . . 60
4.22.7.1 Syntax: **window:next_sub**() . . . 60
4.22.8 window:prev_sub . . . 61
4.22.8.1 Syntax: **window:prev_sub**() . . . 61
4.22.9 window:resize . . . 61
4.22.9.1 Syntax: **window:resize**(*class*, *width*, *height*) . . . 61
4.22.9.2 Parameters . . . 61
4.22.10 window:screenshot . . . 61
4.22.10.1 Syntax: **window:screenshot**(*class*, *filename*) . . . 61
4.22.10.2 Parameters . . . 61
4.22.11 window:sendkey . . . 61
4.22.11.1 Syntax: **window:sendkey**(*class*, *key code*) . . . 61
4.22.11.2 Parameters . . . 62
4.22.11.3 See also . . . 62
4.22.12 window:tile . . . 62
4.22.12.1 Syntax: **window:tile**() . . . 62
4.22.13 window:tile_vertical . . . 62
4.22.13.1 Syntax: **window:tile_vertical**() . . . 62

5 Plugins 63
5.1 Plugin Reference . . . 63
5.2 about (About Kwave) . . . 63
5.3 amplifyfree (Amplify Free) . . . 64
5.4 band_pass (Band Pass Filter) . . . 65
5.5 codec_ascii (ASCII Codec) . . . 66
5.6 codec_audiofile (Audiofile Codec) . . . 66
5.7 codec_flac (FLAC Codec) . . . 68
5.8 codec_mp3 (MP3 Codec) . . . 68
5.9 codec_ogg (Ogg Codec) . . . 69
5.10 codec_wav (WAV Codec) . . . 70
5.11 debug (Debug Functions) . . . 70
5.12 export_k3b (Export to K3b Project) . . . 70
5.13 fileinfo (File Info) . . . 72
5.14 goto (Goto Position) . . . 73
5.15 insert_at (Insert At) . . . 74
5.16 lowpass (Low Pass Filter) . . . 75
5.17 newsignal (New Signal) . . . 76
5.18 noise (Noise Generator) . . . 77
5.19 normalize (Normalizer) . . . 77
5.20 notch_filter (Notch Filter) . . . 78
5.21 pitch_shift (Pitch Shift) . . . 79
5.22 playback (Playback) . . . 80
5.23 record (Record) . . . 81
5.24 reverse (Reverse) . . . 82
5.25 samplerate (Sample Rate Conversion) . . . 82
5.26 saveblocks (Save Blocks) . . . 83
5.27 selectrange (Select Range) . . . 85
5.28 sonagram (Sonagram) . . . 86
5.29 stringenter (Enter Command) . . . 88
5.30 volume (Volume) . . . 88
5.31 zero (Zero Generator) . . . 89

6 Questions and Answers 90

7 Credits and License 92
7.1 Main Authors . . . 92
7.2 Major Contributors . . . 92
7.3 Minor contributors, copyright holders and others . . . 93
7.4 Thanks To . . . 94

A File Info 95

List of Tables

4.1 URL Encoding Translation Table 34

A.1 List of File Info Identifiers 98

Abstract

Kwave is a simple sound editor built on KDE Frameworks 5.

Chapter 1

Introduction

This is "Kwave", a simple sound editor built on KDE Frameworks 5. Its features include:

- a user interface that can be switched to SDI, MDI or Tab mode
- simple cut, copy and paste functions
- multi-level undo/redo
- labeling of signals
- Recording functionality, including pre-recording
- Playback via Qt, PulseAudio, ALSA and OSS
- Recording via PulseAudio, ALSA and OSS
- MP3 import/export
- Ogg/Vorbis and Opus import/export
- FLAC import/export
- some analysis functions such as Sonagram
- internally uses 24 bit fixed precision for sample data
- free selectable sample rates
- support for editing of multi channel files
- playback of multi channel audio files (audio output will be mixed down to mono or stereo if needed)
- extendible through an easy-to-use plugin interface
- import/export of other audio formats through audiofile

If you are interested what has been done and what has still to be done, then look at the files `CHANGES` and `TODO` included in the source package. Help and constructive critics are always welcome.

1.1 Kwave Resources

So if you want to get in contact with the developers, need some further help on using Kwave, submit patches, bug reports or other stuff, the following resources might be of interest for you:

- Project Homepage

  For information about new up-to-date releases or some other information about this project, take a look at the Kwave homepage.

- GIT Repository

  There also is a new GIT repository hosted on KDE servers, and a mirror repository hosted by SourceForge where you can get the sources of the latest development version. For instructions on how to get access to the repository, read in the chapter about building from GIT in the developer documentation. There also is a GIT web interface on KDE and on SourceForge that you can use to browse through the sources.

1.2 Kwave Revision History

This project has been started by Martin Wilz in summer 1998 and has been developed and improved by him and some other people. In November 1999 Thomas Eschenbacher has started to fix some little bugs here and there and stepped into the source code of the program deeper and deeper. Up to today he has extended, rewritten or revised nearly every component of the program and spent much time on improving it.

Since Kwave v0.8.0 the changelog is no longer included in this manual. So if you are interested in a complete list of changes, you can find the full history here: https://invent.kde.org/multimedia/kwave/-/blob/master/CHANGES or browse through the sources on your own through the GIT web interface.

Kwave version v0.9.0 is the first version hosted on KDE (kdereview) and SourceForge servers, followed by v0.9.1, the first version for KDE Frameworks 5.

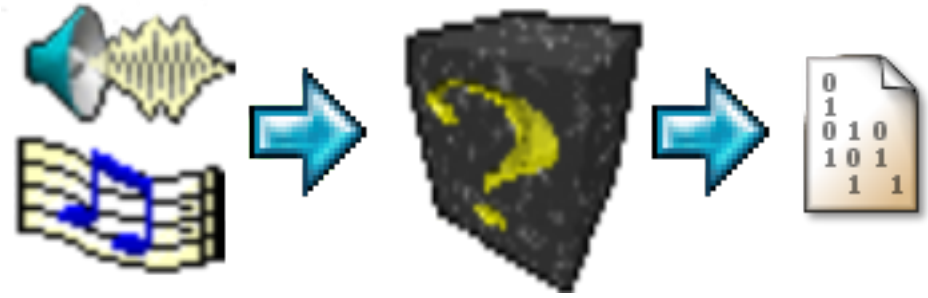
Chapter 2

Basics about digital audio

This chapter should give a short introduction about the basics of digital audio processing, without going too much into details.

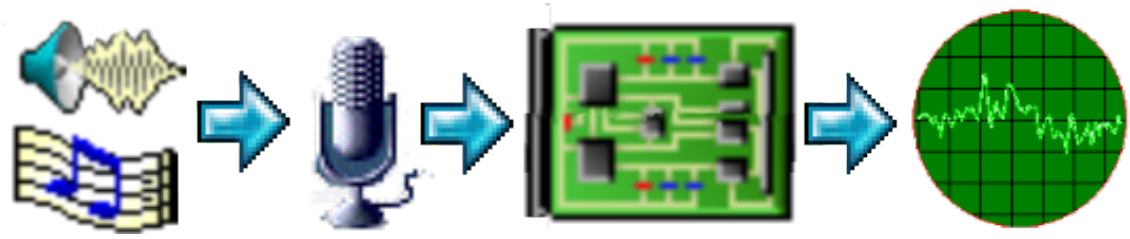
2.1 The analogue world

First of all, one must know that the world is *analogue* - but computers work *digitally*. So there are several ways to convert analogue audio to digital audio and back again. As the way from digital to analogue normally is the reversion of the way from analogue to digital, we only describe the way from analogue to digital.



Conversion from sound to bits

Before continuing, analogue audio has to be transformed into electronic signals in order to find its way into a computer. One common way to do this is by using a microphone and an amplifier. This combination gets sound (changes of air pressure) at its input and a voltage at its output. Higher amplitude of the pressure changes will be represented by higher voltages at the amplifier's output. This output is also called a *'signal'*. Instead of a microphone you can of course also imagine other sources of audio. And the "amplifier" can be the one that is integrated into your sound card, where you normally cannot see it.



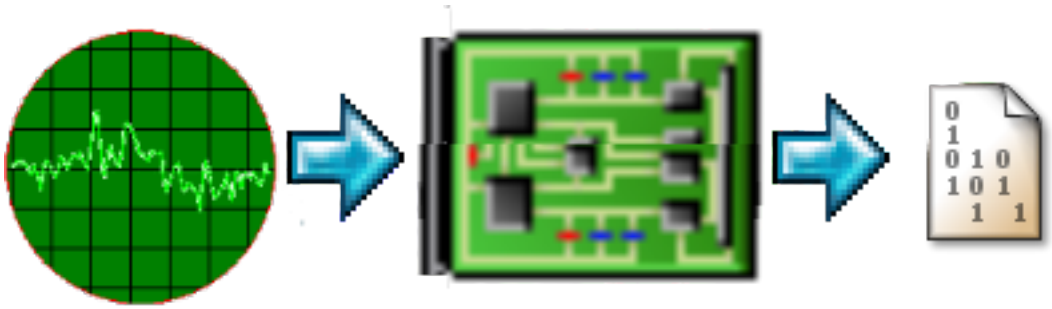
Conversion to electronic signal

At this stage, the electrical signal has three limitations that one should keep in mind:

1. The *amplitude* (volume) is limited to some maximum level. This is a consequence of the electronic (amplifiers) that are only able to handle voltages within some specific range. That's no problem as long as sounds are not too loud. In that case the signal would be *clipped*, which means that the electrical signal will run against its margins and the result will be disturbed.
2. The *frequency range* is also limited. Due to the mechanical constrains of microphones and the limited frequency range of amplifiers, a signal's frequency range is limited. There are no hard borders besides which the sound abruptly disappears, but below some low and above some higher frequency the amplitude of the signal starts to decrease more and more. The existence of a maximum frequency can be easily understood as a limited speed of the electrical signal to rise and fall. By using high quality amplifiers and microphones, the limits can be spread into ranges where the human ear is no longer able to hear their results and thus get out of interest. The human ear normally is not able to hear sound above 20 kHz.
3. The signal contains *noise*. Noise is the most ugly enemy of everyone who has to handle audio signals in any way. Noise is a typical analogue effect, that makes the audio signal "unsharp" and disturbed, it is always present and cannot be avoided. One can only try to use high quality components that produce as low noise as possible, so that one cannot hear it. Normally noise has a certain volume, so that the interesting sound should be much louder in comparison to the noise. This is called the *signal to noise ratio (SNR)*, the higher it is the better the sound's quality will be. Sounds that have lower volume than the noise cannot be heart.

2.2 Digitalization

When we want to store and play audio in a computer, we must convert the analogue sound into digital data first. This process is called *digitalization*. It converts an electronic signal into a sequence of digital values.

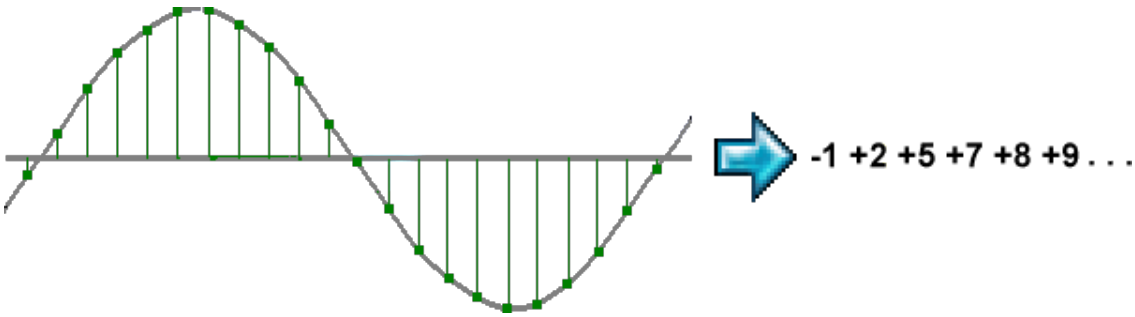


Digitalization of the electronic signal

The conversion can be understood as a repetitive measurement of the electronic signal's value at certain time, thus taking a *sample* of the signal. The result is then encoded as a digital value.

The sampling could be done in arbitrary distances or in constant intervals. The later method is much easier to handle, and thus it is normally used, with a constant rate - the so-called *sample rate*. Usual sample rates are 8000, 11025, 22050, and 44100 samples per second. In practice sample rates are also given as frequencies, in Hz or kHz.

The sample rate limits the highest frequency a digitized signal can represent. Due to Shannon's theorem the highest usable frequency is half of the sample rate, so with 44.1 kHz sample rate you cannot sample signals with more than 22 kHz. To avoid a violation of that half-sample rate rule, your soundcard already has built-in filters that filter away frequencies that are higher than half of the used sample rate.



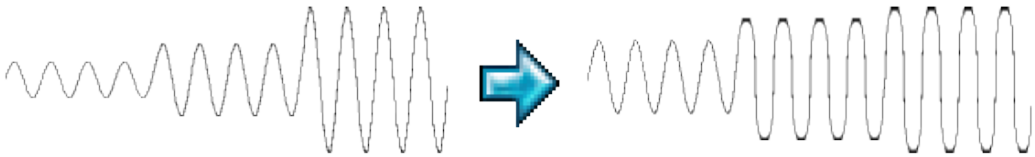
Sampled signal

2.3 Sample Encoding

The result of the digital sampling process is a sequence of single *samples*. One sample is a digital representation of a signal's value at a certain time.

The value of a sample can be interpreted and encoded in several ways. The simplest one is *linear* encoding. This means that each sample's value directly represents the analogue signal's value multiplied with a constant factor. This is easy to handle, but has the disadvantage that noise will be audible especially on low amplitudes, where it disturbs most, and less audible on high amplitudes, where it is less audible.

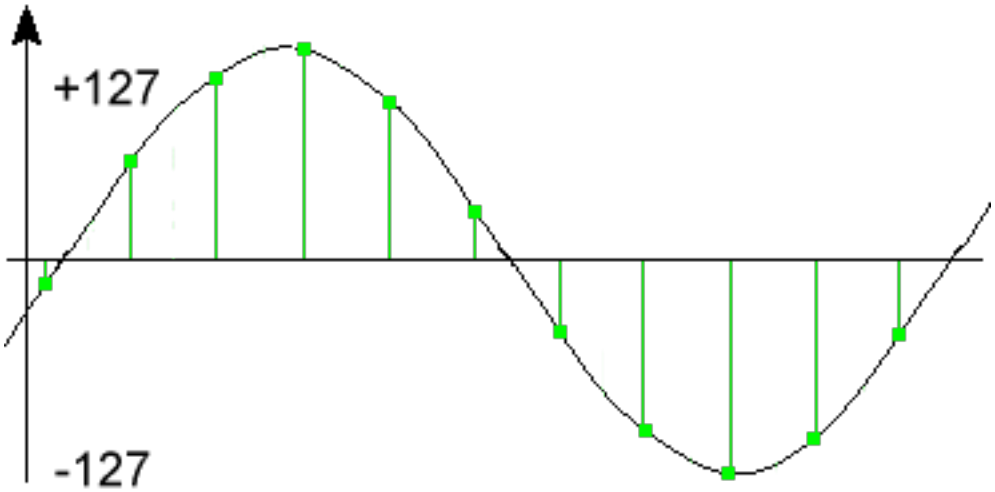
One way to reduce the influence of noise is *non-linear* encoding. This means that lower amplitudes are amplified before processing. As lower amplitudes are amplified, their distance from noise increases and the quality improves. The most common methods for this are *A-Law* and *U-Law* encoding - some standardized logarithmic amplification curves, used in digital telephony (ITU G.711 standard).



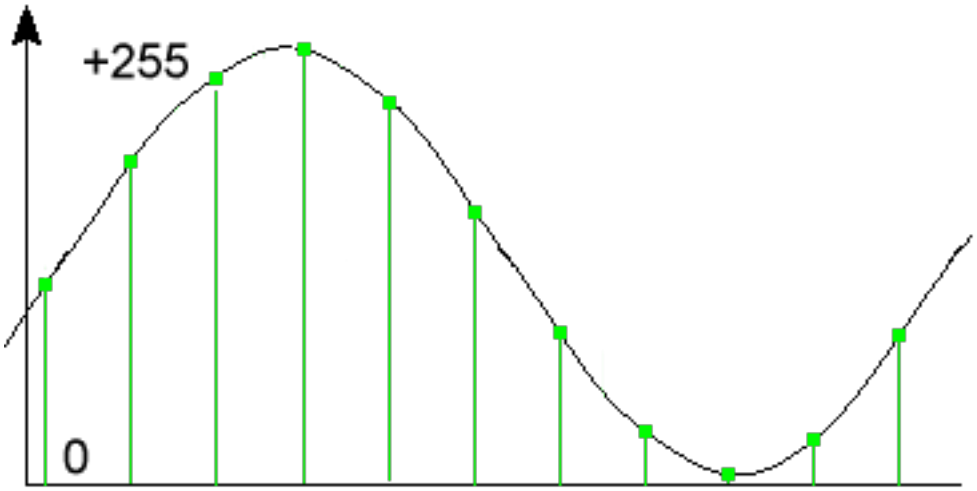
Nonlinear Encoding

2.4 Sample Formats

Samples can be stored in different formats and precisions. The most common ones are integer (fixed-point) formats, that store values with *fixed quantisations*. Depending on where the zero line is defined, it has to be destinguished between *unsigned* (only positive values, "zero line" is at half of the numeric range) and *signed* (positive and negative values) integer formats.



Signed Format



Unsigned Format

As the quantisation loses some accuracy, it produces noise, the so-called *quantisation noise*. That kind of noise has more effect on low amplitudes, so this method of storing samples is not optimal, but quite easy and very fast to handle (computers are fast in calculating with fixed point numbers).

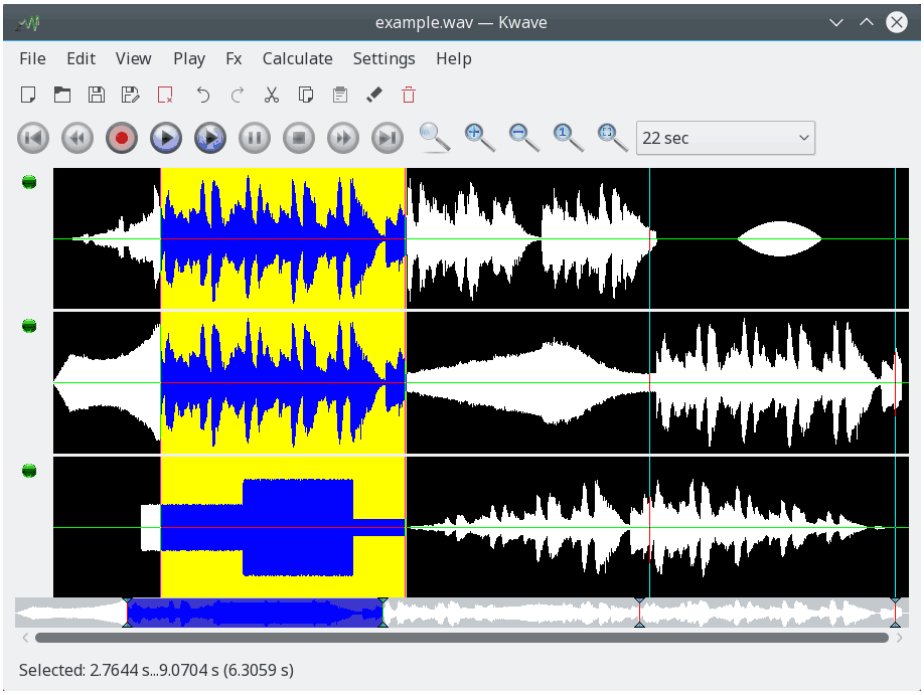
The second way of encoding samples is with *floating point* numbers. With floating point numbers, noise is spread nearly equal over all ranges of amplitudes and has advantages especially on low amplitudes. However, this format is much slower when used for processing (computers are much slower on calculating with floating point values in comparison to fixed point numbers).

> **NOTE**
> Kwave internally uses *signed integer* format with 24 bit precision, stored in 32 bit integers. This has the disadvantage of higher memory consumption when processing files with lower precision (e.g. 8 bits), but processing 32 bit numbers is very fast and also leaves some reserves for internal calculations, as only 24 bits are normally used.

Chapter 3

Using Kwave

Here is a little screenshot of the Kwave main window, so that you get an impression what Kwave looks like...

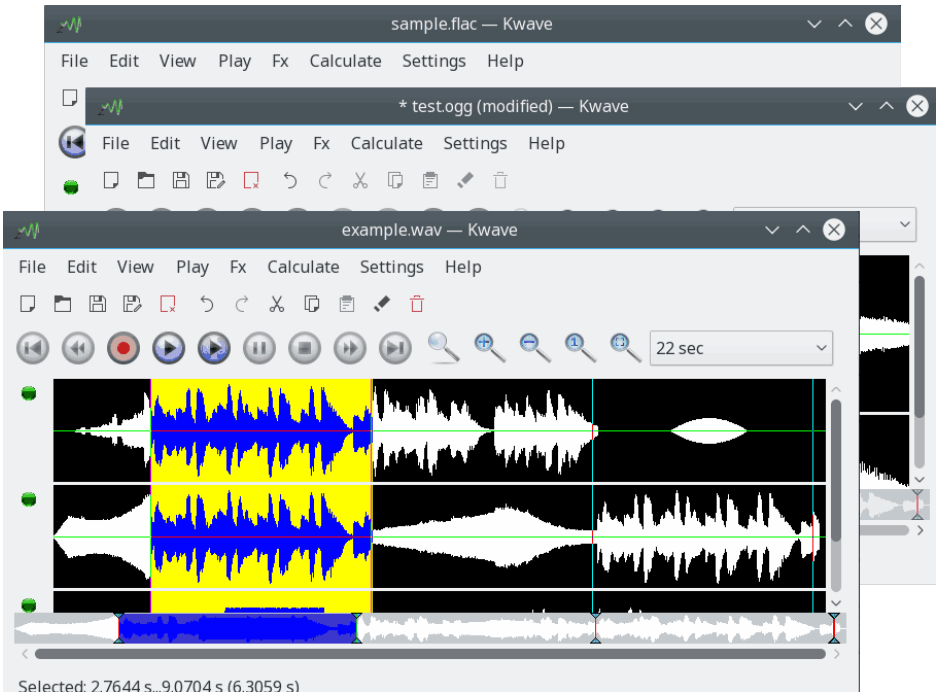


3.1 User Interface Mode

Depending on your personal preferences or use cases you can configure how Kwave handles multiple open files. You can switch this setting on the fly through the menu **Settings** → **Show Files in...**.

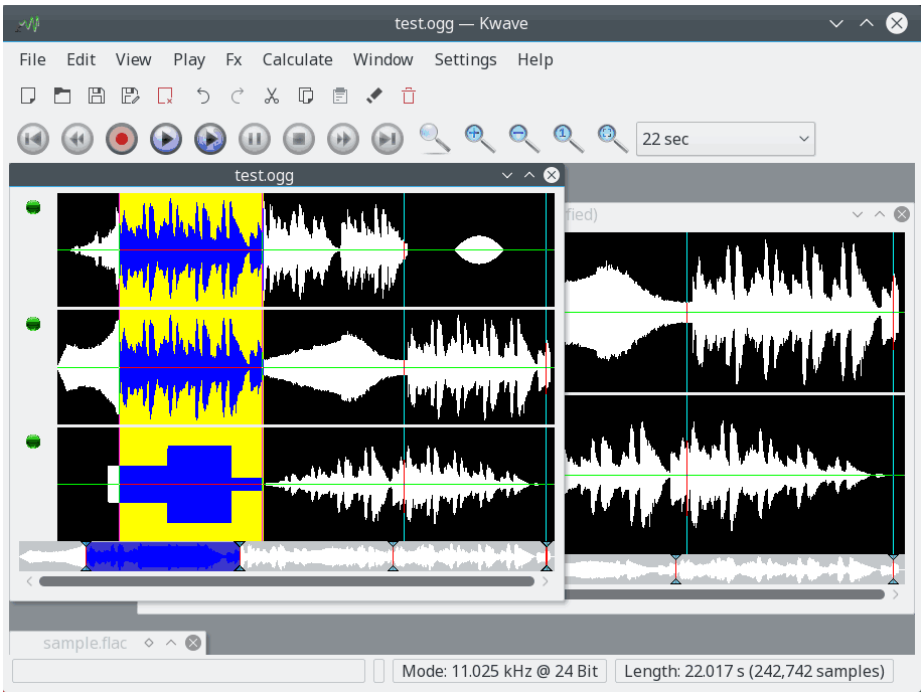
The three possible settings are:

- **Separate Windows (SDI)**: When using the *Single Document Interface* (SDI), each file will be shown in a separate main window.



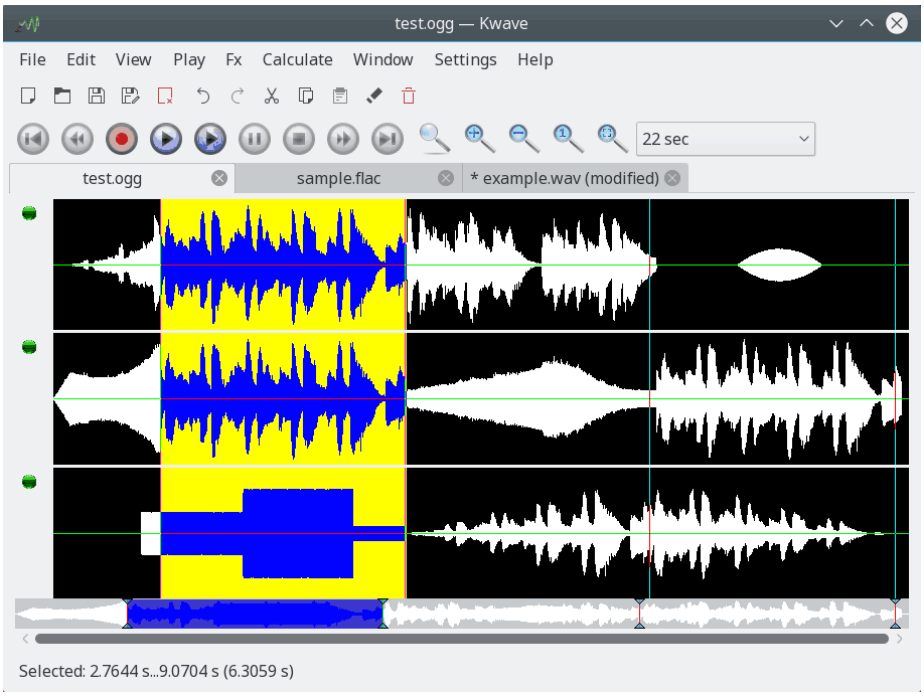
- **Same Window (MDI)**: When using the *Multi Document Interface* (MDI), there is only one Kwave main window, but within this window you have an area which shows sub-windows, which can be resized, moved, tiled, cascaded or minimized.

  You can select a window from the **Window** menu or cycle through the sub-windows with **Ctrl-Tab**.



- **Tabs**: This is a variant of the MDI interface, where files are shown within separate tabs. You may know this mode from some popular Internet browsers.

  You can activate a tab via the **Window** menu or cycle through the tabs with **Ctrl-Tab**.



3.2 Command Line

3.2.1 List of Files or Commands

If you start Kwave from the command line, you can specify a list of files that should be opened. The first specified file will be opened first, then the other files. Each file will be opened in an own new window or sub-window of the same Kwave instance. If you specify wildcards, you can open a large number of files at once.

For example, the following command starts a Kwave and opens all sounds provided by the freedesktop XDG sound theme, each in a new window or sub-window:

```
% kwave /usr/share/sounds/freedesktop/stereo/*.ogg
```

It is also possible to pass Kwave text commands, encoded as a special URI[1], this will be described in a later section.

3.2.2 GUI Type

The parameter `--gui=SDI|MDI|TAB` lets Kwave start in one of the three possible GUI modes: SDI, MDI or TAB.

3.2.3 Qt Toolkit options

In addition to a list of files, you can specify a list of *Qt toolkit* options like `-qwindowgeometry` for specifying the size and/or position of the first opened Kwave window and/or `-display` for starting the Kwave on a different display.

[1] universal resource identifier

For example, the following command starts a Kwave window with an initial width of 600 pixels and a height of 400 pixels, with the right border positioned 30 pixels away from the right and 0 pixels away from the top of the screen.

```
% kwave --disable-splashscreen -qwindowgeometry 600x400-30+0
```

3.2.4 Startup Options

With the option `--disable-splashscreen` you can disable the splash screen that comes up when starting Kwave. This might be useful when you start Kwave from a script.

The command line option `--iconic` lets Kwave start up minimized (iconified). This might be useful when you want to start Kwave without GUI interaction, e.g. when running from a script. This option also implicitly disables the splash screen!

By using the command line option `--logfile=kwave.log` you can log the sequence of actions of a Kwave session into a file. This is useful for debugging, you might be asked for such a logfile when reporting an error.

3.3 Opening and Saving files

Opening files with Kwave works like in most other applications, you can

- specify a list of files on the command line when starting Kwave,
- open an empty Kwave window (for example with **File** → **New...** (**Ctrl-W**)) and put a file into it via drag and drop, or you can
- open a file through the menu with **File** → **Open** (**Ctrl-O**)
- or one of the last recently opened files under **File** → **Open Recent**
- save the current file with **File** → **Save** (**Ctrl-S**),
- save under a different name with **File** → **Save** → **As...** (**Shift-Ctrl-S**)
- save all areas that are separated by markers, each one to an own file, with **File** → **Save** → **Blocks...**
- or only the current selection with **File** → **Save** → **Selection...**

3.3.1 Supported File formats

Kwave supports the following file formats:

- The favourite file format of Kwave is (like you can guess from the name) `.wav`. This format is very common to other "operating systems" and also is commonly used within the Plasma environment.
- The second format that Kwave supports is "ASCII". You can export to and also import from ASCII. Please be aware that storing in this format might produce very large files! The file format will be described below.
- `.mp3` and `.mp2` import is available through libmad for the MP3 decoding in combination with id3lib for decoding ID3 tags and lame for encoding.
- Ogg/Vorbis (`*.ogg`) import and export. See https://www.xiph.org for details.

- FLAC (`*.flac`) import and export. See https://xiph.org/flac/ for details.
- Additionally you can import file formats like `*.8svx` (Amiga IFF/8SVX Sound File Format), `*.au` (NeXT, Sun Audio), `*.aiff` (Audio Interchange Format), `*.avr` (Audio Visual Research File Format), `*.caf` (Core Audio Format), `*.nist` (NIST SPHERE Audio File Format), `*.sf` (Berkeley, IRCAM, Carl Sound Format), `*.smp` (Sample Vision Format), `*.snd` (NeXT, Sun Audio), `*.voc` (Creative Voice) and others through the audiofile plugin.

3.3.2 Converting to and from .wav

The best way to work with formats other than those supported by Kwave is to use an external converter program. A good set of tools for this is in the SoX package, they have also some nice documentation!

The plans for future include support for import and also export filters for more formats and maybe some filter that uses a user-definable script with a call to an external filter, so that even formats not supported by `SoX` can be read and/or written.

3.3.3 Format of ASCII files

The ASCII format is quite useful for scientific and educational purposes. Due to it's simple format, you can either write simple files on your own with a text editor or you can use the output of some other application and convert it into ASCII. As the format is *really* simple, you should not have big problems in writing a converter and most scientific applications use to have some kind of their own ASCII format for export.

The format of an ASCII file is quite simple and has the following rules:

1. At the start of the file comes a block of properties, with one property per line.
2. Each property line starts with `##`.
3. After the properties comes a list of samples, with one sample per line. When using multiple channels, the samples are separated by commas.
4. Lines might end with a carriage return and/or a line feed character (so DOS files are supported too). But when saving, files will always be saved with line feed character as the end of the line.
5. Empty lines and characters after a `#` are treated as comments and are ignored.
6. Values have to be given in signed integer format with a 24 bit range, which is the internal storage format of Kwave.
7. Everything after a `#` (except property lines, see above) will be treated as comment and will be ignored. Empty lines will also be ignored.

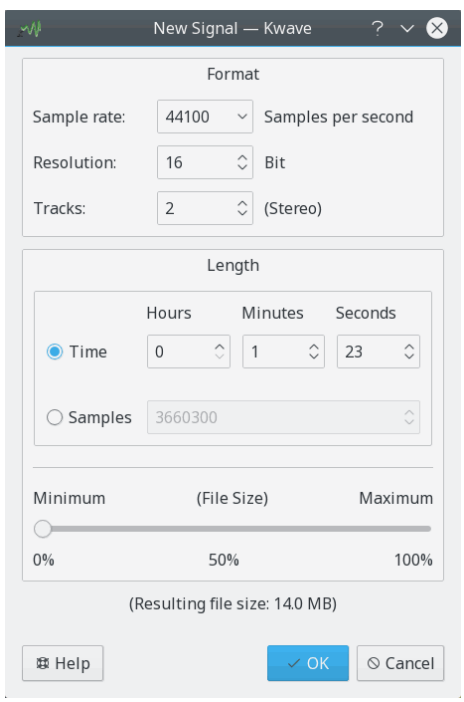
Here is an example of a simple ASCII file that represents a sine wave with eleven samples:

Example 3.1 content of an ASCII file with a single sine wave

```
## 'rate'=44100
## 'tracks'=2
## 'bits'=16
## 'length'=11
## 'Date'='2013-11-09'
## 'Software'='Kwave-0.8.11 for KDE 4.11.3'
  5930496,    5930496 #            0
        0,    8388352 #            1
 -5930752,    5930496 #            2
 -8388608,          0 #            3
 -5930752,   -5930752 #            4
        0,   -8388608 #            5
  5930496,   -5930752 #            6
  8388352,          0 #            7
  5930496,    5930496 #            8
        0,    8388352 #            9
 -5930752,    5930496 #           10
# EOF
```

3.4 Creating a New File

You can create a new and empty file menu under **File** → **New...**.



You can select the sample rate, resolution in bits per sample and the number of tracks. Per default the file format will be ".wav", but it can still be changed at the time when the file is saved.

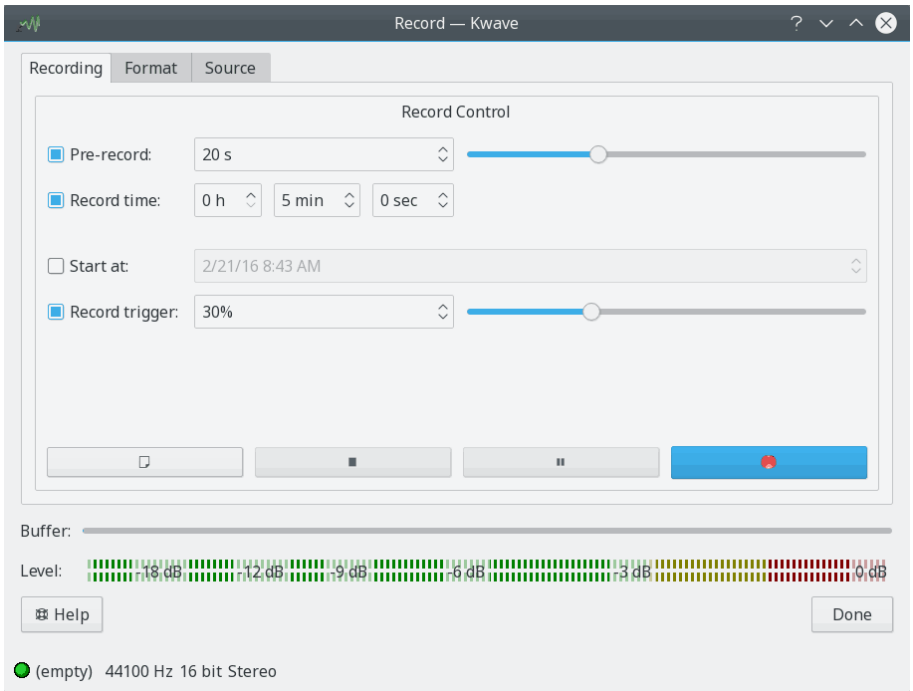
The length of the new signal can be set by time (hours, minutes, seconds) or by the number of samples. Additionally you can select it relative to the highest possible length, which is limited by the available memory and Kwave's internal limit (2 GB).

3.5 Recording

Kwave is able to record audio data from various sources, with all sample rates, sample formats and other modes that your sound hardware supports. Currently Kwave records through the old OSS sound interface, and since v0.7.4 also the newer and more powerful ALSA interface that is the preferred choice for linux kernel 2.6.

The recording can be reached from the menu under **File** → **Record**.

Here is a screenshot of the Kwave record dialog, showing the first page with the recording controls during a running recording session. Like in most dialogs of Kwave you can get some help or see tooltips on the controls.



Here you have the following controls:

- **Pre-Record:** If the pre-recording feature of Kwave is enabled and the recording is started, Kwave records into an internal buffer which is some seconds long. If you press the **Record** ( ) button again, then the recording really starts, and also keeps the already pre-recorded data. This is useful for example if you want to record your favorite song from radio, but you recognize too late that the song has started. In this case you can still press the record button and get the start of the song from what Kwave has already pre-recorded before, so that you will no longer miss a start.
- **Record Time:** If the length of the recording should be limited to some time, you can activate this setting and select a time in hours, minutes, seconds for your recording. If this option is not enabled, the recording runs until you press the **Stop** ( ) button.
- **Start At:** If this setting is activated, you can set a date and time when the recording will be started. Please keep in mind that if the configured time is in the past, the recording will start immediately.
- **Record Trigger:** If enabled, the recording starts only if the volume of the input goes over a certain limit, which can be defined from 0 to 100% of the highest possible input volume. This

is useful if you do not want to record leading silence. (Hint: combine this with the prerecording feature mentioned above to catch also some seconds before reaching the trigger, so that you don't miss any silent fade-ins.)

- The **New** ( ) button is active when the recording is not running or is finished, to discard the current file content and start again.

- The **Stop** ( ) button is active when the recording or pre-recording is running or Kwave is waiting for the trigger. If pressed, the current progress will be stopped.

- The **Pause** ( ) button is active when the recording or pre-recording is running. The first time you press it, the recording will be halted and the button starts blinking. When you press it again the button will stop blinking and recording will continue immediately, without waiting for a trigger.

- The **Record** ( ) button starts the recording and/or prerecording, depending on the features enabled above:
  1. If neither prerecording nor trigger level are used, the recording starts as soon as you press the record button.
  2. If prerecording is not used and a trigger level is set, the first press will let Kwave wait for the trigger level to be reached. While waiting for a trigger, you can force the recording to start immediately by pressing the record button again, otherwise the recording will start automatically when the trigger level has been reached.
  3. If prerecording is enabled, the first press starts only the prerecording and the second press really starts the recording.

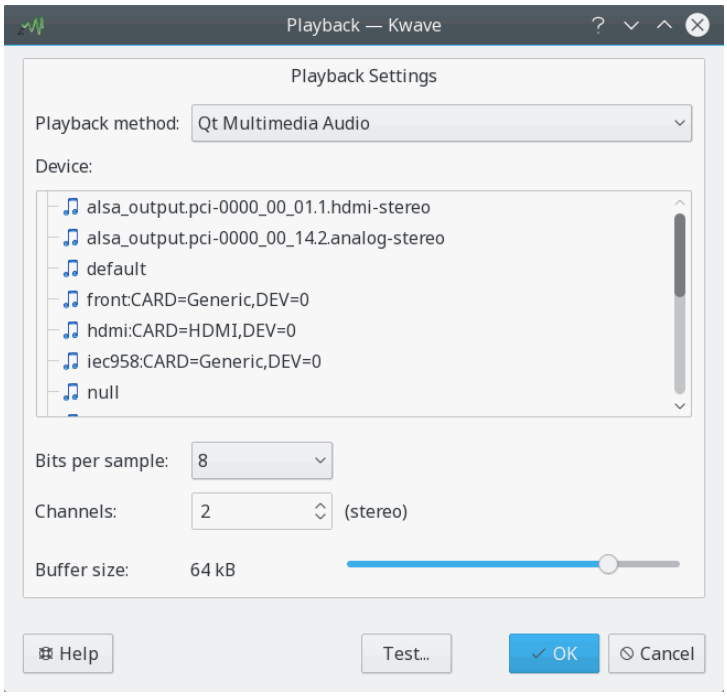
3.6 Playback

Depending on the compilation options Kwave is able to play sounds trough one of the following playback methods:

- ALSA (Advanced Linux Sound Architecture): Supercedes OSS, supports more features and more hardware. Might collide with other sound applications like OSS does, but has a plugin called "dmix" as a way out. Newer versions of ALSA use a dmix like plugin per default, so this should the best choice for you!

- OSS (Open Sound System): The oldest linux implementation, capable of mono and stereo output. Deprecated since linux kernel 2.6, but still wide spread. Might collide with other sound applications, only one application at a time can use OSS playback !

Before trying to play sounds, you should take a look on the playback configuration dialog:



Currently Kwave supports only 8 and 16 bit playback, with mono or stereo output through the OSS interface, but many also all modes your sound hardware supports through the ALSA interface.

If your sound file uses more or less channels than the playback allows, all channels will be mixed together during playback. For example if you have a file with three channels and you use stereo playback, the left channel will play channel 0 (upper) and half of channel 1 (middle), the right channel will play the half of channel 1 (middle) and channel 2 (lower).

For getting a smooth playback without interruptions, you should also set the buffer size to an appropriate value. If you encounter problems with interrupted playback, you should increase the buffer size here. But the bigger you set the buffer, the bigger is the latency between the audible sound and the display of the playback position in the signal display.

The playback settings dialog also provides a button for playing a simple test sound. You should hear a 440Hz tone that wanders over all speakers, from one to the next.

Once you have configured playback, you can use the playback controls of the Kwave main window or through the **Play** menu or with keyboard shortcuts:

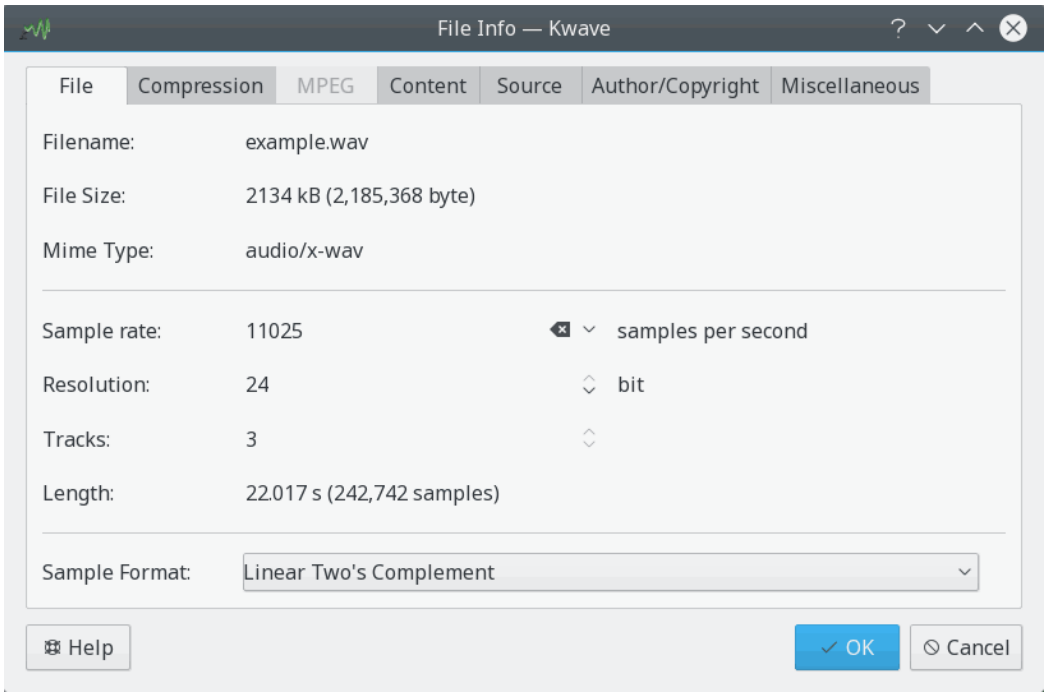
- **Play** → **Start** (**P**): Start playback of the current selection from its beginning or the whole file from the current cursor position if nothing was selected. Play only once.
- **Play** → **Loop**: Like before, but repeat in a loop.
- **Play** → **Pause** (**Space**): Pause the playback at the current position. Only available when the playback is running.
- **Play** → **Continue** (**Space**): Continue the playback from the position where it has been paused. Only available if the playback is paused.
- **Play** → **Stop** (**Esc**): Stop the playback, go back to the start of the selection.

3.7 File Properties

Kwave is able to handle several meta information that is stored within an audio file. It tries to import and export as much of that information as possible. For example, if you import an MP3

file with ID3 tags, you can keep that information when exporting to a Wave file. If Kwave would lose meta information when saving, it shows a warning.

You can view and modify the meta information under **Edit** → **File Properties...**. There you can also change things like sample format, resolution and compression.



3.8 Zooming and navigating

Kwave provides several ways to zoom and navigate, using keyboard shortcuts, menu commands, toolbar buttons and by using the mouse. The following sections should give an overview on how to use all of these functions.

3.8.1 Zooming in and out

- *zoom in:* zooms in to see more details, magnifies by factor 3.
  - menu entry / keyboard shortcut: **View** → **Zoom In** (**Ctrl-+**)
  - toolbar button: 'zoom in'
- *zoom out:* zooms in to see less details, shrinks by factor 3.
  - menu entry / keyboard shortcut: **View** → **Zoom Out** (**Ctrl--**)
  - toolbar button: 'zoom out'
- *zoom selection:* zooms to a factor where the current selection is completely visible in the current view.
  - menu entry / keyboard shortcut: **View** → **Zoom to Selection** (**Ctrl-Space**)
  - toolbar button: 'zoom to selection'
- *zoom to whole signal:* selects a zoom factor that makes the whole signal visible in the current window.
  - menu entry: **View** → **Zoom to whole signal**

  - toolbar button: 'zoom all'
- *zoom to 100%:* zooms in up to a scale where one sample is represented by one pixel on the screen.
  - menu entry: **View** → **Zoom to 100%**
  - toolbar button: 'zoom to 100%'
- *select predefined zoom:* select a zoom factor from the zoom combo box in the toolbar.

3.8.2 Scrolling left and right

- *goto position:* opens a dialog to enter the position where you want scrolls the current view.
  menu entry / keyboard shortcut: **View** → **Go to Position...** (**Ctrl-G**)
- *to begin:* scrolls the current view so that it starts at the beginning of the signal.
  menu entry / keyboard shortcut: **View** → **Begin** (**Ctrl-Home**)
- *to end:* scrolls the current view so that it ends at the end of the signal.
  menu entry / keyboard shortcut: **View** → **End** (**Ctrl-End**)
- *previous page:* scrolls to the position right before the current view (left).
  menu entry / keyboard shortcut: **View** → **Previous Page** (**PgUp**)
  toolbar button: 'previous page'
- *next page:* scrolls to the position right after the current view (right).
  menu entry / keyboard shortcut: **View** → **Next Page** (**PgDn**)
  toolbar button: 'next page'
- *scroll right:* scrolls to the end of the signal by 1/3 of the current view.
  menu entry / keyboard shortcut: **View** → **Scroll right** (**Right**)
  toolbar button: 'scroll right'
- *scroll left:* scrolls to the start of the signal by 1/3 of the current view.
  menu entry / keyboard shortcut: **View** → **Scroll left** (**Left**)
  toolbar button: 'scroll left'

3.8.3 Using the overview

The main screen of Kwave shows a small *overview* of the whole signal above the horizontal scroll bar of the main window. This overview also provides some functionality for navigating:

- *single click with left mouse button*: directly move the current view to the clicked position.
- *double click with left mouse button*: directly move the current view to the clicked position and additionally zoom in.
- *double click with left mouse button, with Shift pressed*: directly move the current view to the clicked position and additionally zoom out.

3.8.4 Vertical zoom

You can zoom the current view vertically by pressing the **Alt** key and scrolling with the mouse wheel.

3.9 How to select

Kwave allows you to select a continuous range of samples as well as any combination of channels (if you edit a multi-channel file). By selecting a range of samples (time scope) all following commands will be limited to that range and by de-selecting a channel its content will not be changed.

3.9.1 Selecting channels

Selecting or de-selecting a channel is quite simple. Just click on the lamp symbol on the left side of the signal to toggle its state:

a green lamp means ″enabled″, whereas

a red lamp means ″disabled″.

> NOTE
> Note: If a channel is de-selected it will also not be audible for playback!

3.9.2 Selecting samples

If you select a range of samples in Kwave, that range will be *inclusive*. That means that the first and the last selected sample both belong to the selection and will be used for the following actions. So even if you not selected a *range* but only a single sample, the selection will never be really ″empty″. So for example if you see no selected range, the ″delete″ function applies to that single sample.

The easiest way of selecting a range of samples is just to do that with mouse. It works like you are used from other applications: just press the left mouse button at the point you want to let the selection start and release the button where you want it to end.

If you want to adjust or move the selection's start or end, you can move the mouse cursor near to the start or the end of the selection until it changes from the standard arrow cursor into the left-right arrow cursor and then press the left mouse button and adjust.

You can also extend or shrink the selection to a specific point by holding down the **Shift** key while clicking with the left mouse button. Depending on which border is nearer, the left or right border of the selection will be set until the new position. If there was nothing selected, will be set from the beginning.

There are also some functions available via the menu and of course some keyboard shortcuts:

- select the whole signal: **Edit** → **Selection** → **All** (**Ctrl-A**)
- select a range: **Edit** → **Selection** → **Range** (**R**)
- the currently visible area: **Edit** → **Selection** → **Visible Area** (**V**)
- the next block of samples, starting one sample after the end of the current selection and with the same length: **Edit** → **Selection** → **Next** (**Shift-+**)

  (Hint: use the ″**+**″ key from the numeric keypad!)
- the previous block of samples, ending one sample before the start of the current selection and with the same length: **Edit** → **Selection** → **Previous** (**Shift--**)

  (Hint: use the ″**-**″ key from the numeric keypad!)

- remove any selection and select "nothing": **Edit** → **Selection** → **Nothing** (**N**)
- expand the selection to the start of the signal (first sample): **Edit** → **Selection** → **To Start** (**Shift-Home**)
- expand the selection to the end of the signal (last sample): **Edit** → **Selection** → **To End** (**Shift-End**)
- expand the current selection left and right up to the next label (or start/end of the signal if there is none), starting at the current cursor position: **Edit** → **Selection** → **Expand to labels** (**E**)
- select the area between the next two labels that are right from the current selection or up to the end of the signal: **Edit** → **Selection** → **To Next Labels** (**Ctrl-Shift-N**)
- select the area between the previous two labels that are left from the current selection or up to the start of the signal: **Edit** → **Selection** → **To Previous Labels** (**Ctrl-Shift-P**)

3.10 Clipboard

Kwave uses the clipboard of Plasma. This way it is possible to exchange audio data between different Kwave windows. It might be possible as well to exchange data between Kwave and other audio applications, depending on their ability to use the Plasma clipboard.

When copying data to the clipboard through the `copy` function Kwave uses the mime type `audio/vnd.wave` as data format, conforming to RFC 2361 which is the same as the well known `wav` format. When pasting from the clipboard into Kwave all data formats that are available as file import formats are supported, like for example Ogg/Vorbis, FLAC and so on.

3.11 Drag and Drop

Kwave supports the KDE Frameworks Drag and Drop protocol. This enables you to open files just by picking them up in a Dolphin or Konqueror window or the Desktop and let them drop into a window of Kwave.

Please note that if you drop a file into a Kwave window that already contains an opened file, the currently opened file will be closed first and then the file you dropped will be opened in it. If you don't want that, you should open a new empty Kwave window first.

You can also select a range of samples and drag or drop them into a Kwave window. Per default the drag operation is done in *move* mode where the selected range is deleted from the original place and inserted at the drop position. By pressing the **Ctrl** key you can modify this and drag in *copy* mode instead.

Chapter 4

Automization and Scripting with Kwave

Kwave since its first version uses an internal text command language. This command language is used internally for menu handling, GUI control, builtin effects and plugin invocations. The commands will be described later in the section .

4.1 General Syntax

- All commands consist of a *command name* and an optional *parameter list* in round brackets, depending on the command.
- Allowed characters for command names are *letters*, *digits* and *colon*. Commands are case sensitive and are always in lower case.
- Parameters within a parameter list are separated by *comma*.
- Numerical parameters can be given as fixed point numbers or as floating point numbers, *using a dot as decimal separator*.
- String parameters are automatically trimmed (all white space at the start and at the end is removed). If that is not wanted, they can be surrounded by double quotes (˝). If a string parameter contains special characters (like ',', ';', '#' or a '\' itself), these special characters have to be escaped by preceding a '\'.
- Multiple commands can be concatenated to a *command list* by using a ';' as separator.

Example:

```
fileinfo(Comments,"This is an \"example\" comment.")
```

This example consists of the command **fileinfo**() and has two parameters: the keyword `Comments` and the text `'˜This is an \˜example\˜ comment.˜'`. (These parameters are explained in the corresponding section in the command reference).

4.2 Using the Command Line

In addition to the command line options listed in the section Command Line which are used to start Kwave in iconified mode or without splash screen, you can pass text commands on the command line, encoded in a special URI format:

```
kwave:command[?parameter[,parameter ...]  ]
```

The rules for transforming a Kwave text command into a valid URI are as follows:

- The URI starts with the word `kwave`, followed by a ':' and the command name.
- If the command has parameters, they have to be appended after the command name, using a '?' as separator.
- Multiple parameters can be appended by using a ',' as separator.
- All special characters in command name and parameters have to be URL encoded. Here a list of the translations:

original	encoded		original	encoded		original	encoded		original	encoded
(space)	%20		(	%28		:	%3A		\	%5C
!	%21		)	%29		;	%3B		]	%5D
"	%22		*	%2A		<	%3C		^	%5E
#	%23		+	%2B		=	%3D		_	%5F
$	%24		,	%2C		>	%3E		`	%60
%	%25		-	%2D		?	%3F		{	%7B
&	%26		.	%3E		@	%40		|	%7C
'	%27		/	%2F		[	%5B		}	%7D
									~	%7E

Table 4.1: URL Encoding Translation Table

4.3 Kwave Script Files

4.3.1 General Structure

A Kwave *script* consists of a list of lines, where each line can be:

- a single *command*,
- a *command list*, with two or more commands concatenated by a ';'.
- a *comment*,
- a *label*
- or an *empty line*, that contains white space only

4.3.2 Comments and Empty Lines

All characters that follow a '#' (except when used in quotes or when escaped) are treated as comments, they will be silently ignored.

Lines that contain only white space or comments are ignored as well.

4.3.3 Termination

A Kwave script terminates either when all commands have been executed successfully without an error or when a command has returned an error code. There is no special command for aborting the execution of a script. If you want to implement a possibility for the user to end a script, you can use the command '**msgbox**(`text`)'. This shows a message box with the two buttons **OK** (which lets the script continue) and **Cancel** (which returns an error code and stops the script).

4.3.4 Labels

Lines that consist only of an identifier, followed by a ':' are treated *labels*. They can be referenced later in the script by the special keyword **GOTO** [1] , which makes the execution of the script continue at the location of that label (see example below).

A line that contains a label must not contain any other content (except comments or white space) after the ':'.

Example:

```
start:  # <= this is a label
        # do something...
        msgbox(once again?)
        GOTO start
```

4.4 Command Reference

4.5 a

4.5.1 about_kde

4.5.1.1 Syntax: about_kde()

Shows a dialog window with information about the KDE community.

4.5.2 add_track

4.5.2.1 Syntax: add_track()

Adds a new track after all existing tracks.

4.5.2.2 See also

insert_track()

[1] Note: Please don't mix up the keyword '**GOTO**' with the text command '**goto** (`position`)' !

4.6 c

4.6.1 clipboard_flush

4.6.1.1 Syntax: clipboard_flush()

Discards the current content of the clipboard (might free some memory).

4.6.2 close

4.6.2.1 Syntax: close()

Closes the current file. If the GUI is configured to MDI or Tab mode, this also closes the corresponding sub-window.

4.6.2.2 See also

open(*filename*), **quit**()

4.6.3 continue

4.6.3.1 Syntax: continue()

Corresponds to the **Continue** toolbar button and lets the playback continue if it is paused.

4.6.3.2 See also

pause()

4.6.4 copy

4.6.4.1 Syntax: copy()

Copies the content of the current selection to the clipboard. If the selection is empty, this command does nothing and the content of the clipboard remains unchanged. Only the content of the currently selected tracks is copied to the clipboard!

4.6.4.2 See also

paste()

4.6.5 crop

4.6.5.1 Syntax: crop()

Crops the signal to the current selection by deleting everything that is after and before the current selection. Affects all tracks. If nothing is selected this command does nothing.

4.6.6 cut

4.6.6.1 Syntax: cut()

Copies the content of the current selection to the clipboard and removes it from the signal. If the selection is empty, this command does nothing and the content of the clipboard remains unchanged. Only the content of the currently selected tracks is copied to the clipboard, but the selected range is deleted from all tracks.

4.7 d

4.7.1 delayed

4.7.1.1 Syntax: delayed(*milliseconds*, *command*)

Executes a *command* after a given delay. Please note that the command is executed asynchronously after the given time has elapsed. Multiple commands can be queued, where the delays are relative to the last queued command. This command is intended to be used for queuing commands when taking screenshots for documentation purposes.

4.7.1.2 Parameters

milliseconds:	number of whole milliseconds to wait before executing the command
command:	a command, including parameters to be executed after the given delay

4.7.1.3 See also

sync(), **window:resize**(), **window:click**(), **window:sendkey**(), **window:close**(), **window:screenshot**()

4.7.2 delete

4.7.2.1 Syntax: delete()

Deletes the currently selected range of samples. If the selection is empty, this command does nothing. Affects all tracks.

4.7.3 delete_track

4.7.3.1 Syntax: delete_track(*index*)

Deletes a track, identified by its index (starting from zero). If no track with the given index exists, this command exits with an error.

4.7.3.2 Parameters

index:	index of the track to delete, starting with 0

4.7.4 dump_metadata

4.7.4.1 Syntax: dump_metadata()

Prints a list of all meta data entries to the console, for diagnostic purposes. (Only available when Kwave has been compiled with the option `WITH_DEBUG` switched on).

4.8 e

4.8.1 expandtolabel

4.8.1.1 Syntax: expandtolabel()

Expands the current selection to the labels left and right from the current selection borders. If the border of selection already is on a label, it stays unchanged. If there is no label left or right of the current selection, it will be expanded to the start or end of the file.

4.9 f

4.9.1 fileinfo

4.9.1.1 Syntax: fileinfo(*index*)

Set a file info entry to a new value.

4.9.1.2 Parameters

keyword:	keyword of the entry
value:	value of the entry

4.9.2 forward

4.9.2.1 Syntax: forward()

Corresponds to the **Forward** toolbar button. If the playback is currently running, it skips forward by 1/10 of the visible range. If the playback is not running, this does the same as the command **view:scroll_right**().

4.9.2.2 See also

view:scroll_right(), **rewind**()

4.10 g

4.10.1 goto

4.10.1.1 Syntax: goto(*pos*)

Sets the cursor to the given position and makes it visible in current view. After this the selection has zero length.

4.10.1.2 Parameters

pos:	position in samples where to go to

4.11 i

4.11.1 insert_at

4.11.1.1 Syntax: insert_at(*pos*)

Inserts the content of the clipboard at the given position, like the command **paste**(). If the clipboard is currently empty, this function does nothing.

4.11.1.2 Parameters

pos:	position in samples where to insert

4.11.1.3 See also

paste()

4.11.2 insert_track

4.11.2.1 Syntax: insert_track(*index*)

Inserts a new track at the given index, using the current length and sample rate settings of the signal. If the index is higher than or equal to the current number of tracks, it will be appended as the last track, same as by the command **add_track**(). The index of all existing tracks at and after the given index will be incremented by one.

4.11.2.2 Parameters

index:	index of the track to insert, starting with 0

4.11.2.3 See also

add_track()

4.12 l

4.12.1 label:add

4.12.1.1 Syntax: label:add(*pos*[,*text*])

Add a new label at a given position. If the given position already contains a label, then this command does nothing. The label can be given an optional description.

4.12.1.2 Parameters

pos:	position in samples where to insert the label
text:	some descriptive text (optional)

4.12.2 label:delete

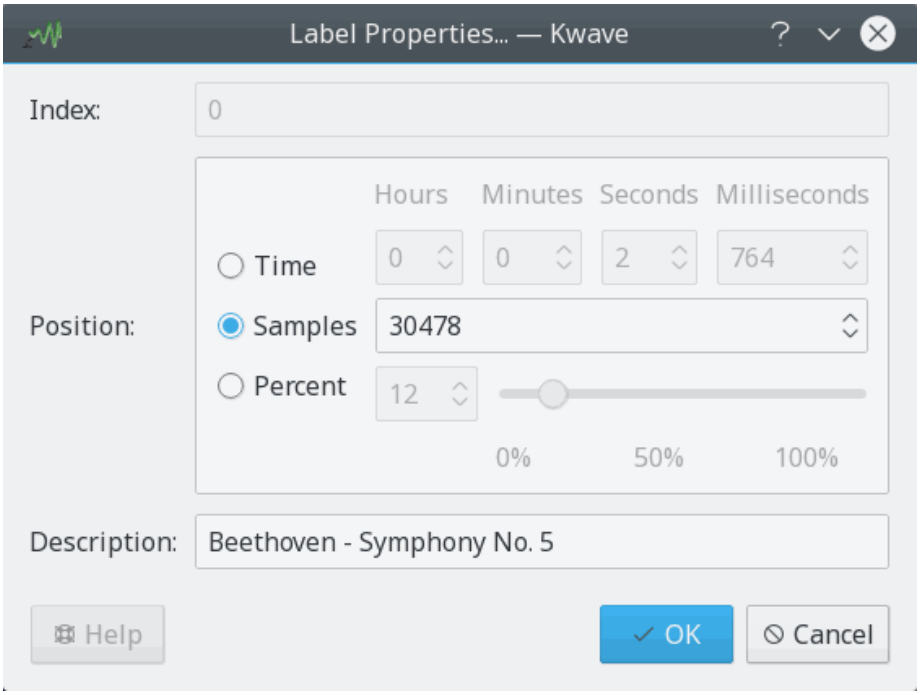
4.12.2.1 Syntax: label:delete(*index*)

Deletes a label, identified by its index (starting from zero), or all labels when using the special value -1 as index. If no label with the given index exists, this command does nothing.

4.12.2.2 Parameters

index:	index of the label to delete, starting with 0 or -1 to delete all labels

4.12.3 label:edit



4.12.3.1 Syntax: label:edit(*index*)

Opens a dialog window in which the user can edit the position and the description of a label, which is identified by its index (starting from zero). If no label with the given index exists, this command does nothing.

4.12.3.2 Parameters

index:	index of the label to edit, starting with 0

4.12.4 loadbatch

4.12.4.1 Syntax: loadbatch(*filename*)

Opens a Kwave script file and processes the commands in it. Uses the context of the currently opened file or the current main window if no file is loaded.

4.12.4.2 Parameters

filename:	name of the kwave script file including path and extension

4.12.5 loop

4.12.5.1 Syntax: loop()

Corresponds to the **Loop** toolbar button. Starts the playback (if it is not already running) and lets it play in a loop.

4.12.5.2 See also

playback_start()

4.13 m

4.13.1 menu

4.13.1.1 Syntax: menu (*command*, *path*, [*hotkey*], [*id*])

This is a very powerful command, which is used to add a new entry to the menu or to modify an existing entry. It determines which *command* is executed when the menu entry is activated, which *icon* is shown in the menu and which *hotkey* is used. Each menu entry can be *disabled* or *hidden*, can be assigned a unique *id* and can also join a *menu group*.

You normally do not need this command within a Kwave script!

4.13.1.2 Parameters

`command`:	A text command (including parameters) or a command list that will be executed when the menu entry gets activated. If a menu entry does not have a corresponding command (for example if it is a sub menu and not menu entry), you should use the special command **ignore**().
`path`:	The path within the menu, using a '/' as separator. The last part of the path can be a *sub command* which modifies some property of the menu entry (see below). The last portion of the path (that is not a sub command) produces a menu entry, the parts before produce the main menu entry or sub menus that lead to it. Main menu entries or sub menus are automatically created when a menu entry is created, you do not need them manually.
`hotkey`:	A bitmask that consists of a combination of predefined *keys* and *modifiers*, concatenated with a '+'. The *key* can be either a digit, an upper case letter, a function key ('F1' ... 'F12') or any other key name understood by the Qt class '`QKeySequence`', including key names for predefined actions (like for example '`::Copy`'). Typical *modifiers* are '`SHIFT`', '`ALT`' and '`CTRL`'.

id:	A unique id that can be internally used to identify this menu entry or menu / sub menu. Only uppercase letters, digits and '_' should be used and it should start with 'ID_'. It is in your own responsibility to make sure that the same id is not used twice.

4.13.1.3 Sub Commands

#checkable:	Makes the menu entry *checkable*, so that it can be switched on or off.
#disabled:	*Disables* the menu, sub menu or menu entry.
#enabled:	*Enables* the menu, sub menu or menu entry.
#exclusive(*group*):	Adds the menu entry to an *exclusivegroup* (one of many selection). The group that is given as parameter should not be used for any other purpose. Only one entry within that group can be selected at a time.

#group(`list`):	Adds the menu, sub menu or menu entry to one or more a *groups*, so that the application can enable/disable a bunch of menu entries without need to know all their unique ids. Multiple groups can be passed as a list with a '`,`' as separator. Group names have to start with a '`@`'. The following groups are predefined: **@CLIPBOARD:** Only enabled when the clipboard is not empty. **@LABELS:** Only enabled when the current signal contains at least one label. **@NOT_CLOSED:** Enabled when the current signal is not closed (the signal might be empty or zero length). **@SELECTION:** Enabled when the selection is not empty (more than one sample is selected). **@SIGNAL:** Enabled when there is some signal loaded and it is not empty or zero length.
#hidden:	*Hides* the menu, sub menu or menu entry.
#icon(`name`):	Assigns an *icon* to the menu entry. The icon *name* should correspond to an icon file (without path and file extension) that is installed on the system or with Kwave.
#listmenu(`id`,`command`):	Inserts a placeholder for a list of menu entries into a sub menu. The unique *id* specified in this sub command is used to add/remove or clear the list of menu entries. The *command* parameter has to contain '`%1`' as parameter, which will be replaced with the text of the menu entry when it is activated. This sub command is internally used for the list of recent files, list of tracks and window list.
#separator:	Inserts a separator into a sub menu.

4.13.2 msgbox

4.13.2.1 Syntax: msgbox(*text*)

Shows a message box with some *text* and the two buttons **OK** (returns without error code) and **Cancel** (returns and error code). You can use this command to give the user a possibility to abort a running script.

4.13.2.2 Parameters

text:	A message that will be shown in the message box, should contain a question that can be answered with **OK** or **Cancel**.

4.14 n

4.14.1 newsignal

4.14.1.1 Syntax: newsignal(*samples*, *rate*, *bits*, *tracks*)

Creates a new signal, with a given length in *samples*, a *rate* in samples per second (floating point number), a number of *bits* per sample and number of *tracks*. You can calculate the length in samples by multiplying the desired length in seconds with the sample rate.

4.14.1.2 Parameters

samples:	Length of the signal in samples.
rate:	Sample rate in samples per second.
bits:	Number of bits per sample, must not be zero, should be a number from 8...32.
tracks:	Number of tracks.

4.14.2 next

4.14.2.1 Syntax: next()

Corresponds to the **Next** toolbar button. If the playback is currently running, it skips forward to the next label. If the playback is not running, this does the same as the command **view:scroll_next_label**().

4.14.2.2 See also

view:scroll_next_label(), **prev**()

4.15 o

4.15.1 open

4.15.1.1 Syntax: open([*filename*])

Opens a file, which can be either a sound file or a Kwave script. If no file name is passed, then a dialog window will be opened that allows to select an existing file. Depending on the GUI mode the file will be opened in the context of a new sub-window (MDI and tab) or in a new main window (SDI, if there was already something loaded).

4.15.1.2 Parameters

filename:	name of a file including path and extension

4.15.1.3 See also

close()

4.15.2 openrecent

4.15.2.1 Syntax: openrecent(*filename*)

Practically the same as the command **open**(), but intended to be used internally for the list of recently opened files in the menu **File**+**Open Recent**. In this command the parameter *filename* is not optional.

4.15.2.2 Parameters

filename:	entry of the list of recently opened files

4.15.2.3 See also

open()

4.16 p

4.16.1 paste

4.16.1.1 Syntax: paste()

Replaces the current selection with the content of the clipboard. If the clipboard is empty, this command does nothing. The sample rate of the inserted data is adjusted to match the sample rate of the current signal if necessary. Only enabled tracks are affected, disabled tracks remain

unchanged. Please be aware that this might produce a time shift between enabled and disabled tracks! If the number of tracks of the clipboard data differs from the number of enabled tracks, then the data is mixed to be spread equally over all selected tracks.

4.16.1.2 See also

copy()

4.16.2 pause

4.16.2.1 Syntax: continue()

Corresponds to the **Pause** toolbar button and lets the playback pause if it is currently running, or continue if it is currently paused.

4.16.2.2 See also

continue()

4.16.3 playback_start

4.16.3.1 Syntax: playback_start()

Corresponds to the **Start** toolbar button and lets the playback start if it is currently paused.

4.16.4 plugin

4.16.4.1 Syntax: plugin(*name*, [*parameter* ...])

Executes a plugin, with an optional list of parameters. If no parameter list is given, then the setup function of the plugin will be called, using the parameters of the previous invocation or default parameters as input (normally shows a setup dialog, depending on the plugin). Please refer to the chapter about plugins for a description of the various plugins.

4.16.4.2 Parameters

name:	the (internal) name of a Kwave plugin
parameter:	a list of parameters understood by the plugin (optional)

4.16.4.3 See also

plugin:execute(), **plugin:setup**()

4.16.5 plugin:execute

4.16.5.1 Syntax: plugin:execute(*name*, [*parameter* ...])

Similar to the command **plugin**(), but without calling the setup function of the plugin if no parameters were passed.

4.16.5.2 Parameters

name:	the (internal) name of a Kwave plugin
parameter:	a list of parameters understood by the plugin

4.16.6 plugin:setup

4.16.6.1 Syntax: plugin:setup(*name*, [*parameter* ...])

Calls the *setup* function of a plugin, with an optional list of parameters. If no parameter list is given, the parameters of the previous invocation or default parameters will be used as input. This normally shows a setup dialog, depending on the plugin. Please refer to the chapter about plugins for a description of the various plugins.

4.16.6.2 Parameters

name:	the (internal) name of a Kwave plugin
parameter:	a list of parameters understood by the plugin (optional)

4.16.7 prev

4.16.7.1 Syntax: prev()

Corresponds to the **Previous** toolbar button. If the playback is currently running, it skips back to the previous label or start of the selection. If the playback is not running, this does the same as the command **view:scroll_prev_label**().

4.16.7.2 See also

view:scroll_prev_label(), **next**()

4.17 q

4.17.1 quit

4.17.1.1 Syntax: quit()

Closes the current main window, including all sub-windows. In SDI mode this is the same as the command **close**().

4.17.1.2 See also

close()

4.18 r

4.18.1 redo

4.18.1.1 Syntax: redo()

Corresponds to the **Redo** toolbar button and repeats one operation that has been reverted with **undo**().

4.18.1.2 See also

undo()

4.18.2 redo_all

4.18.2.1 Syntax: redo_all()

Similar to **redo**(), but re-does as many operations as possible.

4.18.2.2 See also

undo()

4.18.3 reenable_dna

4.18.3.1 Syntax: reenable_dna()

Some message boxes offer the possibility to prevent them from appearing again ('do not ask again'). This command makes all of them appear again.

4.18.4 reset_toolbars

4.18.4.1 Syntax: reset_toolbars()

Resets all toolbar settings, like location, icon size and text location back to defaults.

4.18.5 revert

4.18.5.1 Syntax: revert()

Reverts the currently loaded file back to the last saved state, discarding all changes that are not saved.

4.18.6 rewind

4.18.6.1 Syntax: rewind()

Corresponds to the **Rewind** toolbar button. If the playback is currently running, it skips backward by 1/10 of the visible range. If the playback is not running, this does the same as the command **view:scroll_left**().

4.18.6.2 See also

view:scroll_left(), **forward**()

4.19 s

4.19.1 save

4.19.1.1 Syntax: save()

Corresponds to the **Save** toolbar button. Saves the current file if it has modifications. If the file does not already have a name (e.g. a file that has just been created and does not yet have a file name), this command does the same as **saveas**().

4.19.1.2 See also

saveas()

4.19.2 saveas

4.19.2.1 Syntax: saveas([*filename*])

Saves the currently opened file under a given file name. If no file name is given as parameter, a dialog will be shown to select the directory and to enter a file name.

4.19.2.2 Parameters

filename:	file name for saving (optional)

4.19.3 saveselect

4.19.3.1 Syntax: saveselect()

This command does the same as **save**(), but saves only the currently selected range and the activated tracks instead of the whole file.

4.19.3.2 See also

save()

4.19.4 select_gui_type

4.19.4.1 Syntax: select_gui_type(*mode*)

Select a GUI mode, which can be either SDI, MDI or Tab mode. Please be aware that this change will immediately take effect!

4.19.4.2 Parameters

filename:	name of the mode, must be either 'SDI', 'MDI' or 'TAB'.

4.19.5 select_track:all

4.19.5.1 Syntax: select_track:all()

Mark all tracks 'enabled'. This is the same as calling the command **select_track:on**() for all existing tracks.

4.19.5.2 See also

select_track:on()

4.19.6 select_track:invert

4.19.6.1 Syntax: select_track:all()

Invert the 'enabled' state of all tracks. This is the same as calling the command **select_track:toggle**() for all existing tracks.

4.19.6.2 See also

select_track:toggle()

4.19.7 select_track:none

4.19.7.1 Syntax: select_track:none()

Mark all tracks 'disabled'. This is the same as calling the command **select_track:off**() for all existing tracks.

4.19.7.2 See also

select_track:off()

4.19.8 select_track:off

4.19.8.1 Syntax: select_track:off(*index*)

Disables a single track, so that it does not get affected by most operations.

4.19.8.2 Parameters

index:	index of the track, starting with zero

4.19.9 select_track:on

4.19.9.1 Syntax: select_track:on(*index*)

Enables a single track, so that it gets affected by all operations.

4.19.9.2 Parameters

index:	index of the track, starting with zero

4.19.10 select_track:toggle

4.19.10.1 Syntax: select_track:toggle(*index*)

Enables a track if it is currently disabled, or disables it if it is currently enabled.

4.19.10.2 Parameters

index:	index of the track, starting with zero

4.19.11 selectall

4.19.11.1 Syntax: selectall()

Selects the range of the whole signal, from the first to the last sample.

4.19.12 selectnext

4.19.12.1 Syntax: selectnext()

Selects a range of samples that starts right after the current selection, using the same length as the current selection. The selection is automatically clipped to the end of the signal. For example: if you have selected samples 1000 ... 1019, then the result will be a selection from sample 1020 ... 1039.

4.19.12.2 See also

selectprev()

4.19.13 selectnextlabels

4.19.13.1 Syntax: selectnextlabels()

Selects a range of samples between the next two labels after the current selection. If nothing is selected, it selects from the start of the signal up to the first label. Otherwise the left border of the new selection will be the position of first label after the selection (or the last label if there are no more labels right from the selection), and the right border of the new selection will be the first label after the left border of the new selection (or the end of the signal if there is none). This command returns an error when there are no labels at all.

4.19.13.2 See also

selectprevlabels()

4.19.14 selectnone

4.19.14.1 Syntax: selectnone()

Resets the selection to zero length.

4.19.15 selectprev

4.19.15.1 Syntax: selectprev()

Selects a range of samples that starts left from the current selection, using the same length as the current selection. The selection is automatically clipped to the start of the signal. For example: if you have selected samples 1000 ... 1019, then the result will be a selection from sample 980 ... 999.

4.19.15.2 See also

selectnext()

4.19.16 selectprevlabels

4.19.16.1 Syntax: selectprevlabels()

Selects a range of samples between the previous two labels before the current selection. If nothing is selected, it selects from the start of the signal up to the first label. Otherwise the right border of the new selection will be the position of first label before the selection (or the first label if there are no more labels left from the selection), and the left border of the new selection will be the first label before the right border of the new selection (or the start of the signal if there is none). This command returns an error when there are no labels at all.

4.19.16.2 See also

selectnextlabels()

4.19.17 selecttoleft

4.19.17.1 Syntax: selecttoleft()

Sets the start of the selection to the start of the signal, the end of the current selection stays unchanged.

4.19.17.2 See also

selecttoright()

4.19.18 selecttoright

4.19.18.1 Syntax: selecttoright()

Sets the end of the selection to the end of the signal, the start of the current selection stays unchanged.

4.19.18.2 See also

selecttoleft()

4.19.19 selectvisible

4.19.19.1 Syntax: selectvisible()

Selects the range of samples that is visible in the current window.

4.19.20 start

4.19.20.1 Syntax: start()

Corresponds to the **Start** toolbar button and lets the playback start from the beginning of the selection or continue if it is currently paused.

4.19.20.2 See also

stop()

4.19.21 stop

4.19.21.1 Syntax: stop()

Corresponds to the **Stop** toolbar button and lets the playback stop if it is currently running.

4.19.21.2 See also

start()

4.19.22 sync

4.19.22.1 Syntax: sync()

Waits until all commands which have been started asynchronously have finished. If nothing is currently queued for delayed execution this command has no effect.

4.19.22.2 See also

delayed()

4.20 u

4.20.1 undo

4.20.1.1 Syntax: undo()

Corresponds to the **Undo** toolbar button and reverts the last operation.

4.20.1.2 See also

redo()

4.20.2 undo_all

4.20.2.1 Syntax: undo_all()

Similar to **undo**(), but reverts as many operations as possible.

4.20.2.2 See also

undo()

4.21 v

4.21.1 view:scroll_end

4.21.1.1 Syntax: view:scroll_end()

Scrolls the current view to the *end* of the signal.

4.21.1.2 See also

view:scroll_start()

4.21.2 view:scroll_left

4.21.2.1 Syntax: view:scroll_left()

Scrolls the current view by 1/10 of the currently visible range towards the start of the signal. If the start of the signal is reached the visible area starts at offset zero.

4.21.2.2 See also

view:scroll_right()

4.21.3 view:scroll_next

4.21.3.1 Syntax: view:scroll_next()

Scrolls the current view towards the end of the signal by the currently visible range.

4.21.3.2 See also

view:scroll_prev()

4.21.4 view:scroll_next_label

4.21.4.1 Syntax: view:scroll_next_label()

Scrolls right and tries to show the next label centered in the view. If there was no label right from the current position, it will scroll to the end of the signal.

4.21.4.2 See also

view:scroll_prev_label()

4.21.5 view:scroll_prev

4.21.5.1 Syntax: view:scroll_prev()

Scrolls the current view towards the start of the signal by the currently visible range.

4.21.5.2 See also

view:scroll_next()

4.21.6 view:scroll_prev_label

4.21.6.1 Syntax: view:scroll_prev_label()

Scrolls left and tries to show the previous label centered in the view. If there was no label left from the current position, it will scroll to the start of the signal.

4.21.6.2 See also

view:scroll_next_label()

4.21.7 view:scroll_right

4.21.7.1 Syntax: view:scroll_right()

Scrolls the current view by 1/10 of the currently visible range towards the end of the signal. If the end of the signal is reached the visible area ends at the end of the signal.

4.21.7.2 See also

view:scroll_left()

4.21.8 view:scroll_start

4.21.8.1 Syntax: view:scroll_start()

Scrolls the current view to the start of the signal.

4.21.8.2 See also

view:scroll_end()

4.21.9 view:zoom_all

4.21.9.1 Syntax: view:zoom_all()

Adjusts the zoom factor so that the complete signal is visible in the current view.

4.21.10 view:zoom_in

4.21.10.1 Syntax: view:zoom_in([*position*])

Reduces the zoom factor (in samples per pixel) by 30%, so that more details get visible. If a *position* is given, it tries to show that position centered in the current view, otherwise the center of the view before the zoom change is used for centering. The minimum zoom factor is limited to a minimum of five samples per width of the view.

4.21.10.2 Parameters

position:	a zero based position in samples to center the view (optional)

4.21.10.3 See also

view:zoom_out()

4.21.11 view:zoom_normal

4.21.11.1 Syntax: view:zoom_normal()

Sets the zoom factor to one pixel per sample (factor 1.0) and tries to keep the previous center of the view.

4.21.12 view:zoom_out

4.21.12.1 Syntax: view:zoom_out([*position*])

Increases the zoom factor (in samples per pixel) by 30%, so that less details get visible. If a *position* is given, it tries to show that position centered in the current view, otherwise the center of the view before the zoom change is used for centering. The maximum zoom factor is limited to the number of samples of the complete signal and the width of the view.

4.21.12.2 Parameters

position:	a zero based position is samples to center the view (optional)

4.21.12.3 See also

view:zoom_in()

4.21.13 view:zoom_selection

4.21.13.1 Syntax: view:zoom_selection()

Adjusts the view (zoom factor and start of visible area) so that it matches the current selection. This command does nothing if the selection is empty.

4.22 w

4.22.1 window:activate

4.22.1.1 Syntax: window:activate(*title*])

Activates a sub-window, identified by its window *title*. If the sub-window is minimized it will be restored. Only available if in MDI and Tab mode. This command is internally used by the **Window** menu.

4.22.1.2 Parameters

title:	the title of the sub-window that should be activated

4.22.2 window:cascade

4.22.2.1 Syntax: window:cascade()

Cascades all sub-windows when in MDI mode. All sub-windows that are currently minimized stay minimized, they will not be restored.

4.22.3 window:click

4.22.3.1 Syntax: window:click(*class*, *x*, *y*)

Sends a mouse click event to window, identified by its *class* name. The event will only be sent to the first window that has the given class name, therefore you should make sure that you have only one instance of the given window when this command gets executed.

4.22.3.2 Parameters

class:	name of the window class
x:	x position, relative to the left border of the window (in pixels)
y:	y position, relative to the top border of the window (in pixels)

4.22.4 window:close

4.22.4.1 Syntax: window:close(*class*)

Closes a window, identified by its *class* name. Only the first window that has the given class name will be closed, therefore you should make sure that you have only one instance of the given window when this command gets executed.

4.22.4.2 Parameters

class:	name of the window class

4.22.5 window:minimize

4.22.5.1 Syntax: window:minimize

Minimizes the currently active sub-window when in MDI mode or the current toplevel window when in SDI or Tab mode.

4.22.6 window:mousemove

4.22.6.1 Syntax: window:resize(*class*, *x*, *y*)

Sends a mouse move event to window, identified by its *class* name. The event will only be sent to the first window that has the given class name, therefore you should make sure that you have only one instance of the given window when this command gets executed.

4.22.6.2 Parameters

class:	name of the window class
x:	x position, relative to the left border of the window (in pixels)
y:	y position, relative to the top border of the window (in pixels)

4.22.7 window:next_sub

4.22.7.1 Syntax: window:next_sub()

Activates the *next* sub-window when in MDI or Tab mode. If the next sub-window is minimized it will be restored.

4.22.8 window:prev_sub

4.22.8.1 Syntax: window:prev_sub()

Activates the *previous* sub-window when in MDI or Tab mode. If the previous sub-window is minimized it will be restored.

4.22.9 window:resize

4.22.9.1 Syntax: window:resize(*class*, *width*, *height*)

Changes the size of a window, identified by its *class* name to a new *width* and *height*. The change will only be applied to the first window that has the given class name, therefore you should make sure that you have only one instance of the given window when this command gets executed.

4.22.9.2 Parameters

class:	name of the window class
width:	new width of the window (in pixels)
height:	new height of the window (in pixels)

4.22.10 window:screenshot

4.22.10.1 Syntax: window:screenshot(*class*, *filename*)

Takes a screenshot of a window, identified by its *class* and saves it to a file. The screenshot will be taken from the first window that has the given class name, therefore you should make sure that you have only one instance of the given window when this command gets executed. Currently the format of the file is hardcoded and has to be `*.png`.

4.22.10.2 Parameters

class:	name of the window class
filename:	name of the file to save the screenshot, must have the extension `*.png`

4.22.11 window:sendkey

4.22.11.1 Syntax: window:sendkey(*class*, *key code*)

Sends a key press and release event to a window, identified by its *class* name. The key will only be sent to the first window that has the given class name, therefore you should make sure that you have only one instance of the given window when this command gets executed.

4.22.11.2 Parameters

class:	name of the window class
key code:	the key code that should be sent, using the same syntax as used for setting up menus

4.22.11.3 See also

Description of the parameter *hotkey* of the **menu**() command.

4.22.12 window:tile

4.22.12.1 Syntax: window:tile()

Tiles all sub-windows when in MDI mode, using some scheme of Plasma. All sub-windows that are currently minimized stay minimized, they will not be restored.

4.22.13 window:tile_vertical

4.22.13.1 Syntax: window:tile_vertical()

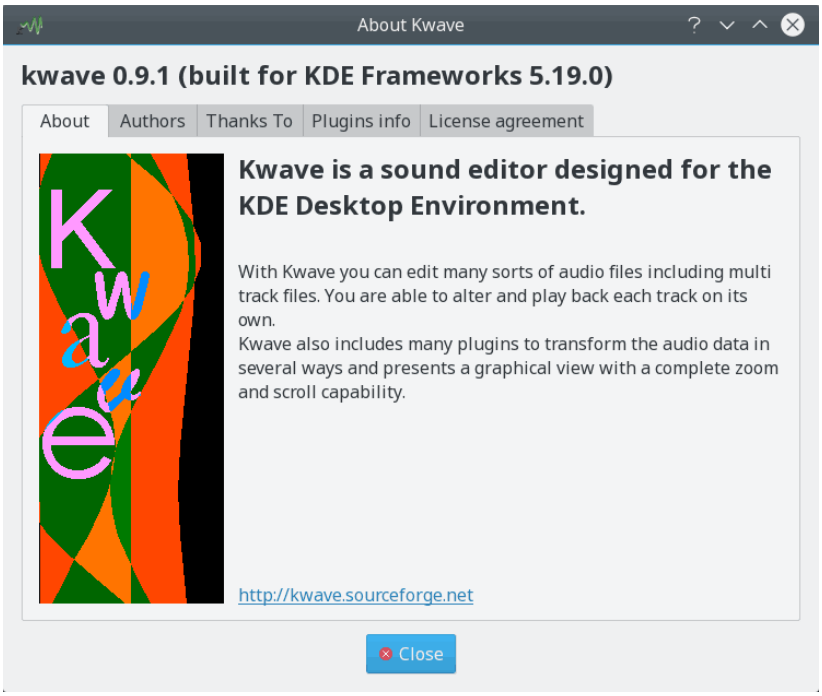
Tiles all sub-windows vertically when in MDI mode. All windows that are currently minimized stay minimized, they will not be restored.

Chapter 5

Plugins

5.1 Plugin Reference

5.2 about (About Kwave)



Internal Name:

`about`

Plugin Type:

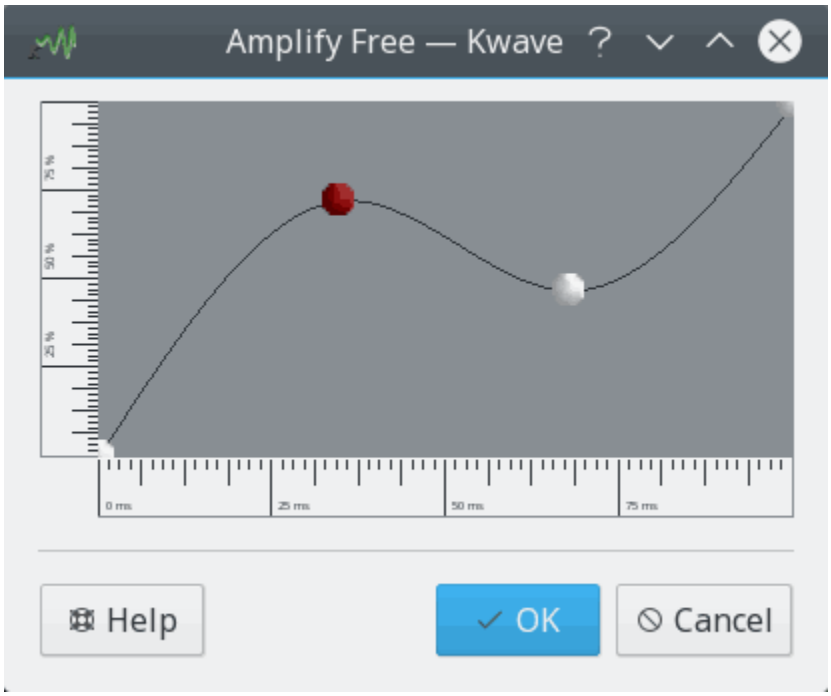
GUI

Description:

Shows a window with multiple tabs, including the following information:

- general information about the program
- authors, contributors and copyright holders
- all found plugins including their versions and authors
- information about the translation team
- copyright and licensing information

5.3 amplifyfree (Amplify Free)



Internal Name:

`amplifyfree`

Plugin Type:

effect

Description:

Amplifies the current selection with a curve that consists of a set of coordinates and an interpolation method. The coordinates on the time axis as well as on the amplitude axis must be normed between 0.0 and 1.0.

Parameters

`operation`

Internal name, for undo/redo handling. Possible values are:

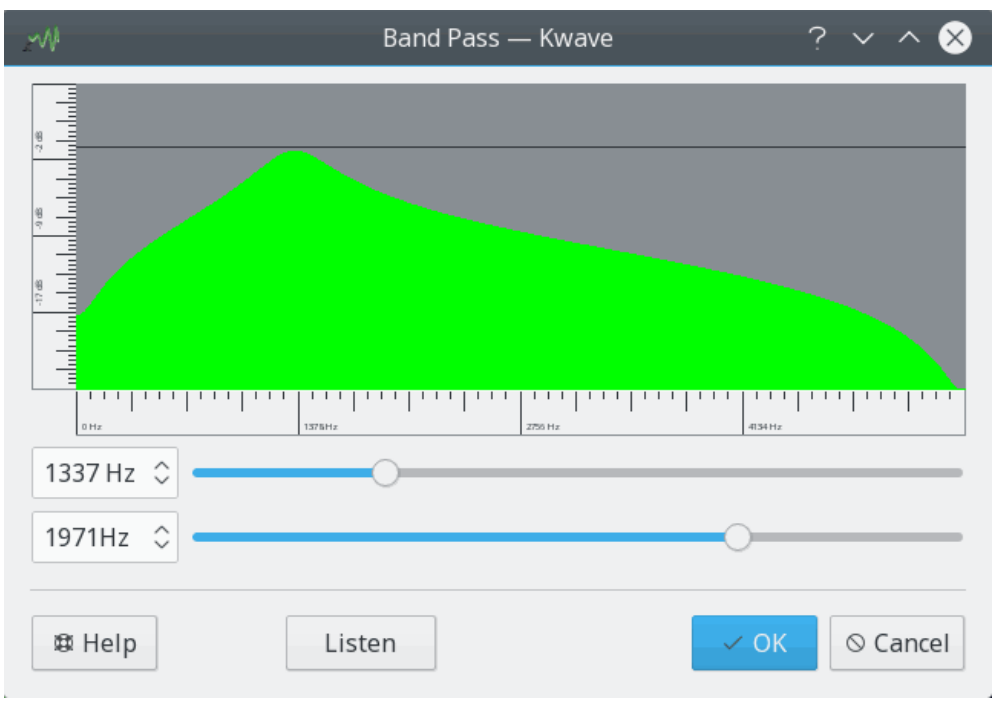
keyword	description
fade in	fade in, curve from 0.0/0.0 to 1.0/1.0
fade out	fade out, curve from 0.0/1.0 to 1.0/0.0
fade intro	fade intro, one second pause, then fade in
fade outro	fade outro, first fade out, then one second pause
amplify free	user defined curve

interpolation
Interpolation type, possible values are:

keyword	description
linear	Linear
spline	Spline
n-polynom	Polynom, nth Degree
3-polynom	Polynom, 3rd Degree
5-polynom	Polynom, 5th Degree
7-polynom	Polynom, 7th Degree
sample_hold	Sample and Hold

curve
A comma separated list of pairs of coordinates, normed between 0.0 and 1.0, must be sorted by time axis (ascending), should start at time 0.0 and end with time 1.0.

5.4 band_pass (Band Pass Filter)



Internal Name:
`band_pass`

Plugin Type:
effect

Description:
Applies a simple band pass filter to the current selection. A band pass lets a certain range of frequencies around a *center frequency* pass and filters out frequencies that are below or above the center frequency by more than half of the *bandwidth* of the filter.

The filter has grade two and is implemented as described in the book *"An introduction to digital filter theory"* by Julius O. Smith and in Moore's book, where the normalized version from Moore's book is used.

Parameters:

frequency
Center frequency of the filter in Hz, must be below half of the sample rate of the file.

bandwidth
Bandwidth of the filter in Hz.

5.5 codec_ascii (ASCII Codec)

Internal Name:
`codec_ascii`

Plugin Type:
codec

Supported File Types:

Description:	ASCII encoded audio
File Extensions:	`*.ascii`
Mime Types:	`audio/x-audio-ascii`

Supported Meta Data:
(all known file info items, see section)

5.6 codec_audiofile (Audiofile Codec)

Internal Name:
`codec_audiofile`

Plugin Type:
codec [import only]

Supported File Types:

Description:	Amiga IFF/8SVX Sound File Format
File Extensions:	`*.8svx`
Mime Types:	`audio/x-8svx`

Description:	NeXT, Sun Audio
File Extensions:	`*.au`, `*.snd`
Mime Types:	`audio/basic`

Description:	Compressed Audio Interchange Format
File Extensions:	`*.aifc`
Mime Types:	`audio/x-aifc`

Description:	Audio Interchange Format
File Extensions:	`*.aif`, `*.aiff`
Mime Types:	`audio/x-aiff`

Description:	Audio Visual Research File Format
File Extensions:	`*.avr`
Mime Types:	`audio/x-avr`

Description:	Core Audio File Format
File Extensions:	`*.caf`
Mime Types:	`audio/x-caf`

Description:	Berkeley, IRCAM, Carl Sound Format
File Extensions:	`*.sf`
Mime Types:	`audio/x-ircam`

Description:	NIST SPHERE Audio File Format
File Extensions:	`*.nist`
Mime Types:	`audio/x-nist`

Description:	Sample Vision Format
File Extensions:	`*.smp`
Mime Types:	`audio/x-smp`

Description:	Creative Voice
File Extensions:	`*.voc`
Mime Types:	`audio/x-voc`

Supported Meta Data:

(none)

5.7 codec_flac (FLAC Codec)

Internal Name:

`codec_flac`

Plugin Type:

codec

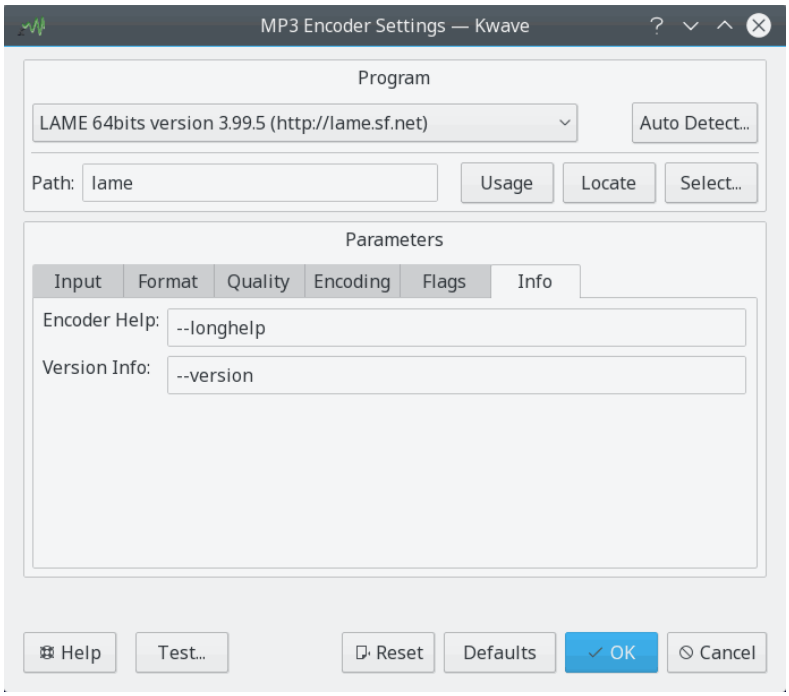
Supported File Types:

Description:	FLAC audio (Free Lossles Audio Codec)
File Extensions:	`*.flac`
Mime Types:	`audio/x-flac`

Supported Meta Data:

Date, Name, Version, Album, Track, Author, Performer, Copyright, License, Organization, Subject, Genre, Source, Contact, ISRC, Software, Engineer, Base Quality

5.8 codec_mp3 (MP3 Codec)



Internal Name:

`codec_mp3`

Plugin Type:

codec

Supported File Types:

Description:	MPEG layer III audio
File Extensions:	`*.mp3`
Mime Types:	`audio/x-mp3`, `audio/mpegs`

Description:	MPEG layer II audio
File Extensions:	`*.mp2`
Mime Types:	`audio/x-mp2`, `audio/mpeg`

Description:	MPEG layer I audio
File Extensions:	`*.mp1`, `*.mpg`, `*.mpga`
Mime Types:	`audio/x-mpga`, `audio/mpeg`

Supported Meta Data:

Album, Annotation, Author, CD, CDS, Comments, Commissioned, Contact, Copyright, Date, Genre, ISRC, Length, License, Medium, Name, Organization, Performer, Software, Technician, Track, Tracks, Version

5.9 codec_ogg (Ogg Codec)

Internal Name:

`codec_ogg`

Plugin Type:

codec

Supported File Types:

Description:	Ogg Opus audio
File Extensions:	`*.opus`
Mime Types:	`audio/ogg`, `application/ogg`, `audio/opus`

Description:	Ogg Vorbis audio
File Extensions:	`*.ogg`
Mime Types:	`audio/ogg`, `audio/x-ogg`, `application/x-ogg`, `audio/x-vorbis+ogg`

Supported Meta Data:

Album, Author, Contact, Copyright, Date, Engineer, Genre, ISRC, License, Name, Organization, Performer, Software, Source, Subject, Track, Base Quality, Version,

5.10 codec_wav (WAV Codec)

Internal Name:

`codec_wav`

Plugin Type:

codec

Supported File Types:

Description:	WAV audio
File Extensions:	`*.wav`
Mime Types:	`audio/x-wav`, `audio/vnd.wave`, `audio/wav`

Supported Meta Data:

Album, Annotation, Archival location, Author, CD, Comments, Commissioned, Contact, Copyright, Date, Engineer, Genre, ISRC, Keywords, License, Medium, Name, Organization, Performer, Product, Software, Source, Source form, Subject, Technician, Track, Version,

5.11 debug (Debug Functions)

Internal Name:

`debug`

Plugin Type:

function

Description:

Provides various internal commands useful for debugging and scripting Kwave. These functions are only available through the main menu if Kwave has been compiled in debug mode (built with the option CMAKE_WITH_DEBUG).

Commands:

`, ,,,`

5.12 export_k3b (Export to K3b Project)

Screenshot of the K3b Export Plugin

Internal Name:

`export_k3b`

Plugin Type:

function

Description:

Saves all sections between markers into a separate file and creates a K3b project file. After having successfully written all files it is possible to start K3b and burn the result to an audio CD. This is useful for splitting a file with a recording that consists of several parts, which are separated by labels, and then burn it to an audio CD with multiple tracks, including CD text meta data which is extracted from the descriptions of the labels.

(This plugin is internally using the plugin.)

Parameters:

filename

The name of the K3b project file, will be used as base name for the exported file names.

pattern

A pattern that will be used for detecting title and artist from the label at the start of a section. It supports the following wildcards which will be replaced by the corresponding content when creating the CD text meta data:

wildcard	description
[%artist]	Will be replaced with the artist that performed the corresponding block or alternatively the author.
[%title]	Will be replaced with the title of the block, which is taken from the descriptive text of the label at the *start* of the block. If that text is empty it will fall back to the title of the file (see file information item "Name"). If this also does not exist, it will fall back to the base file name as described above.

Example: '**`[%title] ([%artist])`**' will detect author '**`Beethoven`**' and title '**`Symphony No. 5`**' from the string '**`Symphony No. 5 (Beethoven)`**'.

selection only

value	description
0	Save all sections of the whole file.
1	Save only the sections that are within the current selection. If nothing is selected, the whole file will be saved.

export location

Determines where the blocks should be saved.

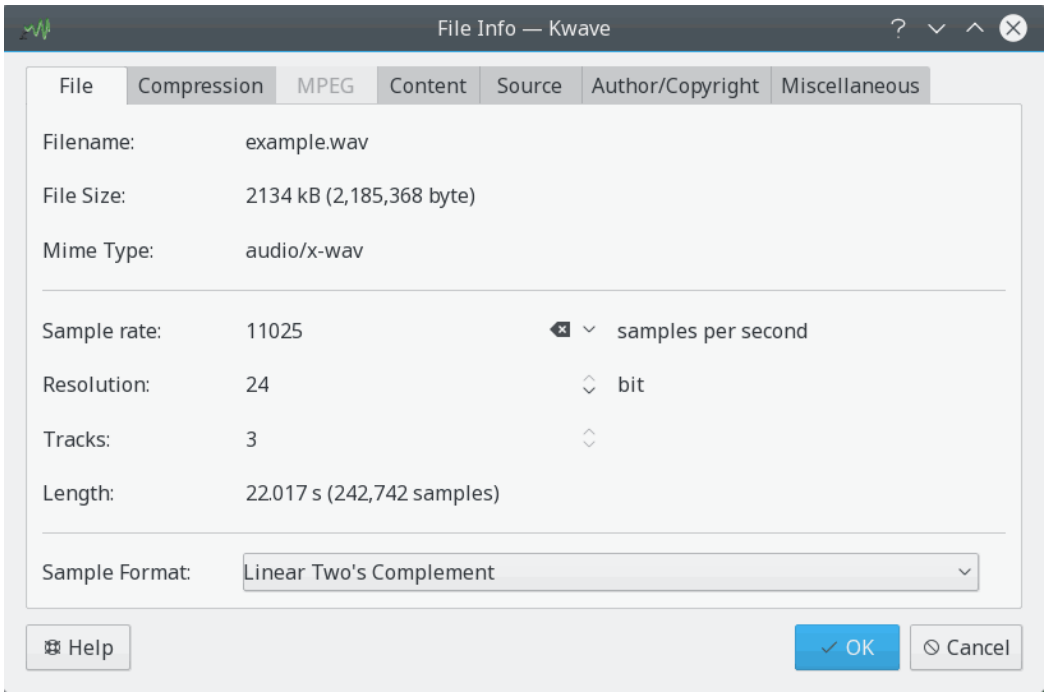
value	description
0	Save to the same directory as the K3b project file.
1	Save into a subdirectory of the directory of the K3b project file, using the K3b project file name as base and appending '**`.dir`**'.

overwrite policy

Determines where the numbering should start.

value	description
0	Always start with index 1, with the risk of overwriting existing files.
1	Continue after the index of the highest index that already exists, this avoids overwriting existing files.

5.13 fileinfo (File Info)



Internal Name:

`fileinfo`

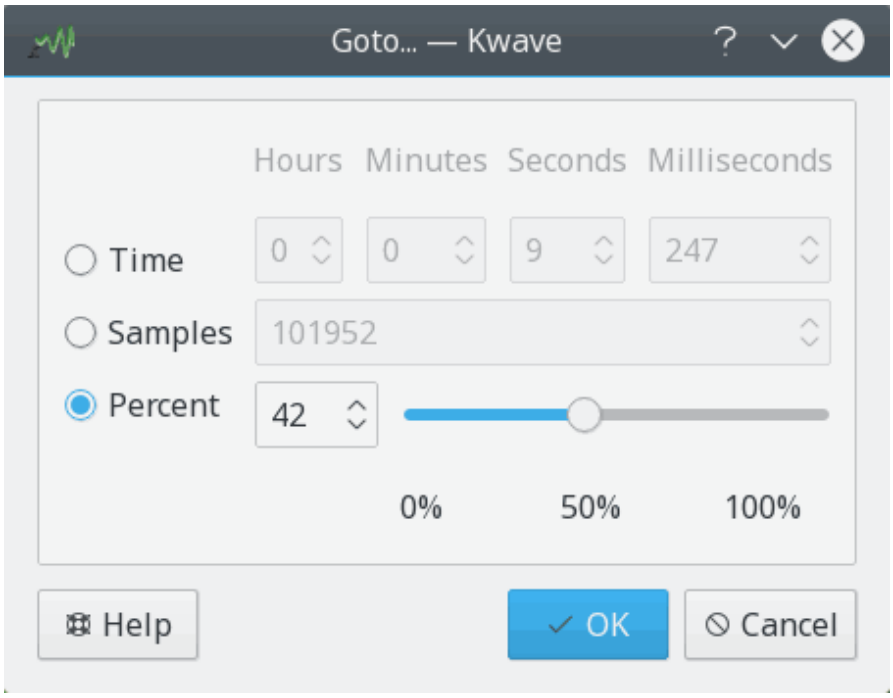
Plugin Type:

GUI

Description:

Provides a dialog window to view and change parameters and meta data of the currently opened file. See section in this manual.

5.14 goto (Goto Position)



Internal Name:

`goto`

Plugin Type:

function

Description:

Shows a dialog with the possibility to set the current position of the selection to a new value, either by a time in milliseconds, by a position in samples or by percentage of the length of the current file.

Commands:

Parameters:

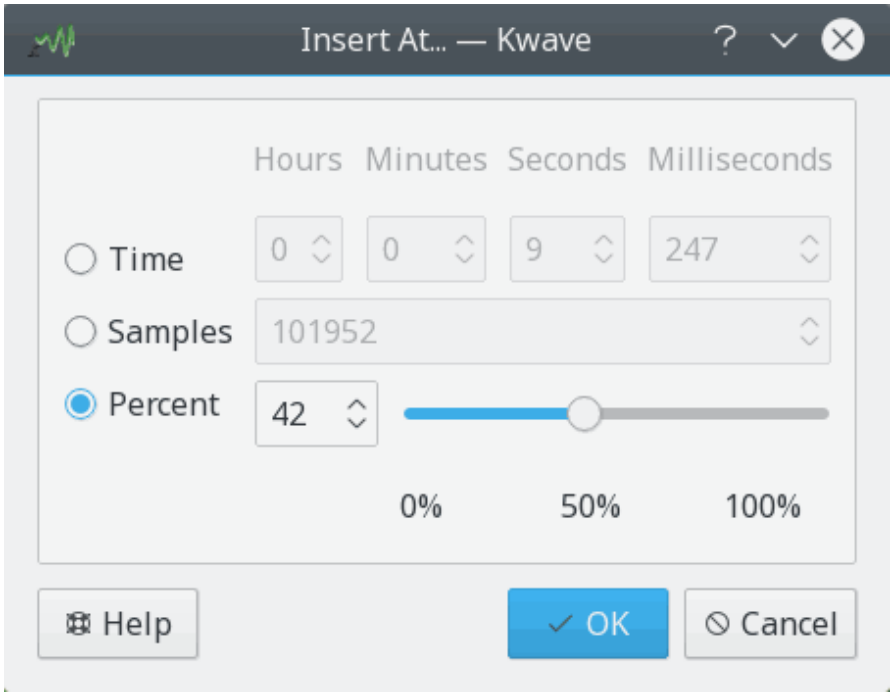
mode

value	description
0	position is given in milliseconds
1	position is given in samples
2	position is given in percentage of the file length

position

position to go to, in milliseconds, samples or percentage of the length of the file, depending on the parameter `mode`.

5.15 insert_at (Insert At)



Internal Name:

`insert_at`

Plugin Type:

function

Description:

Similar to the plugin, but shows a dialog with the possibility to insert the current content of the clipboard at a given position, either by a time in milliseconds, by a position in samples or by percentage of the length of the current file.

Commands:

Parameters:

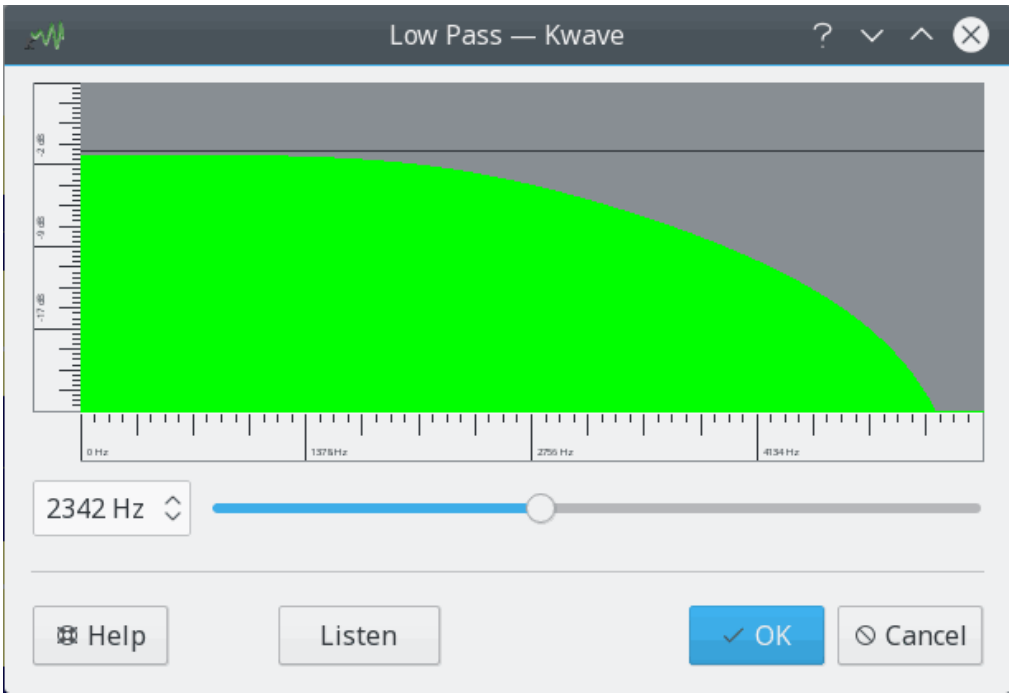
mode

value	description
0	position is given in milliseconds
1	position is given in samples
2	position is given in percentage of the file length

position

position where to insert the clipboard data, in milliseconds, samples or percentage of the length of the file, depending on the parameter `mode`.

5.16 lowpass (Low Pass Filter)



Internal Name:

`lowpass`

Plugin Type:

effect

Description:

Applies a simple low pass filter to the current selection. A low pass filter lets frequencies below a *border frequency* pass and filters out frequencies that are above the border frequency.

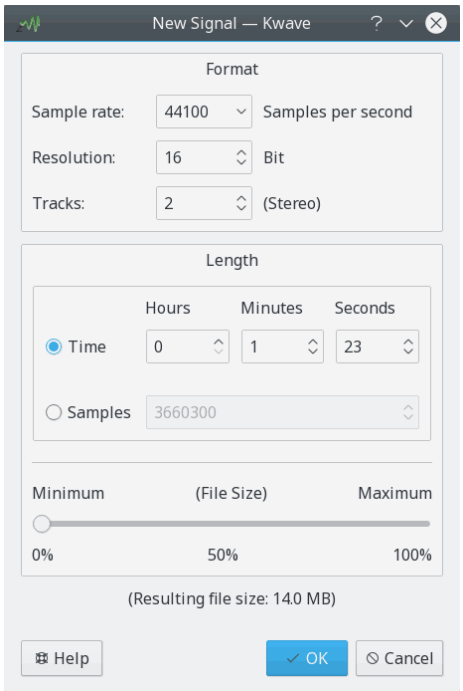
The filter has grade two and is implemented as described in the book *"The manifold joys of conformal mapping, applications to digital filtering in the studio* by James A. Moorer (JAES, Vol. 31, No. 11, 1983 November).

Parameters:

frequency

The border frequency of the low pass filter in Hz.

5.17 newsignal (New Signal)



Internal Name:

`newsignal`

Plugin Type:

function

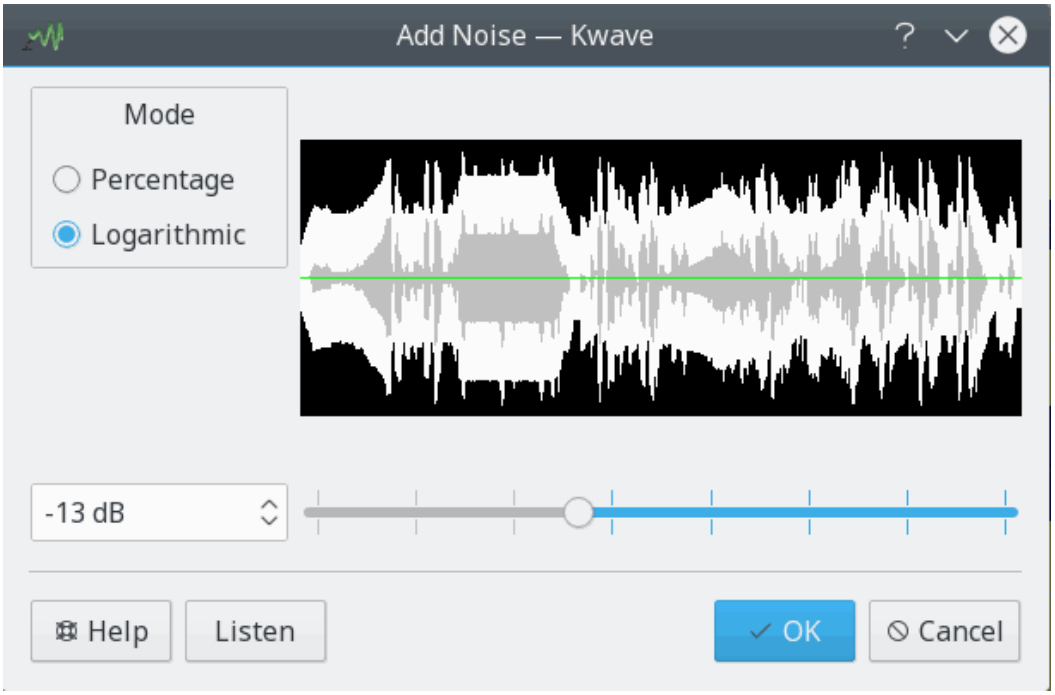
Description:

Provides a dialog to create a new file. Please refer to the section in this manual for more information.

Commands:

,

5.18 noise (Noise Generator)



Internal Name:

`noise`

Plugin Type:

effect

Description:

Adds some amount of white noise to the current selection. The amount of noise can be selected between zero (no noise, original remains unchanged) and one (original will be replaced by 100% noise).

Parameters:

level

Noise level, always has to be a floating point number above zero and below or equal to one.

mode

value	description
0	Enter the noise value as percentage of the amplitude, from 0 to 100.
1	Enter the noise in decibel, from -21 dB to 0 dB.

5.19 normalize (Normalizer)

Internal Name:

`normalize`

Plugin Type:

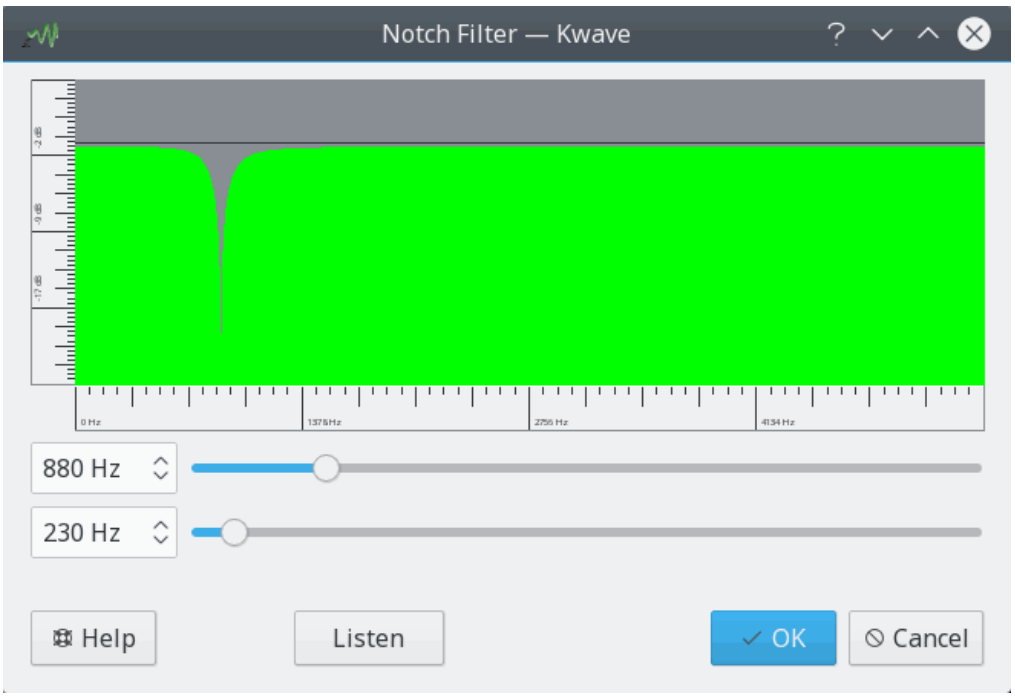
effect

Description:

Normalizes the volume level of the current selection. Use this if the volume of your signal is too low or too high.

The algorithm is taken from the *normalize* project, and was originally written by Chris Vaill .

5.20 notch_filter (Notch Filter)



Internal Name:

`notch_filter`

Plugin Type:

effect

Description:

Applies a notch filter to the current selection. A notch filter removes a small range of frequencies around a *center frequency* and lets all other frequencies below and above the center frequency by more than half of the *bandwidth* pass.

Use this to filter out single distortion frequencies.

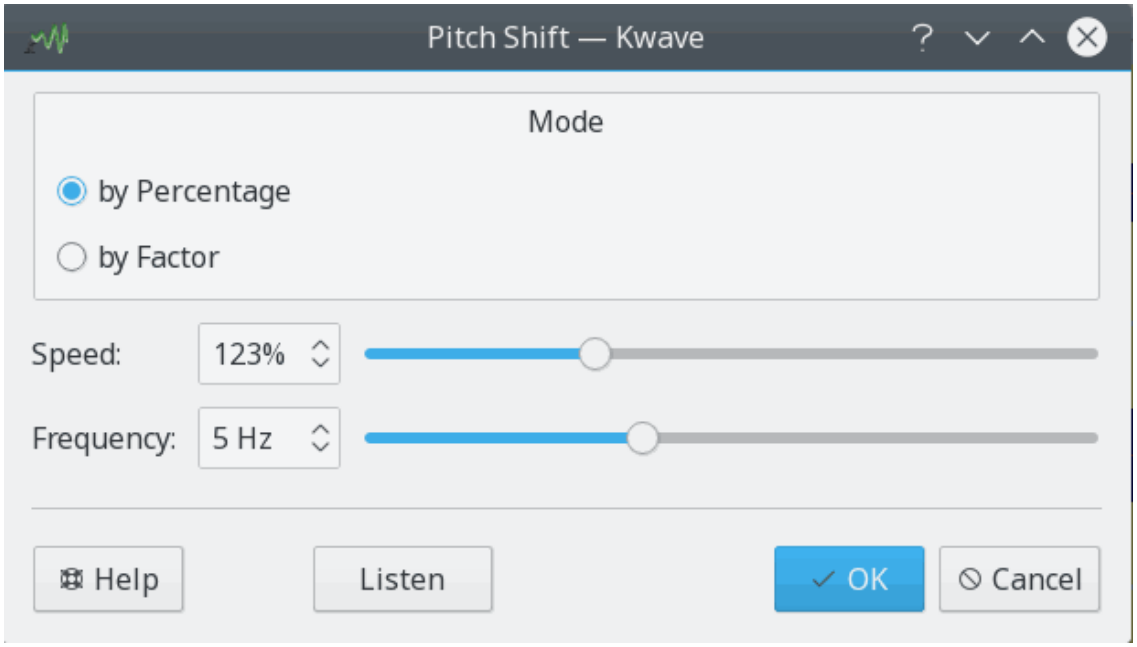
The filter has grade two and is based on the implementation of Juhana Sadeharju .

Parameters:

frequency
: Center frequency of the filter in Hz, must be below half of the sample rate of the file.

bandwidth
: Bandwidth of the filter in Hz.

5.21 pitch_shift (Pitch Shift)



Internal Name:

`pitch_shift`

Plugin Type:

effect

Description:

The pitch shift effect modifies the signal by changing the speed of the content, but with keeping the original length. You can select the relative speed either by factor from 1/10 to x5, or as a percentage from 1% to 400% of the original speed.

A speed factor below 1.0 pitches the signal down (lower voice, makes voices sound older), factor 1.0 does no change and a factor above 1.0 pitches the signal up (higher voice, Mickey Mouse effect).

The implementation is based on the work of Jeff Tranter and Stefan Westerfeld

Parameters:

speed

Factor for changing the speed, has to be a floating point number between 0.001 and 4.0.

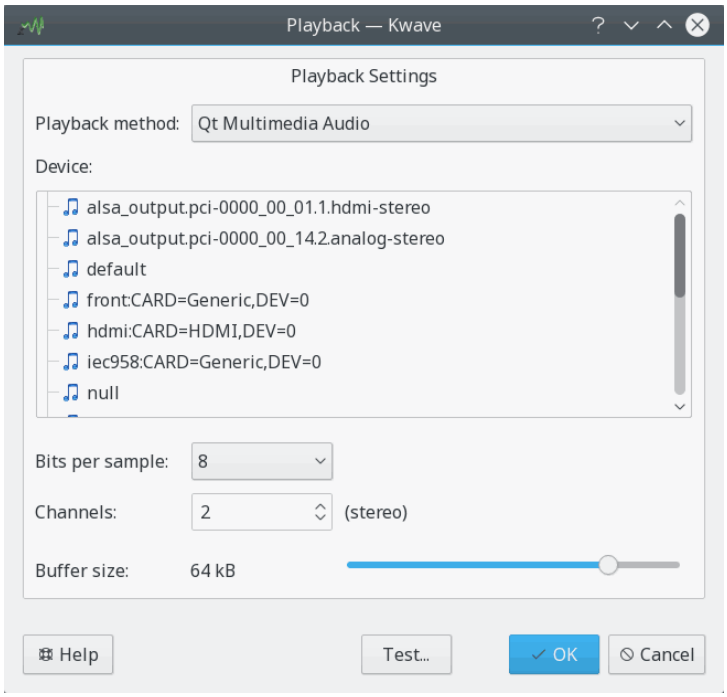
frequency

Frequency internally used by the filter in Hz, has to be between 2.0 and 10.0.

mode

value	description
0	Enter the speed value as factor from 1/10 to x5.
1	Enter the speed value as percentage from 1 to 400.

5.22 playback (Playback)



Internal Name:

`playback`

Plugin Type:

function

Description:

Provides a dialog to set up the playback parameters. Please refer to the section in this manual for more information.

Parameters:

playback method

The method used for playback, see `PlayBackParam.h`.

playback device

A string that determines the playback device or channel. The meaning depends on the playback method.

channels

The number of channels to use for playback, currently supports only 1 (mono) or 2 (stereo).

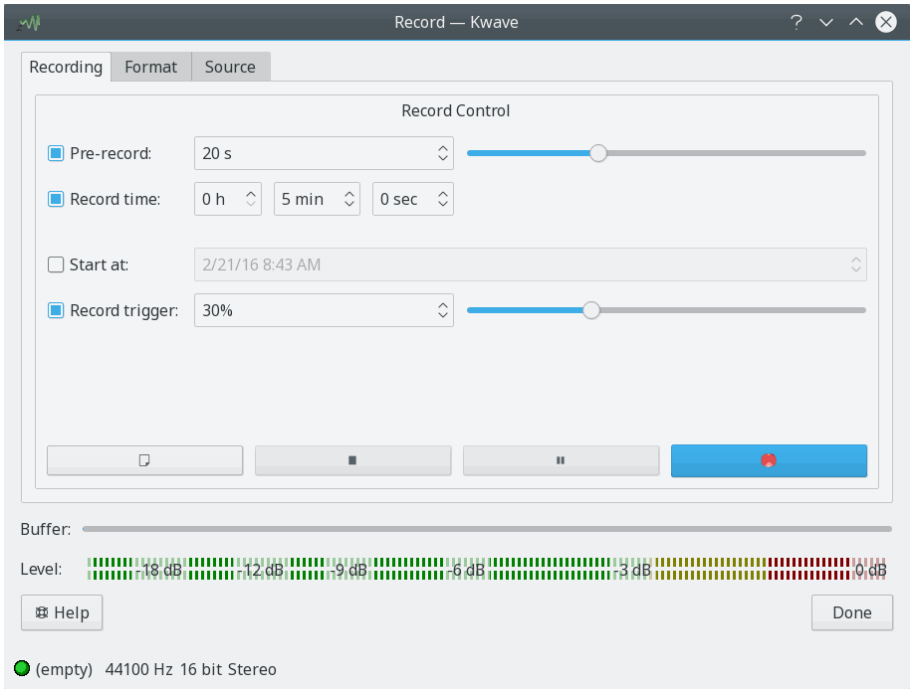
bits per sample

The number of bits per sample for playback, should be 8, 16, 24 or 32. Depends on the playback method and the playback device.

buffer size

Determines the size of the playback buffer, used as exponent for calculating the real buffer size as 2^n, e.g. setting this to 16 gives a buffer size of 2^16 = 64 kB.

5.23 record (Record)



Internal Name:

`record`

Plugin Type:

function

Description:

Provides a dialog to set up the record parameters and to do a recording. Please refer to the section in this manual for more information.

Parameters:

recording method
: The method used for recording, see `RecordParams.h`.

pre recording enabled
: Enable/disable pre recording (1 if enabled, 0 if disabled).

pre recording time
: Number of seconds for pre recording.

limit recording time
: Enable/disable limiting of recording time (1 if limited, 0 if not limited).

recording time
: Duration of the recording in seconds.

use starting time
: Enable/disable starting time (1 if used, 0 if not used).

starting time
: Date/time to start the recording, in ISO format.

use trigger level
: Enable/disable trigger level (1 if used, 0 if not used).

trigger level
Trigger level in percent.

recording device
A string that determines the recording device.

channels
The number of channels to use for recording.

sample rate
Sample rate in samples per second.

compression
Compression to use for storing the samples.

sample format
Sample format to use for storing the samples, see section about sample formats.

bits per sample
The number of bits per sample for recording, should be 8, 16, 24 or 32.

buffer count
Determines the number buffers used for recording.

buffer size
Determines the size of the recording buffer, used as exponent for calculating the real buffer size as 2^n, e.g. setting this to 16 gives a buffer size of 2^16 = 64 kB.

Alternative Parameters:

record plugin direct mode
Can be used as a single parameter for setting up the plugin. The following values are possible:

value	description
format	Open the recording dialog and select the **Format** tab.
source	Open the recording dialog and select the **Source** tab.
start_now	Open the recording dialog and directly start recording.

5.24 reverse (Reverse)

Internal Name:
`reverse`

Plugin Type:
effect

Description:
This simple effect reverses the content of the current selection.

5.25 samplerate (Sample Rate Conversion)

Internal Name:

`samplerate`

Plugin Type:

effect

Description:

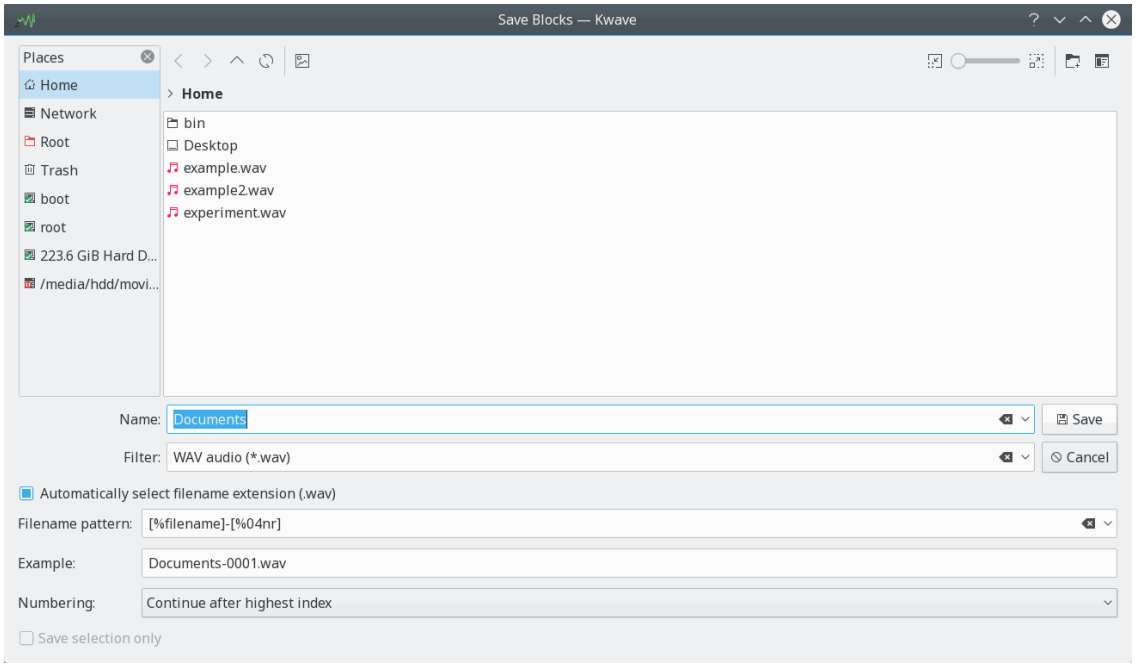
Changes the sample rate of the current selection or the whole signal.

Parameters:

new rate
The new sample rate in samples per second (floating point value).

***mode* (optional)**
If this parameter is used and set to the value "`all`", then this effect will be applied to the whole signal. Otherwise it will be applied to the current selection only.

5.26 saveblocks (Save Blocks)



Internal Name:

`saveblocks`

Plugin Type:

function

Description:

Saves all sections between markers, each into a separate file. Each file is given a name that can be customized by using a pattern that can contain the original file name, an index and the number of sections.
It is also allowed that the file name pattern contains forward slashes as path separators, which allows saving the sections into different sub directories. Please note that all whitespace characters around such path separators are silently removed, to avoid creation of directory names which begin or end with a whitespace.

Parameters:

name

The name of the original file, will be used as base name for the file names.

pattern

A pattern that will be used for creating the names of the files. It can contain the following wildcards which will be replaced by the corresponding content when creating the final file name:

wildcard	description
[%nr]	Will be replaced with the current index of the file to save.
[%count]	Will be replaced with the number of sections that will be saved.
[%total]	Will be replaced with the index of the last file to save.
[%filename]	Will be replaced with the base file name, without path and without extension.
[%fileinfo{*keyword*}]	Will be replaced with the content of a file information identified by `keyword`. See section for a list of all available keywords.
[%title]	Will be replaced with the title of the block, which is taken from the descriptive text of the label at the *start* of the block. If that text is empty it will fall back to the title of the file (see file information item "Name"). If this also does not exist, it will fall back to the base file name as described above.

All numeric wildcards can also contain a numerical argument after the "%" and the identifier, to force a certain number of digits. If the number is preceded by a 0 then the result will contain leading zeroes, otherwise it will contain leading spaces.

Example: **[%04nr]** produces a number between 0001 and 9999.

numbering mode

Determines where the numbering should start.

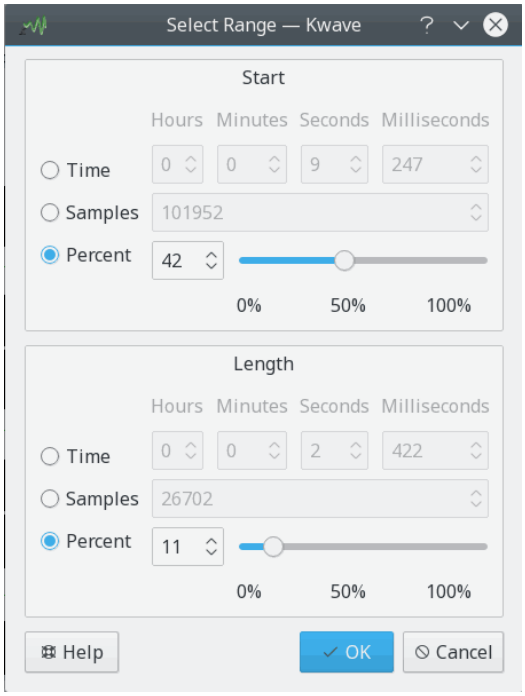
value	description
0	Continue after the index of the highest index that already exists, this avoids overwriting existing files.
1	Always start with index 1, with the risk of overwriting existing files.

selection only

value	description
0	Save all sections of the whole file.

1	Save only the sections that are within the current selection. If nothing is selected, the whole file will be saved.

5.27 selectrange (Select Range)



Internal Name:

`selectrange`

Plugin Type:

function

Description:

Shows a dialog to select a range of samples. The start and the length of the selection can be set either by a time in milliseconds, in samples, or as a percentage of the total length of the file.

Parameters:

start mode

Determines the units in which the *start* of the selection will be given.

value	description
0	milliseconds
1	samples
2	percentage of the length of the file

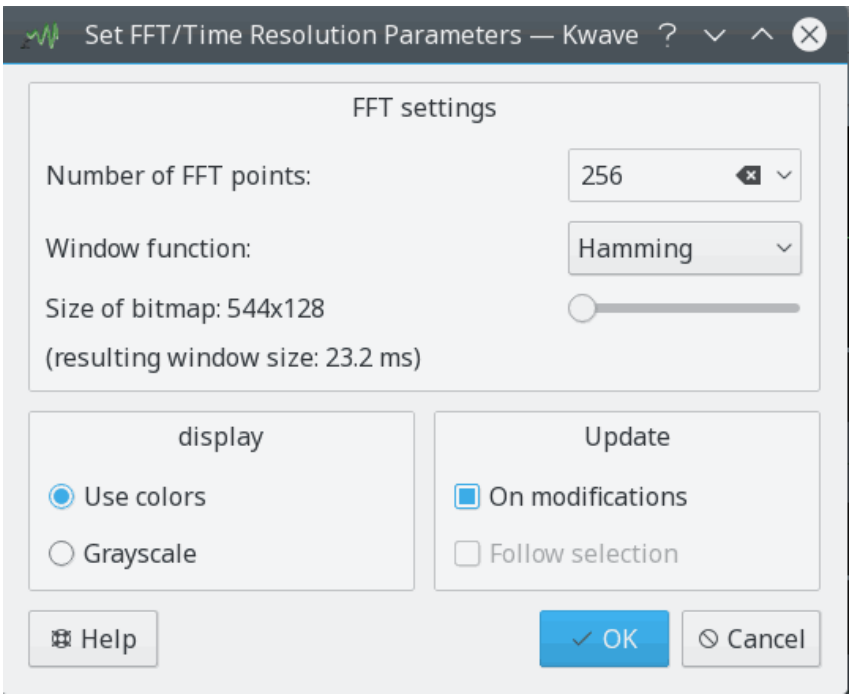
range mode

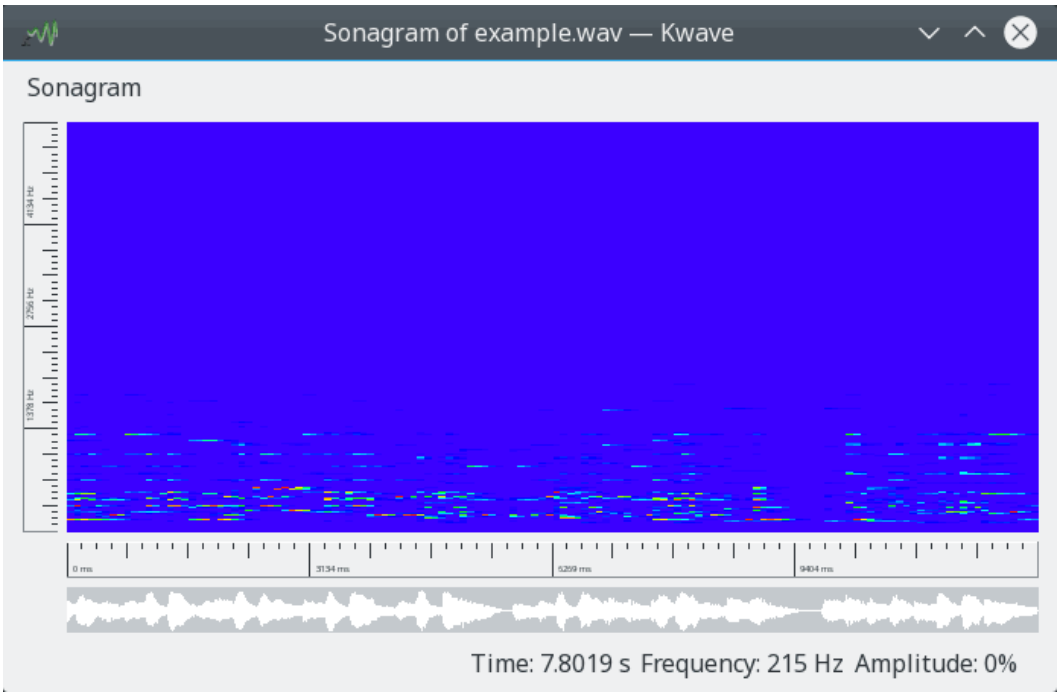
Determines the units in which the *length* of the selection will be given. See the description if the parameter `start mode` for a list of possible values.

start
The start of the selection, in milliseconds, samples or percentage of the length of the file, depending on the parameter `range mode`.

length
The length of the selection, in milliseconds, samples or percentage of the length of the file, depending on the parameter `range mode`.

5.28 sonagram (Sonagram)





Internal Name:

`sonagram`

Plugin Type:

function

Description:

Evaluates the current selection by generating a *sonagram*. A sonagram is an evaluation of a signal over time (x axis), frequency (y axis) and intensity (color).

Parameters:

FFT points

Number of points of the FFT, a whole number between 4 and 32767 which determines the frequency resolution.

window function

The window function used for the FFT calculation, supported values are:

value	description
none	no window function
hamming	Hamming window
hanning	Hanning window
blackman	Blackman window
triangular	Triangular window

use colors

If set to non-zero, use colors for intensity, if set to zero use grayscales.

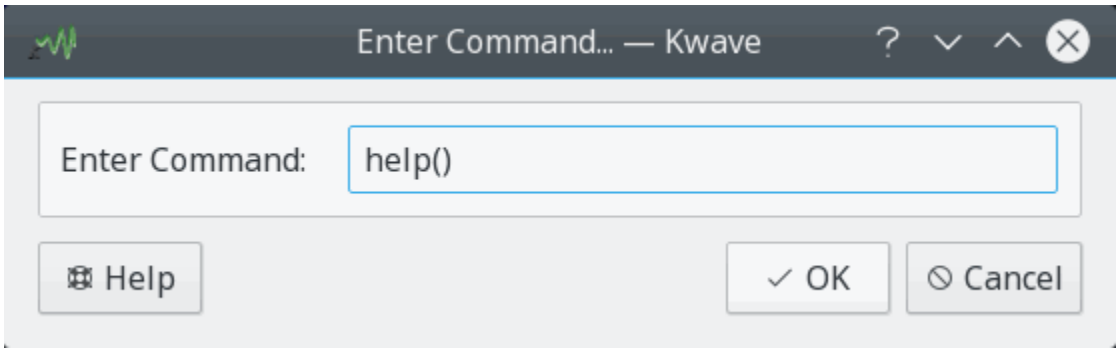
track changes

If set to non-zero, the sonagram will be updated when the area that was evaluated has changed. If set to zero it will never be updated.

follow selection

Not yet implemented, use zero for this parameter.

5.29 stringenter (Enter Command)



Internal Name:

`stringenter`

Plugin Type:

function

Description:

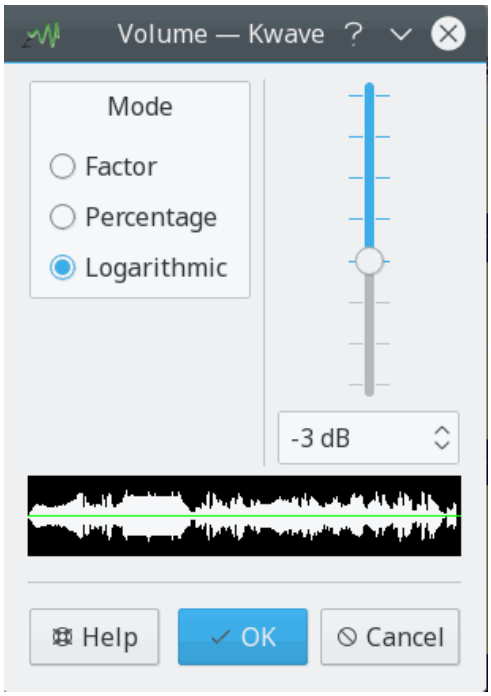
A small dialog window that allows to enter a Kwave text command. Please refer to the chapter in this manual.

Parameters:

***preset* (optional)**

A text that is shown in the edit field when entering the dialog. This parameter is optional, if omitted the edit field of the dialog will be empty at start.

5.30 volume (Volume)



Internal Name:

`volume`

Plugin Type:

effect

Description:

With this plugin you can change the volume of the current selection by a constant factor. The corresponding dialog allows to enter this factor as a *numeric factor* given as a floating point value between 0.10 and 10.0, a *percentage* between 1 and 1000, or in *decibel* between -21 and +21.

Use a factor above 1.0 (or percentage above 100 or more than 0 dB) if the file is too silent, or a factor below 1.0 (percentage below 100 or less than 0 dB) if the file is too loud.

Parameters:

factor

A floating point value with the amplification factor.

mode

value	description
0	factor
1	percentage
1	decibel

5.31 zero (Zero Generator)

Internal Name:

`zero`

Plugin Type:

effect

Description:

This plugin has two modes of operation. If used without parameters it wipes the current selection by overwriting it with silence. When used with two parameters, it inserts some amount of silence at the start of the current selection.

Parameters:

length mode

Determines the units in which the *length* of the inserted silence will be given.

value	description
0	milliseconds
1	samples
2	percentage of the length of the file

length

Length of the silence to insert, in milliseconds, samples or percentage of the length of the file, depending on the parameter `length mode`.

Chapter 6

Questions and Answers

1. *What do I need to compile Kwave?*

   Read in the developer documentation.

2. *Which sound cards does Kwave support?*

   Kwave does not need support for any special sound card. The sound card only has to be supported by your operating system and Kwave uses its interface to the operating system's sound driver through a OSS or ALSA interface.

3. *Why does Kwave consume more memory than it can be expected from the size of the opened file?*

   The reason for this is that Kwave internally stores all samples in 32-bit integers. This was easy to program, made the application faster and a bit more reliable. So if you load an 8-bit file with about one megabyte it will consume about four megabytes. Maybe we will change this somewhere in the future...

4. *Which sound formats does Kwave support?*

   Kwave currently supports .wav files with 8, 16 and 24 bits per sample, with any number of channels (of course including mono and stereo). Additionally it can import all file types that libaudiofile supports and some other formats like Ogg/Vorbis and MP3.

5. *What if I have files with a format not supported by Kwave?*

   If you have to work on a different format, you can convert it into .wav format. A good set of tools for this is in the SoX package, they have also some nice documentation!

6. *I get errors when I want to do playback?*

   Maybe you have chosen a combination of playback rate and sample size that is not supported by your sound driver and / or sound hardware. Try playback with 8 bits per sample and mono first, this should always work. Then try to increase the bits per sample and stereo playback step by step. Note that some playback rates are not at all supported by some sound hardware.

7. *The playback seems to do something but I hear nothing?*

   Maybe you have forgotten to increase the volume of the playback channel. Kwave is not responsible for changing the playback volume.

8. *Some files are played with half-speed?*

   Try choosing a different sound playback device.

9. *The playback sometimes is disturbed and interrupted?*

   You should increase the size of the playback buffer to get a "smoother" playback (this also makes the playback control reacting a bit slower).

10. *The playback does not stop if I immediately press the stop button?*

    The reason for this is that the sound driver already has received some playback data from Kwave at the moment when you press the stop button. Decrease the size of the playback buffer and it should react faster (but makes interruptions more probable).

11. *Is ALSA supported?*

    Yes, since v0.7.4 for playback and recording

12. *What about playback with 18, 20, 24 or 32 bits per sample or more than two channels?*

    This is possible through the ALSA interface, since v0.7.4.

Chapter 7

Credits and License

Kwave

Program copyright from 1998-2000 Martin Wilz martin@wilz.de

Program copyright since 2000 Thomas Eschenbacher thomas.eschenbacher@gmx.de

For a complete list of authors and licenses of all files, please refer to the LICENSES file, which is included in the sources. There also is a file with the name AUTHORS that lists all authors and contributors of Kwave.

Documentation copyright (C) 2020 Thomas Eschenbacher thomas.eschenbacher@gmx.de

This documentation is licensed under the terms of the GNU Free Documentation License.

This program is licensed under the terms of the GNU General Public License.

7.1 Main Authors

- **Martin Wilz martin@wilz.de**
  creator of the project, active development 1998-2000
- **Thomas Eschenbacher thomas.eschenbacher@gmx.de**
  maintainer since 2000, core development

7.2 Major Contributors

- **Aurelien Jarno aurel32@debian.org**
  debian packager, patches
- **Carlos R pureacetone@gmail.com**
  spanish translation
- **David Flogeras dflogera@nbnet.nb.ca**
  Notch Filter plugin
- **Gilles Caulier caulier.gilles@free.fr**
  i18n, french translations, splashscreen, beta tester
- **Pavel Fric pavelfric@seznam.cz**
  Czech translation

- **Ralf Waspe rwaspe@web.de**
  Help/About plugin
- **Sven-Steffen Arndt ssa29@gmx.de**
  homepage, german translation

7.3 Minor contributors, copyright holders and others

- **Aaron Holtzman aholtzma@ess.engr.uvic.ca**
  libkwave/cpu_accel.cpp
- **Bertrand Songis bsongis@gmail.com** *[historic]*
  french translation fixes, substitutes for patented libaudiofile code, debian bug 419124
- **Carsten Lohrke carlo@gentoo.org**
  svn r2163, patch for libaudiofile detection
- **Chris Vaill chrisvaill@gmail**
  code base for the normalize plugin
- **David Faure faure@kde.org**
  cmake/FindAlsa.cmake
- **Diederick de Vries diederick76@gmail.com**
  packaging for Crux Linux
- **Espen Sand espen@kde.org + Mirko Boehm mirko@kde.org**
  K3AboutContainer, base of KwaveAboutContainer
- **Everaldo Coelho contact@everaldo.com**
  the crystal icon theme http://www.everaldo.com/crystal/
- **Jaroslav Kysela**
  parts of plugins/playback/PlayBack-ALSA.cpp
- **Jeff Tranter**
  parts of plugins/pitch_shift/PitchShiftFilter.{h,cpp}
- **Juhana Sadeharju kouhia@nic.funet.fi**
  plugins/band_pass/BandPass.{h,cpp}, plugins/lowpass/LowPassFilter.cpp, plugins/notch_filter/NotchFilter.{h,cpp}
- **Kurt Roeck Q@ping.be**
  svn r1370, fix for debian bug#288781, compilation for amd64
- **Mark Donohoe (KDE) donohoe@kde.org**
  some icons and bitmaps for toolbar and GUI
- **Martin Hinsch vidas@sourceforge.net**
  Matrix class
- **Matthias Kretz kretz@kde.org**
  cmake/FindAlsa.cmake
- **Miguel Freitas**
  parts of libkwave/memcpy.c

- **Richard Laerkaeng, richard@goteborg.utfors.se**
  cmake/FindOggVorbis.cmake
- **Rik Hemsley rik@kde.org**
  level meter
- **Stefan Westerfeld stefan@space.twc.de**
  parts of plugins/pitch_shift/PitchShiftFilter.{h,cpp}
- **Joerg-Christian Boehme joerg@chaosdorf.de**
  plugins/record/Record-PulseAudio.cpp plugins/record/Record-PulseAudio.h
- **Sebastian Trueg trueg@k3b.org, Gustavo Pichorim Boiko gustavo.boiko@kdemail.net, Michal Malek michalm@jabster.pl**
  parts of plugins/export_k3b/K3BExportPlugin.cpp

7.4 Thanks To

- **Martin Kuball makube@user.sourceforge.net**
  beta tester
- **Jorge Luis Arzola arzolacub@gmx.de**
  packaging for SuSE Linux
- **Michael Favreau michel.favreau@free.fr**
  packaging for Arch Linux
- **T.H.F. Klok and Cedric Tefft**
  maintainers of the id3lib library
- **Robert Leslie rob@mars.org**
  author of the mad mp3 decoder library
- **Robert M. Stockmann stock@stokkie.net**
  packaging for Mandrake / X86_64
- **Erik de Castro Lopo erikd@zip.com.au**
  author of the sndfile library
- **Michael Pruett mpruett@sgi.com**
  author of the audiofile library

Appendix A

File Info

Keyword		Description
Album		Name of the album if the source is an album that consist of more medias.
Annotation		Provides general comments about the file or the subject of the file. If the comment is several sentences long, end each sentence with a period. Do not include newline characters!
Archival location		Indicates where the subject of the file is archived.
Author		Identifies the name of the author of the original subject of the file. Example: 'van Beethoven, Ludwig'
Lower Bitrate		Specifies the lower limit in a VBR bitstream.
Bitrate Mode		Bitrate Mode (ABR, VBR, CBR, etc.)
Bitrate		Nominal bitrate of the audio stream in bits per second
Upper Bitrate		Specifies the upper limit in a VBR bitstream.
Bits per Sample		Specifies the number of bits per sample.
CD		Number of the CD, if the source is an album of more CDROMs
CDS		Number of CDs, if the source is an album of more CDROMs
Commissioned		Lists the name of the person or organization that commissioned the subject of the file.

Comments		Provides general comments about the file or the subject of the file. If the comment is several sentences long, end each sentence with a period. Do not include newline characters!
Compression		Sets a mode for compressing the audio data to reduce disk space.
Contact		Contact information for the creators or distributors of the track. This could be a URL, an email address, the physical address of the producing label.
Copyright		Records the copyright information for the file. If there are multiple copyrights, separate them by a semicolon followed by a space. Example: 'Copyright Linux community 2002'
Copyrighted		Indicates whether the file is protected by copyright or not.
Date		Specifies the date the subject of the file was created. Example: '2001-12-24'
Engineer		Shows the name of the engineer who worked on the file. If there are multiple engineers, separate the names by a semicolon and a blank.
Estimated Length		Estimated length of the file in samples
Filename		Name of the opened file
File Size		Size of the file in bytes
Genre		Describes the genre or style of the original work. Examples: 'classic', 'pop'
ISRC		ISRC number for the track; see the ISRC intro page for more information on ISRC numbers. http://isrc.ifpi.org/
Keywords		Provides a list of keywords that refer to the file or subject of the file.
Labels		The list of labels/markers.
Length		Length of the file in samples.

License		License information, e.g., 'All Rights Reserved', 'Any Use Permitted', an URL to a license or the Attribution-ShareAlike 4.0 International ('distributed under the terms of the Attribution-ShareAlike 4.0 International License. See https://creativecommons.-org/licenses/by-sa/4.0/ for details'), etc.
Medium		Describes the original subject of the file, where it was first recorded. Example: 'orchestra'
Mime Type		Mime type of the file format
Emphasis		Audio emphasis mode
Layer		MPEG Layer, I, II or III
Mode Extension		MPEG Mode Extension (only if Joint Stereo)
Version		MPEG Version, 1, 2 or 2.5
Name		Stores the title of the subject of the file. Example: 'Symphony No.6, Op.68 "Pastoral"'
Opus Frame Length		Opus Frame Length in ms (supported values are 2.5, 5, 10, 20, 40, or 60 ms)
Organization		Name of the organization producing the track (i.e. the 'record label')
Original		Indicates whether the file is an original or a copy
Performer		The artist(s) who performed the work. In classical music this would be the conductor, orchestra, soloists. In an audio book it would be the actor who did the reading.
Private		Indicates whether the subject is private
Product		Specifies the name or the title the file was originally intended for. Example: 'Linux audio collection'
Sample Format		Format used for storing the digitized audio samples. Example: '32-bit IEEE floating-point'
Sample Rate		Number of samples per second

Software		Identifies the name of the software package used to create the file. Example: 'Kwave v0.6.4-1'
Source		Identifies the name of the person or organization who supplied the original subject of the file. Example: 'Chaotic Sound Research'
Source form		Identifies the original form of the material that was digitized. Examples: 'Record/Vinyl/90RPM', 'Audio DAT', 'tape/CrO2/60min'
Subject		Describes the subject of the file. Example: 'Bird voices at early morning'
Technician		Identifies the technician who digitized the subject file. Example: 'Torvalds, Linus'
Track		Track of the CD if the source was a CDROM.
Tracks		Number of tracks of the CD if the source was a CDROM.
Channels		Specifies the number of channels of the signal.
Base Quality		Base quality of the compression in VBR mode
Version		May be used to differentiate multiple versions of the same track title in a single collection. (e.g. remix info)

Table A.1: List of File Info Identifiers