Linux Standard Base Trial Use Specification 3.2

Linux Standard Base Trial Use Specification 3.2
LSB Trial Use Specification

Contents

List of Tables

Foreword

This is version 3.2 of the Trial Use Specification. This version is a preliminary version for review only. Conclusion of work on this version will result in version 3.2 of the Trial Use Specification.

This specification describes components which have Trial Use Specification status, and as such there is no formal compliance process for this specification. Implementations may claim to provide these components in a manner that agrees with this specification, but such a claim is not part of a conformance statement for the LSB version in which this module appears.

Applications may not assume that the components of this specification are present or operate as described in this specification on any given implementation.

Introduction

The Trial Use Specification describes components which may or may not be present on an otherwise conforming system. The purpose is to indicate that these components are on a Standards Track, that is, they are intended to become part of the LSB Specification in a future edition.

This document should be used in conjunction with the documents it references. Information referenced in this way is as much a part of this document as is the information explicitly included here.

I Introductory Elements

1 Scope

The Trial Use Specification defines components which are not required parts of the LSB Specification.

2 Normative References

The specifications listed below are referenced in whole or in part by the Trial Use Specification. Such references may be normative or informative; a reference to specification shall only be considered normative if it is explicitly cited as such. The Trial Use Specification may make normative references to a portion of these specifications (that is, to define a specific function or group of functions); in such cases, only the explicitly referenced portion of the specification is to be considered normative.

Table 2-1 Informative References

Name	Title	URL
ALSA Library API Reference	ALSA Library API Reference	http://www.alsa-project.org/alsa-doc/alsa-lib/
ISO C (1999)	ISO/IEC 9899: 1999, Programming Languages --C	
xdg-utils reference	Portland Project XDG Utilities Reference 1.0	http://portland.freedesktop.org/xdg-utils-1.0/

3 Requirements

3.1 Relevant Libraries

The libraries listed in Table 3-1 shall be available on a Linux Standard Base system, with the specified runtime names. This list may be supplemented or amended by the architecture-specific specification.

Table 3-1 Standard Library Names

Library	Runtime Name
libasound	libasound.so.2

These libraries will be in an implementation-defined directory which the dynamic linker shall search by default.

4 Definitions

For the purposes of this document, the following definitions, as specified in the *ISO/IEC Directives, Part 2, 2001, 4th Edition*, apply:

can

be able to; there is a possibility of; it is possible to

cannot

be unable to; there is no possibilty of; it is not possible to

may

is permitted; is allowed; is permissible

need not

it is not required that; no...is required

shall

is to; is required to; it is required that; has to; only...is permitted; it is necessary

shall not

is not allowed [permitted] [acceptable] [permissible]; is required to be not; is required that...be not; is not to be

should

it is recommended that; ought to

should not

it is not recommended that; ought not to

5 Terminology

For the purposes of this document, the following terms apply:

implementation-defined

Describes a value or behavior that is not defined by this document but is selected by an implementor. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations. The implementor shall document such a value or behavior so that it can be used correctly by an application.

Shell Script

A file that is read by an interpreter (e.g., awk). The first line of the shell script includes a reference to its interpreter binary.

undefined

Describes the nature of a value or behavior not defined by this document which results from use of an invalid program construct or invalid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

unspecified

Describes the nature of a value or behavior not specified by this document which results from use of a valid program construct or valid data input. The value or behavior may vary among implementations that conform to this document. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

II ALSA sound library

6 Libraries

6.1 Interfaces for libasound

Table 6-1 defines the library name and shared object name for the libasound library

Table 6-1 libasound Definition

Library:	libasound
SONAME:	libasound.so.2

The behavior of the interfaces in this library is specified by the following specifications:

[ALSA] ALSA Library API Reference

6.1.1 Default LibGroup for libasound

6.1.1.1 Interfaces for Default LibGroup for libasound

No external functions are defined for libasound - Default LibGroup for libasound in this part of the specification. See also the relevant architecture specific part of this specification.

6.1.2 ALSA Configuration Interface

6.1.2.1 Interfaces for ALSA Configuration Interface

An LSB conforming implementation shall provide the generic functions for ALSA Configuration Interface specified in Table 6-2, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-2 libasound - ALSA Configuration Interface Function Interfaces

snd_config_add(ALSA_0.9) [ALSA]	snd_config_copy(ALSA_0.9) [ALSA]	snd_config_delete(ALSA_0.9) [ALSA]
snd_config_get_ascii(ALSA_0.9) [ALSA]	snd_config_get_id(ALSA_0.9) [ALSA]	snd_config_get_integer(ALSA_0.9) [ALSA]
snd_config_get_integer64(ALSA_0.9) [ALSA]	snd_config_get_string(ALSA_0.9) [ALSA]	snd_config_get_type(ALSA_0.9) [ALSA]
snd_config_imake_integer(ALSA_0.9) [ALSA]	snd_config_imake_integer64(ALSA_0.9) [ALSA]	snd_config_imake_string(ALSA_0.9) [ALSA]
snd_config_iterator_end(ALSA_0.9) [ALSA]	snd_config_iterator_entry(ALSA_0.9) [ALSA]	snd_config_iterator_first(ALSA_0.9) [ALSA]
snd_config_iterator_next(ALSA_0.9) [ALSA]	snd_config_load(ALSA_0.9) [ALSA]	snd_config_make_compound(ALSA_0.9) [ALSA]
snd_config_make_integer(ALSA_0.9) [ALSA]	snd_config_make_integer64(ALSA_0.9) [ALSA]	snd_config_make_string(ALSA_0.9) [ALSA]
snd_config_save(ALSA	snd_config_search(ALS	snd_config_searchv(AL

_0.9) [ALSA]	A_0.9) [ALSA]	SA_0.9) [ALSA]
snd_config_set_ascii(ALSA_0.9) [ALSA]	snd_config_set_integer(ALSA_0.9) [ALSA]	snd_config_set_integer64(ALSA_0.9) [ALSA]
snd_config_set_string(ALSA_0.9) [ALSA]	snd_config_top(ALSA_0.9) [ALSA]	snd_config_update(ALSA_0.9) [ALSA]
snd_config_update_free_global(ALSA_0.9) [ALSA]		

An LSB conforming implementation shall provide the generic data interfaces for ALSA Configuration Interface specified in Table 6-3, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-3 libasound - ALSA Configuration Interface Data Interfaces

snd_config(ALSA_0.9) [ALSA]		

6.1.3 ALSA Control Interface

6.1.3.1 Interfaces for ALSA Control Interface

An LSB conforming implementation shall provide the generic functions for ALSA Control Interface specified in Table 6-4, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-4 libasound - ALSA Control Interface Function Interfaces

snd_async_add_ctl_handler(ALSA_0.9) [ALSA]	snd_async_handler_get_ctl(ALSA_0.9) [ALSA]	snd_card_get_index(ALSA_0.9) [ALSA]
snd_card_get_longname(ALSA_0.9) [ALSA]	snd_card_get_name(ALSA_0.9) [ALSA]	snd_card_load(ALSA_0.9) [ALSA]
snd_card_next(ALSA_0.9) [ALSA]	snd_ctl_card_info(ALSA_0.9) [ALSA]	snd_ctl_card_info_clear(ALSA_0.9) [ALSA]
snd_ctl_card_info_copy(ALSA_0.9) [ALSA]	snd_ctl_card_info_free(ALSA_0.9) [ALSA]	snd_ctl_card_info_get_components(ALSA_0.9) [ALSA]
snd_ctl_card_info_get_driver(ALSA_0.9) [ALSA]	snd_ctl_card_info_get_id(ALSA_0.9) [ALSA]	snd_ctl_card_info_get_longname(ALSA_0.9) [ALSA]
snd_ctl_card_info_get_mixername(ALSA_0.9) [ALSA]	snd_ctl_card_info_get_name(ALSA_0.9) [ALSA]	snd_ctl_card_info_malloc(ALSA_0.9) [ALSA]
snd_ctl_card_info_sizeof(ALSA_0.9) [ALSA]	snd_ctl_close(ALSA_0.9) [ALSA]	snd_ctl_elem_add_boolean(ALSA_0.9) [ALSA]
snd_ctl_elem_add_iec958(ALSA_0.9) [ALSA]	snd_ctl_elem_add_integer(ALSA_0.9) [ALSA]	snd_ctl_elem_id_clear(ALSA_0.9) [ALSA]
snd_ctl_elem_id_copy(ALSA_0.9) [ALSA]	snd_ctl_elem_id_free(ALSA_0.9) [ALSA]	snd_ctl_elem_id_get_device(ALSA_0.9) [ALSA]

snd_ctl_elem_id_get_index(ALSA_0.9) [ALSA]	snd_ctl_elem_id_get_interface(ALSA_0.9) [ALSA]	snd_ctl_elem_id_get_name(ALSA_0.9) [ALSA]
snd_ctl_elem_id_get_numid(ALSA_0.9) [ALSA]	snd_ctl_elem_id_get_subdevice(ALSA_0.9) [ALSA]	snd_ctl_elem_id_malloc(ALSA_0.9) [ALSA]
snd_ctl_elem_id_set_device(ALSA_0.9) [ALSA]	snd_ctl_elem_id_set_index(ALSA_0.9) [ALSA]	snd_ctl_elem_id_set_interface(ALSA_0.9) [ALSA]
snd_ctl_elem_id_set_name(ALSA_0.9) [ALSA]	snd_ctl_elem_id_set_numid(ALSA_0.9) [ALSA]	snd_ctl_elem_id_set_subdevice(ALSA_0.9) [ALSA]
snd_ctl_elem_id_sizeof(ALSA_0.9) [ALSA]	snd_ctl_elem_iface_name(ALSA_0.9) [ALSA]	snd_ctl_elem_info(ALSA_0.9) [ALSA]
snd_ctl_elem_info_clear(ALSA_0.9) [ALSA]	snd_ctl_elem_info_copy(ALSA_0.9) [ALSA]	snd_ctl_elem_info_free(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_count(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_id(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_item_name(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_items(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_max(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_max64(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_min(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_min64(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_name(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_numid(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_step(ALSA_0.9) [ALSA]	snd_ctl_elem_info_get_step64(ALSA_0.9) [ALSA]
snd_ctl_elem_info_get_type(ALSA_0.9) [ALSA]	snd_ctl_elem_info_is_inactive(ALSA_0.9) [ALSA]	snd_ctl_elem_info_is_locked(ALSA_0.9) [ALSA]
snd_ctl_elem_info_is_readable(ALSA_0.9) [ALSA]	snd_ctl_elem_info_is_user(ALSA_0.9) [ALSA]	snd_ctl_elem_info_is_volatile(ALSA_0.9) [ALSA]
snd_ctl_elem_info_is_writable(ALSA_0.9) [ALSA]	snd_ctl_elem_info_malloc(ALSA_0.9) [ALSA]	snd_ctl_elem_info_set_id(ALSA_0.9) [ALSA]
snd_ctl_elem_info_set_item(ALSA_0.9) [ALSA]	snd_ctl_elem_info_sizeof(ALSA_0.9) [ALSA]	snd_ctl_elem_list(ALSA_0.9) [ALSA]
snd_ctl_elem_list_alloc_space(ALSA_0.9) [ALSA]	snd_ctl_elem_list_clear(ALSA_0.9) [ALSA]	snd_ctl_elem_list_copy(ALSA_0.9) [ALSA]
snd_ctl_elem_list_free(ALSA_0.9) [ALSA]	snd_ctl_elem_list_free_space(ALSA_0.9) [ALSA]	snd_ctl_elem_list_get_count(ALSA_0.9) [ALSA]

snd_ctl_elem_list_get_id(ALSA_0.9) [ALSA]	snd_ctl_elem_list_get_name(ALSA_0.9) [ALSA]	snd_ctl_elem_list_get_used(ALSA_0.9) [ALSA]
snd_ctl_elem_list_malloc(ALSA_0.9) [ALSA]	snd_ctl_elem_list_set_offset(ALSA_0.9) [ALSA]	snd_ctl_elem_list_sizeof(ALSA_0.9) [ALSA]
snd_ctl_elem_read(ALSA_0.9) [ALSA]	snd_ctl_elem_remove(ALSA_0.9) [ALSA]	snd_ctl_elem_type_name(ALSA_0.9) [ALSA]
snd_ctl_elem_value_clear(ALSA_0.9) [ALSA]	snd_ctl_elem_value_copy(ALSA_0.9) [ALSA]	snd_ctl_elem_value_free(ALSA_0.9) [ALSA]
snd_ctl_elem_value_get_boolean(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_byte(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_bytes(ALSA_0.9) [ALSA]
snd_ctl_elem_value_get_enumerated(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_id(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_iec958(ALSA_0.9) [ALSA]
snd_ctl_elem_value_get_integer(ALSA_0.9) [ALSA]	snd_ctl_elem_value_get_integer64(ALSA_0.9) [ALSA]	snd_ctl_elem_value_malloc(ALSA_0.9) [ALSA]
snd_ctl_elem_value_set_boolean(ALSA_0.9) [ALSA]	snd_ctl_elem_value_set_byte(ALSA_0.9) [ALSA]	snd_ctl_elem_value_set_enumerated(ALSA_0.9) [ALSA]
snd_ctl_elem_value_set_id(ALSA_0.9) [ALSA]	snd_ctl_elem_value_set_iec958(ALSA_0.9) [ALSA]	snd_ctl_elem_value_set_integer(ALSA_0.9) [ALSA]
snd_ctl_elem_value_set_integer64(ALSA_0.9) [ALSA]	snd_ctl_elem_value_sizeof(ALSA_0.9) [ALSA]	snd_ctl_elem_write(ALSA_0.9) [ALSA]
snd_ctl_event_clear(ALSA_0.9) [ALSA]	snd_ctl_event_copy(ALSA_0.9) [ALSA]	snd_ctl_event_elem_get_id(ALSA_0.9) [ALSA]
snd_ctl_event_elem_get_mask(ALSA_0.9) [ALSA]	snd_ctl_event_free(ALSA_0.9) [ALSA]	snd_ctl_event_malloc(ALSA_0.9) [ALSA]
snd_ctl_event_sizeof(ALSA_0.9) [ALSA]	snd_ctl_hwdep_info(ALSA_0.9) [ALSA]	snd_ctl_hwdep_next_device(ALSA_0.9) [ALSA]
snd_ctl_name(ALSA_0.9) [ALSA]	snd_ctl_nonblock(ALSA_0.9) [ALSA]	snd_ctl_open(ALSA_0.9) [ALSA]
snd_ctl_pcm_info(ALSA_0.9) [ALSA]	snd_ctl_pcm_next_device(ALSA_0.9) [ALSA]	snd_ctl_poll_descriptors(ALSA_0.9) [ALSA]
snd_ctl_poll_descriptors_count(ALSA_0.9) [ALSA]	snd_ctl_rawmidi_info(ALSA_0.9) [ALSA]	snd_ctl_rawmidi_next_device(ALSA_0.9) [ALSA]
snd_ctl_read(ALSA_0.9) [ALSA]	snd_ctl_subscribe_events(ALSA_0.9) [ALSA]	

6.1.4 ALSA Global defines and functions

6.1.4.1 Interfaces for ALSA Global defines and functions

An LSB conforming implementation shall provide the generic functions for ALSA Global defines and functions specified in Table 6-5, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-5 libasound - ALSA Global defines and functions Function Interfaces

snd_asoundlib_version(ALSA_0.9) [ALSA]	snd_async_add_handler(ALSA_0.9) [ALSA]	snd_async_del_handler(ALSA_0.9) [ALSA]
snd_async_handler_get_callback_private(ALSA_0.9) [ALSA]		

6.1.5 ALSA Hardware Dependant Interface

6.1.5.1 Interfaces for ALSA Hardware Dependant Interface

An LSB conforming implementation shall provide the generic functions for ALSA Hardware Dependant Interface specified in Table 6-6, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-6 libasound - ALSA Hardware Dependant Interface Function Interfaces

snd_hwdep_close(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_copy(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_free(ALSA_0.9) [ALSA]
snd_hwdep_dsp_image_get_image(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_get_index(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_get_length(ALSA_0.9) [ALSA]
snd_hwdep_dsp_image_get_name(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_malloc(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_set_image(ALSA_0.9) [ALSA]
snd_hwdep_dsp_image_set_index(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_set_length(ALSA_0.9) [ALSA]	snd_hwdep_dsp_image_set_name(ALSA_0.9) [ALSA]
snd_hwdep_dsp_image_sizeof(ALSA_0.9) [ALSA]	snd_hwdep_dsp_load(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status(ALSA_0.9) [ALSA]
snd_hwdep_dsp_status_copy(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_free(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_get_chip_ready(ALSA_0.9) [ALSA]
snd_hwdep_dsp_status_get_dsp_loaded(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_get_id(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_get_num_dsps(ALSA_0.9) [ALSA]
snd_hwdep_dsp_status_get_version(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_malloc(ALSA_0.9) [ALSA]	snd_hwdep_dsp_status_sizeof(ALSA_0.9) [ALSA]

snd_hwdep_info(ALSA_0.9) [ALSA]	snd_hwdep_info_copy(ALSA_0.9) [ALSA]	snd_hwdep_info_free(ALSA_0.9) [ALSA]
snd_hwdep_info_get_card(ALSA_0.9) [ALSA]	snd_hwdep_info_get_device(ALSA_0.9) [ALSA]	snd_hwdep_info_get_id(ALSA_0.9) [ALSA]
snd_hwdep_info_get_iface(ALSA_0.9) [ALSA]	snd_hwdep_info_get_name(ALSA_0.9) [ALSA]	snd_hwdep_info_malloc(ALSA_0.9) [ALSA]
snd_hwdep_info_set_device(ALSA_0.9) [ALSA]	snd_hwdep_info_sizeof(ALSA_0.9) [ALSA]	snd_hwdep_ioctl(ALSA_0.9) [ALSA]
snd_hwdep_open(ALSA_0.9) [ALSA]	snd_hwdep_poll_descriptors(ALSA_0.9) [ALSA]	snd_hwdep_read(ALSA_0.9) [ALSA]
snd_hwdep_write(ALSA_0.9) [ALSA]		

6.1.6 ALSA High level Control Interface

6.1.6.1 Interfaces for ALSA High level Control Interface

An LSB conforming implementation shall provide the generic functions for ALSA High level Control Interface specified in Table 6-7, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-7 libasound - ALSA High level Control Interface Function Interfaces

snd_hctl_close(ALSA_0.9) [ALSA]	snd_hctl_elem_get_callback_private(ALSA_0.9) [ALSA]	snd_hctl_elem_get_id(ALSA_0.9) [ALSA]
snd_hctl_elem_info(ALSA_0.9) [ALSA]	snd_hctl_elem_next(ALSA_0.9) [ALSA]	snd_hctl_elem_prev(ALSA_0.9) [ALSA]
snd_hctl_elem_read(ALSA_0.9) [ALSA]	snd_hctl_elem_set_callback(ALSA_0.9) [ALSA]	snd_hctl_elem_set_callback_private(ALSA_0.9) [ALSA]
snd_hctl_elem_write(ALSA_0.9) [ALSA]	snd_hctl_find_elem(ALSA_0.9) [ALSA]	snd_hctl_first_elem(ALSA_0.9) [ALSA]
snd_hctl_free(ALSA_0.9) [ALSA]	snd_hctl_get_callback_private(ALSA_0.9) [ALSA]	snd_hctl_last_elem(ALSA_0.9) [ALSA]
snd_hctl_load(ALSA_0.9) [ALSA]	snd_hctl_open(ALSA_0.9) [ALSA]	snd_hctl_set_callback(ALSA_0.9) [ALSA]
snd_hctl_set_callback_private(ALSA_0.9) [ALSA]		

6.1.7 ALSA Input Interface

6.1.7.1 Interfaces for ALSA Input Interface

An LSB conforming implementation shall provide the generic functions for ALSA Input Interface specified in Table 6-8, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-8 libasound - ALSA Input Interface Function Interfaces

snd_input_buffer_open (ALSA_0.9) [ALSA]	snd_input_close(ALSA_ 0.9) [ALSA]	snd_input_stdio_attach(ALSA_0.9) [ALSA]
snd_input_stdio_open(ALSA_0.9) [ALSA]		

6.1.8 ALSA MIDI Sequencer

6.1.8.1 Interfaces for ALSA MIDI Sequencer

An LSB conforming implementation shall provide the generic functions for ALSA MIDI Sequencer specified in Table 6-9, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-9 libasound - ALSA MIDI Sequencer Function Interfaces

snd_seq_client_id(ALS A_0.9) [ALSA]	snd_seq_close(ALSA_0. 9) [ALSA]	snd_seq_get_input_buff er_size(ALSA_0.9) [ALSA]
snd_seq_get_output_bu ffer_size(ALSA_0.9) [ALSA]	snd_seq_nonblock(ALS A_0.9) [ALSA]	snd_seq_open(ALSA_0. 9) [ALSA]
snd_seq_poll_descripto rs(ALSA_0.9) [ALSA]	snd_seq_poll_descripto rs_count(ALSA_0.9) [ALSA]	snd_seq_poll_descripto rs_revents(ALSA_0.9) [ALSA]
snd_seq_set_input_buff er_size(ALSA_0.9) [ALSA]	snd_seq_set_output_bu ffer_size(ALSA_0.9) [ALSA]	snd_seq_system_info(A LSA_0.9) [ALSA]
snd_seq_system_info_c opy(ALSA_0.9) [ALSA]	snd_seq_system_info_fr ee(ALSA_0.9) [ALSA]	snd_seq_system_info_g et_clients(ALSA_0.9) [ALSA]
snd_seq_system_info_g et_ports(ALSA_0.9) [ALSA]	snd_seq_system_info_g et_queues(ALSA_0.9) [ALSA]	snd_seq_system_info_ malloc(ALSA_0.9) [ALSA]
snd_seq_system_info_si zeof(ALSA_0.9) [ALSA]		

6.1.9 ALSA Mixer Interface

6.1.9.1 Interfaces for ALSA Mixer Interface

An LSB conforming implementation shall provide the generic functions for ALSA Mixer Interface specified in Table 6-10, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-10 libasound - ALSA Mixer Interface Function Interfaces

snd_mixer_attach(ALSA_0.9) [ALSA]	snd_mixer_close(ALSA_0.9) [ALSA]	snd_mixer_detach(ALSA_0.9) [ALSA]
snd_mixer_elem_get_callback_private(ALSA_0.9) [ALSA]	snd_mixer_elem_get_type(ALSA_0.9) [ALSA]	snd_mixer_elem_next(ALSA_0.9) [ALSA]
snd_mixer_elem_prev(ALSA_0.9) [ALSA]	snd_mixer_elem_set_callback(ALSA_0.9) [ALSA]	snd_mixer_elem_set_callback_private(ALSA_0.9) [ALSA]
snd_mixer_first_elem(ALSA_0.9) [ALSA]	snd_mixer_free(ALSA_0.9) [ALSA]	snd_mixer_get_callback_private(ALSA_0.9) [ALSA]
snd_mixer_get_count(ALSA_0.9) [ALSA]	snd_mixer_handle_events(ALSA_0.9) [ALSA]	snd_mixer_last_elem(ALSA_0.9) [ALSA]
snd_mixer_load(ALSA_0.9) [ALSA]	snd_mixer_open(ALSA_0.9) [ALSA]	snd_mixer_poll_descriptors(ALSA_0.9) [ALSA]
snd_mixer_poll_descriptors_count(ALSA_0.9) [ALSA]	snd_mixer_poll_descriptors_revents(ALSA_0.9) [ALSA]	snd_mixer_set_callback(ALSA_0.9) [ALSA]
snd_mixer_set_callback_private(ALSA_0.9) [ALSA]	snd_mixer_wait(ALSA_0.9) [ALSA]	snd_pcm_type_name(ALSA_0.9.0) [ALSA]

6.1.10 ALSA Output Interface

6.1.10.1 Interfaces for ALSA Output Interface

An LSB conforming implementation shall provide the generic functions for ALSA Output Interface specified in Table 6-11, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-11 libasound - ALSA Output Interface Function Interfaces

snd_output_buffer_open(ALSA_0.9) [ALSA]	snd_output_buffer_string(ALSA_0.9) [ALSA]	snd_output_close(ALSA_0.9) [ALSA]
snd_output_putc(ALSA_0.9) [ALSA]	snd_output_puts(ALSA_0.9) [ALSA]	snd_output_stdio_attach(ALSA_0.9) [ALSA]
snd_output_stdio_open(ALSA_0.9) [ALSA]		

6.1.11 ALSA PCM Interface - General Functions

6.1.11.1 Interfaces for ALSA PCM Interface - General Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - General Functions specified in Table 6-12, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-12 libasound - ALSA PCM Interface - General Functions Function Interfaces

snd_async_add_pcm_handler(ALSA_0.9) [ALSA]	snd_async_handler_get_pcm(ALSA_0.9) [ALSA]	snd_pcm_avail_update(ALSA_0.9) [ALSA]
snd_pcm_close(ALSA_0.9) [ALSA]	snd_pcm_delay(ALSA_0.9) [ALSA]	snd_pcm_drain(ALSA_0.9) [ALSA]
snd_pcm_drop(ALSA_0.9) [ALSA]	snd_pcm_forward(ALSA_0.9.0rc8) [ALSA]	snd_pcm_hw_free(ALSA_0.9) [ALSA]
snd_pcm_hw_params(ALSA_0.9) [ALSA]	snd_pcm_hw_params_current(ALSA_0.9) [ALSA]	snd_pcm_hwsync(ALSA_0.9) [ALSA]
snd_pcm_info(ALSA_0.9) [ALSA]	snd_pcm_link(ALSA_0.9) [ALSA]	snd_pcm_name(ALSA_0.9) [ALSA]
snd_pcm_nonblock(ALSA_0.9) [ALSA]	snd_pcm_open(ALSA_0.9) [ALSA]	snd_pcm_open_lconf(ALSA_0.9) [ALSA]
snd_pcm_pause(ALSA_0.9) [ALSA]	snd_pcm_poll_descriptors(ALSA_0.9) [ALSA]	snd_pcm_poll_descriptors_count(ALSA_0.9) [ALSA]
snd_pcm_poll_descriptors_revents(ALSA_0.9) [ALSA]	snd_pcm_prepare(ALSA_0.9) [ALSA]	snd_pcm_readi(ALSA_0.9) [ALSA]
snd_pcm_readn(ALSA_0.9) [ALSA]	snd_pcm_recover(ALSA_0.9) [ALSA]	snd_pcm_reset(ALSA_0.9) [ALSA]
snd_pcm_resume(ALSA_0.9) [ALSA]	snd_pcm_rewind(ALSA_0.9) [ALSA]	snd_pcm_start(ALSA_0.9) [ALSA]
snd_pcm_state(ALSA_0.9) [ALSA]	snd_pcm_status(ALSA_0.9) [ALSA]	snd_pcm_stream(ALSA_0.9) [ALSA]
snd_pcm_sw_params(ALSA_0.9) [ALSA]	snd_pcm_sw_params_current(ALSA_0.9) [ALSA]	snd_pcm_type(ALSA_0.9) [ALSA]
snd_pcm_unlink(ALSA_0.9) [ALSA]	snd_pcm_wait(ALSA_0.9) [ALSA]	snd_pcm_writei(ALSA_0.9) [ALSA]
snd_pcm_writen(ALSA_0.9) [ALSA]		

6.1.12 ALSA PCM Interface - Access Mask Functions

6.1.12.1 Interfaces for ALSA PCM Interface - Access Mask Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Access Mask Functions specified in Table 6-13, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-13 libasound - ALSA PCM Interface - Access Mask Functions Function Interfaces

snd_pcm_access_mask_any(ALSA_0.9) [ALSA]	snd_pcm_access_mask_copy(ALSA_0.9) [ALSA]	snd_pcm_access_mask_free(ALSA_0.9) [ALSA]
snd_pcm_access_mask_malloc(ALSA_0.9) [ALSA]	snd_pcm_access_mask_none(ALSA_0.9) [ALSA]	snd_pcm_access_mask_set(ALSA_0.9) [ALSA]
snd_pcm_access_mask_sizeof(ALSA_0.9) [ALSA]	snd_pcm_access_mask_test(ALSA_0.9) [ALSA]	

6.1.13 ALSA PCM Interface - Debug Functions

6.1.13.1 Interfaces for ALSA PCM Interface - Debug Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Debug Functions specified in Table 6-14, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-14 libasound - ALSA PCM Interface - Debug Functions Function Interfaces

snd_pcm_dump(ALSA_0.9) [ALSA]	snd_pcm_hw_params_dump(ALSA_0.9) [ALSA]	snd_pcm_status_dump(ALSA_0.9) [ALSA]
snd_pcm_sw_params_dump(ALSA_0.9) [ALSA]		

6.1.14 ALSA PCM Interface - Description Functions

6.1.14.1 Interfaces for ALSA PCM Interface - Description Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Description Functions specified in Table 6-15, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-15 libasound - ALSA PCM Interface - Description Functions Function Interfaces

snd_pcm_access_name(ALSA_0.9) [ALSA]	snd_pcm_format_description(ALSA_0.9) [ALSA]	snd_pcm_format_name(ALSA_0.9) [ALSA]
snd_pcm_format_value(ALSA_0.9) [ALSA]	snd_pcm_state_name(ALSA_0.9) [ALSA]	snd_pcm_stream_name(ALSA_0.9) [ALSA]

6.1.15 ALSA PCM Interface - Direct Access (MMAP) Functions

6.1.15.1 Interfaces for ALSA PCM Interface - Direct Access (MMAP) Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Direct Access (MMAP) Functions specified in Table 6-16, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-16 libasound - ALSA PCM Interface - Direct Access (MMAP) Functions Function Interfaces

snd_pcm_mmap_begin(ALSA_0.9) [ALSA]	snd_pcm_mmap_commit(ALSA_0.9) [ALSA]	snd_pcm_mmap_readi(ALSA_0.9) [ALSA]
snd_pcm_mmap_readn(ALSA_0.9) [ALSA]	snd_pcm_mmap_writei(ALSA_0.9) [ALSA]	snd_pcm_mmap_writen(ALSA_0.9) [ALSA]

6.1.16 ALSA PCM Interface - Format Mask Functions

6.1.16.1 Interfaces for ALSA PCM Interface - Format Mask Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Format Mask Functions specified in Table 6-17, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-17 libasound - ALSA PCM Interface - Format Mask Functions Function Interfaces

snd_pcm_format_mask_any(ALSA_0.9) [ALSA]	snd_pcm_format_mask_copy(ALSA_0.9) [ALSA]	snd_pcm_format_mask_free(ALSA_0.9) [ALSA]
snd_pcm_format_mask_malloc(ALSA_0.9) [ALSA]	snd_pcm_format_mask_none(ALSA_0.9) [ALSA]	snd_pcm_format_mask_set(ALSA_0.9) [ALSA]
snd_pcm_format_mask_sizeof(ALSA_0.9) [ALSA]	snd_pcm_format_mask_test(ALSA_0.9) [ALSA]	

6.1.17 ALSA PCM Interface - Hardware Parameters

6.1.17.1 Interfaces for ALSA PCM Interface - Hardware Parameters

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Hardware Parameters specified in Table 6-18, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-18 libasound - ALSA PCM Interface - Hardware Parameters Function Interfaces

snd_pcm_hw_params_any(ALSA_0.9) [ALSA]	snd_pcm_hw_params_can_mmap_sample_resolution(ALSA_0.9)	snd_pcm_hw_params_can_pause(ALSA_0.9) [ALSA]

	[ALSA]	
snd_pcm_hw_params_can_resume(ALSA_0.9) [ALSA]	snd_pcm_hw_params_can_sync_start(ALSA_0.9) [ALSA]	snd_pcm_hw_params_copy(ALSA_0.9) [ALSA]
snd_pcm_hw_params_free(ALSA_0.9) [ALSA]	snd_pcm_hw_params_get_access(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_access_mask(ALSA_0.9) [ALSA]
snd_pcm_hw_params_get_buffer_size(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_buffer_size_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_buffer_size_min(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_buffer_time(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_buffer_time_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_buffer_time_min(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_channels(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_channels_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_channels_min(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_format(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_format_mask(ALSA_0.9) [ALSA]	snd_pcm_hw_params_get_period_size(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_period_size_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_period_size_min(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_period_time(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_period_time_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_period_time_min(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_periods(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_periods_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_periods_min(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_rate(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_get_rate_max(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_rate_min(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_get_rate_numden(ALSA_0.9) [ALSA]
snd_pcm_hw_params_get_rate_resample(ALSA_0.9) [ALSA]	snd_pcm_hw_params_get_sbits(ALSA_0.9) [ALSA]	snd_pcm_hw_params_is_double(ALSA_0.9) [ALSA]
snd_pcm_hw_params_is_half_duplex(ALSA_0.9) [ALSA]	snd_pcm_hw_params_is_joint_duplex(ALSA_0.9) [ALSA]	snd_pcm_hw_params_malloc(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_access(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_access_mask(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_buffer_size(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_buffer_size_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_buffer_time(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_buffer_time_near(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_set_channels(ALSA_0.9)	snd_pcm_hw_params_set_channels_near(ALSA	snd_pcm_hw_params_set_format(ALSA_0.9)

[ALSA]	_0.9.0rc4) [ALSA]	[ALSA]
snd_pcm_hw_params_set_format_mask(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_period_size(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_period_size_near(ALSA_0.9.0rc4) [ALSA]
snd_pcm_hw_params_set_period_time(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_period_time_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_periods(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_periods_integer(ALSA_0.9) [ALSA]	snd_pcm_hw_params_set_periods_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_rate(ALSA_0.9) [ALSA]
snd_pcm_hw_params_set_rate_near(ALSA_0.9.0rc4) [ALSA]	snd_pcm_hw_params_set_rate_resample(ALSA_0.9) [ALSA]	snd_pcm_hw_params_sizeof(ALSA_0.9) [ALSA]
snd_pcm_hw_params_test_access(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_buffer_size(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_buffer_time(ALSA_0.9) [ALSA]
snd_pcm_hw_params_test_channels(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_format(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_period_size(ALSA_0.9) [ALSA]
snd_pcm_hw_params_test_period_time(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_periods(ALSA_0.9) [ALSA]	snd_pcm_hw_params_test_rate(ALSA_0.9) [ALSA]

6.1.18 ALSA PCM Interface - Helper Functions

6.1.18.1 Interfaces for ALSA PCM Interface - Helper Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Helper Functions specified in Table 6-19, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-19 libasound - ALSA PCM Interface - Helper Functions Function Interfaces

snd_pcm_area_copy(ALSA_0.9) [ALSA]	snd_pcm_area_silence(ALSA_0.9) [ALSA]	snd_pcm_areas_copy(ALSA_0.9) [ALSA]
snd_pcm_areas_silence(ALSA_0.9) [ALSA]	snd_pcm_build_linear_format(ALSA_0.9) [ALSA]	snd_pcm_bytes_to_frames(ALSA_0.9) [ALSA]
snd_pcm_bytes_to_samples(ALSA_0.9) [ALSA]	snd_pcm_format_big_endian(ALSA_0.9) [ALSA]	snd_pcm_format_cpu_endian(ALSA_0.9) [ALSA]
snd_pcm_format_float(ALSA_0.9) [ALSA]	snd_pcm_format_linear(ALSA_0.9) [ALSA]	snd_pcm_format_little_endian(ALSA_0.9) [ALSA]
snd_pcm_format_physical_width(ALSA_0.9) [ALSA]	snd_pcm_format_set_silence(ALSA_0.9) [ALSA]	snd_pcm_format_signed(ALSA_0.9) [ALSA]

snd_pcm_format_size(ALSA_0.9) [ALSA]	snd_pcm_format_unsigned(ALSA_0.9) [ALSA]	snd_pcm_format_width(ALSA_0.9) [ALSA]
snd_pcm_frames_to_bytes(ALSA_0.9) [ALSA]	snd_pcm_samples_to_bytes(ALSA_0.9) [ALSA]	

6.1.19 ALSA PCM Interface - Software Parameters

6.1.19.1 Interfaces for ALSA PCM Interface - Software Parameters

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Software Parameters specified in Table 6-20, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-20 libasound - ALSA PCM Interface - Software Parameters Function Interfaces

snd_pcm_sw_params_copy(ALSA_0.9) [ALSA]	snd_pcm_sw_params_free(ALSA_0.9) [ALSA]	snd_pcm_sw_params_get_avail_min(ALSA_0.9.0rc4) [ALSA]
snd_pcm_sw_params_get_boundary(ALSA_0.9) [ALSA]	snd_pcm_sw_params_get_silence_size(ALSA_0.9.0rc4) [ALSA]	snd_pcm_sw_params_get_silence_threshold(ALSA_0.9.0rc4) [ALSA]
snd_pcm_sw_params_get_start_threshold(ALSA_0.9.0rc4) [ALSA]	snd_pcm_sw_params_get_stop_threshold(ALSA_0.9.0rc4) [ALSA]	snd_pcm_sw_params_get_tstamp_mode(ALSA_0.9.0rc4) [ALSA]
snd_pcm_sw_params_malloc(ALSA_0.9) [ALSA]	snd_pcm_sw_params_set_avail_min(ALSA_0.9) [ALSA]	snd_pcm_sw_params_set_silence_size(ALSA_0.9) [ALSA]
snd_pcm_sw_params_set_silence_threshold(ALSA_0.9) [ALSA]	snd_pcm_sw_params_set_start_threshold(ALSA_0.9) [ALSA]	snd_pcm_sw_params_set_stop_threshold(ALSA_0.9) [ALSA]
snd_pcm_sw_params_set_tstamp_mode(ALSA_0.9) [ALSA]	snd_pcm_sw_params_set_xfer_align(ALSA_0.9) [ALSA]	snd_pcm_sw_params_sizeof(ALSA_0.9) [ALSA]

6.1.20 ALSA PCM Interface - Status Functions

6.1.20.1 Interfaces for ALSA PCM Interface - Status Functions

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Status Functions specified in Table 6-21, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-21 libasound - ALSA PCM Interface - Status Functions Function Interfaces

snd_pcm_status_copy(ALSA_0.9) [ALSA]	snd_pcm_status_free(ALSA_0.9) [ALSA]	snd_pcm_status_get_avail(ALSA_0.9) [ALSA]
snd_pcm_status_get_avail_max(ALSA_0.9) [ALSA]	snd_pcm_status_get_delay(ALSA_0.9) [ALSA]	snd_pcm_status_get_state(ALSA_0.9) [ALSA]

snd_pcm_status_get_trigger_tstamp(ALSA_0.9) [ALSA]	snd_pcm_status_get_tstamp(ALSA_0.9) [ALSA]	snd_pcm_status_malloc(ALSA_0.9) [ALSA]
snd_pcm_status_sizeof(ALSA_0.9) [ALSA]		

6.1.21 ALSA PCM Interface - Stream Information

6.1.21.1 Interfaces for ALSA PCM Interface - Stream Information

An LSB conforming implementation shall provide the generic functions for ALSA PCM Interface - Stream Information specified in Table 6-22, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-22 libasound - ALSA PCM Interface - Stream Information Function Interfaces

snd_pcm_info_copy(ALSA_0.9) [ALSA]	snd_pcm_info_free(ALSA_0.9) [ALSA]	snd_pcm_info_get_card(ALSA_0.9) [ALSA]
snd_pcm_info_get_class(ALSA_0.9) [ALSA]	snd_pcm_info_get_device(ALSA_0.9) [ALSA]	snd_pcm_info_get_id(ALSA_0.9) [ALSA]
snd_pcm_info_get_name(ALSA_0.9) [ALSA]	snd_pcm_info_get_stream(ALSA_0.9) [ALSA]	snd_pcm_info_get_subdevice(ALSA_0.9) [ALSA]
snd_pcm_info_get_subdevice_name(ALSA_0.9) [ALSA]	snd_pcm_info_get_subdevices_avail(ALSA_0.9) [ALSA]	snd_pcm_info_get_subdevices_count(ALSA_0.9) [ALSA]
snd_pcm_info_malloc(ALSA_0.9) [ALSA]	snd_pcm_info_set_device(ALSA_0.9) [ALSA]	snd_pcm_info_set_stream(ALSA_0.9) [ALSA]
snd_pcm_info_set_subdevice(ALSA_0.9) [ALSA]	snd_pcm_info_sizeof(ALSA_0.9) [ALSA]	

6.1.22 ALSA Sequencer Event Type Checks

6.1.22.1 Interfaces for ALSA Sequencer Event Type Checks

No external functions are defined for libasound - ALSA Sequencer Event Type Checks in this part of the specification. See also the relevant architecture specific part of this specification.

An LSB conforming implementation shall provide the generic data interfaces for ALSA Sequencer Event Type Checks specified in Table 6-23, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-23 libasound - ALSA Sequencer Event Type Checks Data Interfaces

snd_seq_event_types(ALSA_0.9) [ALSA]		

6.1.23 ALSA Error Handling

6.1.23.1 Interfaces for ALSA Error Handling

An LSB conforming implementation shall provide the generic functions for ALSA Error Handling specified in Table 6-24, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-24 libasound - ALSA Error Handling Function Interfaces

snd_lib_error_set_handler(ALSA_0.9) [ALSA]	snd_strerror(ALSA_0.9) [ALSA]	

6.1.24 ALSA RawMidi Interface

6.1.24.1 Interfaces for ALSA RawMidi Interface

An LSB conforming implementation shall provide the generic functions for ALSA RawMidi Interface specified in Table 6-25, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-25 libasound - ALSA RawMidi Interface Function Interfaces

snd_rawmidi_close(ALSA_0.9) [ALSA]	snd_rawmidi_drain(ALSA_0.9) [ALSA]	snd_rawmidi_drop(ALSA_0.9) [ALSA]
snd_rawmidi_info_free(ALSA_0.9) [ALSA]	snd_rawmidi_info_get_id(ALSA_0.9) [ALSA]	snd_rawmidi_info_get_name(ALSA_0.9) [ALSA]
snd_rawmidi_info_get_subdevices_count(ALSA_0.9) [ALSA]	snd_rawmidi_info_malloc(ALSA_0.9) [ALSA]	snd_rawmidi_info_set_device(ALSA_0.9) [ALSA]
snd_rawmidi_info_set_stream(ALSA_0.9) [ALSA]	snd_rawmidi_info_set_subdevice(ALSA_0.9) [ALSA]	snd_rawmidi_nonblock(ALSA_0.9) [ALSA]
snd_rawmidi_open(ALSA_0.9) [ALSA]	snd_rawmidi_poll_descriptors(ALSA_0.9) [ALSA]	snd_rawmidi_poll_descriptors_count(ALSA_0.9) [ALSA]
snd_rawmidi_poll_descriptors_revents(ALSA_0.9) [ALSA]	snd_rawmidi_read(ALSA_0.9) [ALSA]	snd_rawmidi_write(ALSA_0.9) [ALSA]

6.1.25 ALSA Sequencer Client Interface

6.1.25.1 Interfaces for ALSA Sequencer Client Interface

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Client Interface specified in Table 6-26, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-26 libasound - ALSA Sequencer Client Interface Function Interfaces

snd_seq_client_info_copy(ALSA_0.9) [ALSA]	snd_seq_client_info_free(ALSA_0.9) [ALSA]	snd_seq_client_info_get_client(ALSA_0.9) [ALSA]

snd_seq_client_info_get_name(ALSA_0.9) [ALSA]	snd_seq_client_info_get_num_ports(ALSA_0.9) [ALSA]	snd_seq_client_info_get_type(ALSA_0.9) [ALSA]
snd_seq_client_info_malloc(ALSA_0.9) [ALSA]	snd_seq_client_info_set_client(ALSA_0.9) [ALSA]	snd_seq_client_info_set_name(ALSA_0.9) [ALSA]
snd_seq_client_info_sizeof(ALSA_0.9) [ALSA]	snd_seq_get_any_client_info(ALSA_0.9) [ALSA]	snd_seq_get_client_info(ALSA_0.9) [ALSA]
snd_seq_query_next_client(ALSA_0.9) [ALSA]	snd_seq_set_client_info(ALSA_0.9) [ALSA]	

6.1.26 ALSA Sequencer Event API

6.1.26.1 Interfaces for ALSA Sequencer Event API

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Event API specified in Table 6-27, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-27 libasound - ALSA Sequencer Event API Function Interfaces

snd_seq_drain_output(ALSA_0.9) [ALSA]	snd_seq_drop_output(ALSA_0.9) [ALSA]	snd_seq_drop_output_buffer(ALSA_0.9) [ALSA]
snd_seq_event_input(ALSA_0.9) [ALSA]	snd_seq_event_input_pending(ALSA_0.9) [ALSA]	snd_seq_event_length(ALSA_0.9) [ALSA]
snd_seq_event_output(ALSA_0.9) [ALSA]	snd_seq_event_output_direct(ALSA_0.9) [ALSA]	snd_seq_free_event(ALSA_0.9) [ALSA]

6.1.27 ALSA Sequencer Middle Level Interface

6.1.27.1 Interfaces for ALSA Sequencer Middle Level Interface

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Middle Level Interface specified in Table 6-28, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-28 libasound - ALSA Sequencer Middle Level Interface Function Interfaces

snd_seq_connect_from(ALSA_0.9) [ALSA]	snd_seq_connect_to(ALSA_0.9) [ALSA]	snd_seq_control_queue(ALSA_0.9) [ALSA]
snd_seq_create_simple_port(ALSA_0.9) [ALSA]	snd_seq_delete_simple_port(ALSA_0.9) [ALSA]	snd_seq_disconnect_from(ALSA_0.9) [ALSA]
snd_seq_disconnect_to(ALSA_0.9) [ALSA]	snd_seq_parse_address(ALSA_0.9) [ALSA]	snd_seq_set_client_name(ALSA_0.9) [ALSA]
snd_seq_sync_output_queue(ALSA_0.9)		

[ALSA]		

6.1.28 ALSA Sequencer Port Interface

6.1.28.1 Interfaces for ALSA Sequencer Port Interface

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Port Interface specified in Table 6-29, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-29 libasound - ALSA Sequencer Port Interface Function Interfaces

snd_seq_create_port(ALSA_0.9) [ALSA]	snd_seq_delete_port(ALSA_0.9) [ALSA]	snd_seq_get_any_port_info(ALSA_0.9) [ALSA]
snd_seq_get_port_info(ALSA_0.9) [ALSA]	snd_seq_port_info_copy(ALSA_0.9) [ALSA]	snd_seq_port_info_free(ALSA_0.9) [ALSA]
snd_seq_port_info_get_addr(ALSA_0.9) [ALSA]	snd_seq_port_info_get_capability(ALSA_0.9) [ALSA]	snd_seq_port_info_get_client(ALSA_0.9) [ALSA]
snd_seq_port_info_get_name(ALSA_0.9) [ALSA]	snd_seq_port_info_get_port(ALSA_0.9) [ALSA]	snd_seq_port_info_get_type(ALSA_0.9) [ALSA]
snd_seq_port_info_malloc(ALSA_0.9) [ALSA]	snd_seq_port_info_set_capability(ALSA_0.9) [ALSA]	snd_seq_port_info_set_client(ALSA_0.9) [ALSA]
snd_seq_port_info_set_midi_channels(ALSA_0.9) [ALSA]	snd_seq_port_info_set_name(ALSA_0.9) [ALSA]	snd_seq_port_info_set_port(ALSA_0.9) [ALSA]
snd_seq_port_info_set_port_specified(ALSA_0.9) [ALSA]	snd_seq_port_info_set_timestamp_queue(ALSA_0.9) [ALSA]	snd_seq_port_info_set_timestamp_real(ALSA_0.9) [ALSA]
snd_seq_port_info_set_timestamping(ALSA_0.9) [ALSA]	snd_seq_port_info_set_type(ALSA_0.9) [ALSA]	snd_seq_port_info_sizeof(ALSA_0.9) [ALSA]
snd_seq_query_next_port(ALSA_0.9) [ALSA]	snd_seq_set_port_info(ALSA_0.9) [ALSA]	

6.1.29 ALSA Sequencer Port Subscription

6.1.29.1 Interfaces for ALSA Sequencer Port Subscription

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Port Subscription specified in Table 6-30, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-30 libasound - ALSA Sequencer Port Subscription Function Interfaces

snd_seq_get_port_subscription(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_copy(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_free(ALSA_0.9) [ALSA]
snd_seq_port_subscribe	snd_seq_port_subscribe	snd_seq_port_subscribe

_get_dest(ALSA_0.9) [ALSA]	_get_exclusive(ALSA_0.9) [ALSA]	_get_queue(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_get_sender(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_get_time_real(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_get_time_update(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_malloc(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_set_dest(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_set_exclusive(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_set_queue(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_set_sender(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_set_time_real(ALSA_0.9) [ALSA]
snd_seq_port_subscribe_set_time_update(ALSA_0.9) [ALSA]	snd_seq_port_subscribe_sizeof(ALSA_0.9) [ALSA]	snd_seq_query_port_subscribers(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_copy(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_free(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_addr(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_get_exclusive(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_index(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_queue(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_get_root(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_time_real(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_get_time_update(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_malloc(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_set_index(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_set_root(ALSA_0.9) [ALSA]
snd_seq_query_subscribe_set_type(ALSA_0.9) [ALSA]	snd_seq_query_subscribe_sizeof(ALSA_0.9) [ALSA]	snd_seq_subscribe_port(ALSA_0.9) [ALSA]
snd_seq_unsubscribe_port(ALSA_0.9) [ALSA]		

6.1.30 ALSA Sequencer Queue Interface

6.1.30.1 Interfaces for ALSA Sequencer Queue Interface

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer Queue Interface specified in Table 6-31, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-31 libasound - ALSA Sequencer Queue Interface Function Interfaces

snd_seq_alloc_named_queue(ALSA_0.9) [ALSA]	snd_seq_alloc_queue(ALSA_0.9) [ALSA]	snd_seq_free_queue(ALSA_0.9) [ALSA]
snd_seq_get_queue_status(ALSA_0.9) [ALSA]	snd_seq_get_queue_tempo(ALSA_0.9) [ALSA]	snd_seq_queue_status_copy(ALSA_0.9) [ALSA]

snd_seq_queue_status_free(ALSA_0.9) [ALSA]	snd_seq_queue_status_get_real_time(ALSA_0.9) [ALSA]	snd_seq_queue_status_get_tick_time(ALSA_0.9) [ALSA]
snd_seq_queue_status_malloc(ALSA_0.9) [ALSA]	snd_seq_queue_status_sizeof(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_copy(ALSA_0.9) [ALSA]
snd_seq_queue_tempo_free(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_get_ppq(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_get_tempo(ALSA_0.9) [ALSA]
snd_seq_queue_tempo_malloc(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_set_ppq(ALSA_0.9) [ALSA]	snd_seq_queue_tempo_set_tempo(ALSA_0.9) [ALSA]
snd_seq_queue_tempo_sizeof(ALSA_0.9) [ALSA]	snd_seq_set_queue_tempo(ALSA_0.9) [ALSA]	

6.1.31 ALSA Sequencer event - MIDI byte stream coder

6.1.31.1 Interfaces for ALSA Sequencer event - MIDI byte stream coder

An LSB conforming implementation shall provide the generic functions for ALSA Sequencer event - MIDI byte stream coder specified in Table 6-32, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-32 libasound - ALSA Sequencer event - MIDI byte stream coder Function Interfaces

snd_midi_event_decode(ALSA_0.9) [ALSA]	snd_midi_event_encode(ALSA_0.9) [ALSA]	snd_midi_event_encode_byte(ALSA_0.9) [ALSA]
snd_midi_event_free(ALSA_0.9) [ALSA]	snd_midi_event_init(ALSA_0.9) [ALSA]	snd_midi_event_new(ALSA_0.9) [ALSA]
snd_midi_event_reset_decode(ALSA_0.9) [ALSA]	snd_midi_event_reset_encode(ALSA_0.9) [ALSA]	

6.1.32 ALSA Simple Mixer Interface

6.1.32.1 Interfaces for ALSA Simple Mixer Interface

An LSB conforming implementation shall provide the generic functions for ALSA Simple Mixer Interface specified in Table 6-33, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-33 libasound - ALSA Simple Mixer Interface Function Interfaces

snd_mixer_find_selem(ALSA_0.9) [ALSA]	snd_mixer_selem_channel_name(ALSA_0.9) [ALSA]	snd_mixer_selem_get_capture_group(ALSA_0.9) [ALSA]
snd_mixer_selem_get_c	snd_mixer_selem_get_c	snd_mixer_selem_get_c

apture_switch(ALSA_0.9) [ALSA]	apture_volume(ALSA_0.9) [ALSA]	apture_volume_range(ALSA_0.9) [ALSA]
snd_mixer_selem_get_enum_item(ALSA_0.9) [ALSA]	snd_mixer_selem_get_enum_item_name(ALSA_0.9) [ALSA]	snd_mixer_selem_get_enum_items(ALSA_0.9) [ALSA]
snd_mixer_selem_get_id(ALSA_0.9) [ALSA]	snd_mixer_selem_get_index(ALSA_0.9) [ALSA]	snd_mixer_selem_get_name(ALSA_0.9) [ALSA]
snd_mixer_selem_get_playback_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_get_playback_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_get_playback_volume_range(ALSA_0.9) [ALSA]
snd_mixer_selem_has_capture_channel(ALSA_0.9) [ALSA]	snd_mixer_selem_has_capture_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_has_capture_switch_exclusive(ALSA_0.9) [ALSA]
snd_mixer_selem_has_capture_switch_joined(ALSA_0.9) [ALSA]	snd_mixer_selem_has_capture_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_has_capture_volume_joined(ALSA_0.9) [ALSA]
snd_mixer_selem_has_common_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_has_common_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_has_playback_channel(ALSA_0.9) [ALSA]
snd_mixer_selem_has_playback_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_has_playback_switch_joined(ALSA_0.9) [ALSA]	snd_mixer_selem_has_playback_volume(ALSA_0.9) [ALSA]
snd_mixer_selem_has_playback_volume_joined(ALSA_0.9) [ALSA]	snd_mixer_selem_id_copy(ALSA_0.9) [ALSA]	snd_mixer_selem_id_free(ALSA_0.9) [ALSA]
snd_mixer_selem_id_get_index(ALSA_0.9) [ALSA]	snd_mixer_selem_id_get_name(ALSA_0.9) [ALSA]	snd_mixer_selem_id_malloc(ALSA_0.9) [ALSA]
snd_mixer_selem_id_set_index(ALSA_0.9) [ALSA]	snd_mixer_selem_id_set_name(ALSA_0.9) [ALSA]	snd_mixer_selem_id_sizeof(ALSA_0.9) [ALSA]
snd_mixer_selem_is_active(ALSA_0.9) [ALSA]	snd_mixer_selem_is_capture_mono(ALSA_0.9) [ALSA]	snd_mixer_selem_is_enum_capture(ALSA_0.9) [ALSA]
snd_mixer_selem_is_enum_playback(ALSA_0.9) [ALSA]	snd_mixer_selem_is_enumerated(ALSA_0.9) [ALSA]	snd_mixer_selem_is_playback_mono(ALSA_0.9) [ALSA]
snd_mixer_selem_register(ALSA_0.9) [ALSA]	snd_mixer_selem_set_capture_switch(ALSA_0.9) [ALSA]	snd_mixer_selem_set_capture_switch_all(ALSA_0.9) [ALSA]
snd_mixer_selem_set_capture_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_set_capture_volume_all(ALSA_0.9) [ALSA]	snd_mixer_selem_set_capture_volume_range(ALSA_0.9) [ALSA]
snd_mixer_selem_set_enum_item(ALSA_0.9)	snd_mixer_selem_set_playback_switch(ALSA_	snd_mixer_selem_set_playback_switch_all(ALS

[ALSA]	0.9) [ALSA]	A_0.9) [ALSA]
snd_mixer_selem_set_playback_volume(ALSA_0.9) [ALSA]	snd_mixer_selem_set_playback_volume_all(ALSA_0.9) [ALSA]	snd_mixer_selem_set_playback_volume_range(ALSA_0.9) [ALSA]

6.1.33 ALSA Timer Interface

6.1.33.1 Interfaces for ALSA Timer Interface

An LSB conforming implementation shall provide the generic functions for ALSA Timer Interface specified in Table 6-34, with the full mandatory functionality as described in the referenced underlying specification.

Table 6-34 libasound - ALSA Timer Interface Function Interfaces

snd_timer_close(ALSA_0.9) [ALSA]	snd_timer_continue(ALSA_0.9) [ALSA]	snd_timer_id_copy(ALSA_0.9) [ALSA]
snd_timer_id_free(ALSA_0.9) [ALSA]	snd_timer_id_get_card(ALSA_0.9) [ALSA]	snd_timer_id_get_class(ALSA_0.9) [ALSA]
snd_timer_id_get_device(ALSA_0.9) [ALSA]	snd_timer_id_get_sclass(ALSA_0.9) [ALSA]	snd_timer_id_get_subdevice(ALSA_0.9) [ALSA]
snd_timer_id_malloc(ALSA_0.9) [ALSA]	snd_timer_id_set_card(ALSA_0.9) [ALSA]	snd_timer_id_set_class(ALSA_0.9) [ALSA]
snd_timer_id_set_device(ALSA_0.9) [ALSA]	snd_timer_id_set_sclass(ALSA_0.9) [ALSA]	snd_timer_id_set_subdevice(ALSA_0.9) [ALSA]
snd_timer_id_sizeof(ALSA_0.9) [ALSA]	snd_timer_info(ALSA_0.9) [ALSA]	snd_timer_info_copy(ALSA_0.9) [ALSA]
snd_timer_info_free(ALSA_0.9) [ALSA]	snd_timer_info_get_card(ALSA_0.9) [ALSA]	snd_timer_info_get_id(ALSA_0.9) [ALSA]
snd_timer_info_get_name(ALSA_0.9) [ALSA]	snd_timer_info_get_resolution(ALSA_0.9) [ALSA]	snd_timer_info_malloc(ALSA_0.9) [ALSA]
snd_timer_info_sizeof(ALSA_0.9) [ALSA]	snd_timer_open(ALSA_0.9) [ALSA]	snd_timer_params(ALSA_0.9) [ALSA]
snd_timer_params_get_ticks(ALSA_0.9) [ALSA]	snd_timer_params_malloc(ALSA_0.9) [ALSA]	snd_timer_params_set_auto_start(ALSA_0.9) [ALSA]
snd_timer_params_set_ticks(ALSA_0.9) [ALSA]	snd_timer_poll_descriptors(ALSA_0.9) [ALSA]	snd_timer_poll_descriptors_count(ALSA_0.9) [ALSA]
snd_timer_read(ALSA_0.9) [ALSA]	snd_timer_start(ALSA_0.9) [ALSA]	snd_timer_status(ALSA_0.9) [ALSA]
snd_timer_stop(ALSA_0.9) [ALSA]		

6.2 Data Definitions for libasound

This section defines global identifiers and their values that are associated with interfaces contained in libasound. These definitions are organized into groups that correspond to system headers. This convention is used as a convenience for the reader, and does not imply the existence of these headers, or their content. Where an interface is defined as requiring a particular system header file all of the data definitions for that system header file presented here shall be in effect.

This section gives data definitions to promote binary application portability, not to repeat source interface definitions available elsewhere. System providers and application developers should use this ABI to supplement - not to replace - source interface definition specifications.

This specification uses the ISO C (1999) C Language as the reference programming language, and data definitions are specified in ISO C format. The C language is used here as a convenient notation. Using a C language description of these data objects does not preclude their use by other programming languages.

6.2.1 alsa/conf.h

```
typedef struct _snd_config_iterator *snd_config_iterator_t;
typedef struct _snd_config snd_config_t;
typedef enum _snd_config_type {
    SND_CONFIG_TYPE_INTEGER,
    SND_CONFIG_TYPE_INTEGER64,
    SND_CONFIG_TYPE_REAL,
    SND_CONFIG_TYPE_STRING,
    SND_CONFIG_TYPE_POINTER,
    SND_CONFIG_TYPE_COMPOUND
} snd_config_type_t;
typedef struct _snd_config_update snd_config_update_t;
typedef struct snd_devname {
    char *name;
    char *comment;
    snd_devname_t *next;
} snd_devname_t;
extern int snd_config_add(snd_config_t *, snd_config_t *);
extern int snd_config_copy(snd_config_t * *, snd_config_t *);
extern int snd_config_delete(snd_config_t *);
extern int snd_config_get_ascii(const snd_config_t *, char **);
extern int snd_config_get_id(const snd_config_t *, const char
**);
extern int snd_config_get_integer64(const snd_config_t *, long
long int *);
extern int snd_config_get_integer(const snd_config_t *, long int
*);
extern int snd_config_get_string(const snd_config_t *, const char
**);
extern snd_config_type_t snd_config_get_type(const snd_config_t
*);
extern int snd_config_imake_integer64(snd_config_t * *, const
char *,
                                      const long long int);
extern int snd_config_imake_integer(snd_config_t * *, const char
*,
                                    const long int);
extern int snd_config_imake_string(snd_config_t * *, const char
*,
                                   const char *);
```

```
extern     snd_config_iterator_t     snd_config_iterator_end(const
snd_config_t *);
extern snd_config_t *snd_config_iterator_entry(const

snd_config_iterator_t);
extern    snd_config_iterator_t    snd_config_iterator_first(const
snd_config_t
                                                          *);
extern snd_config_iterator_t snd_config_iterator_next(const

snd_config_iterator_t);
extern int snd_config_load(snd_config_t *, snd_input_t *);
extern int snd_config_make_compound(snd_config_t * *, const char
*, int);
extern int snd_config_make_integer64(snd_config_t * *, const char
*);
extern int snd_config_make_integer(snd_config_t * *, const char
*);
extern int snd_config_make_string(snd_config_t * *, const char
*);
extern int snd_config_save(snd_config_t *, snd_output_t *);
extern int snd_config_search(snd_config_t *, const char *,
                             snd_config_t * *);
extern int snd_config_searchv(snd_config_t *, snd_config_t * *,
...);
extern int snd_config_set_ascii(snd_config_t *, const char *);
extern int snd_config_set_integer64(snd_config_t *, long long
int);
extern int snd_config_set_integer(snd_config_t *, long int);
extern int snd_config_set_string(snd_config_t *, const char *);
extern int snd_config_top(snd_config_t * *);
extern int snd_config_update(void);
extern int snd_config_update_free_global(void);
extern snd_config_t *snd_config;
```

6.2.2 alsa/control.h

```
#define SND_CTL_EVENT_MASK_VALUE        (1<<0)
#define SND_CTL_EVENT_MASK_INFO (1<<1)
#define SND_CTL_EVENT_MASK_ADD  (1<<2)
#define SND_CTL_EVENT_MASK_TLV  (1<<3)
#define SND_CTL_POWER_D3hot     (SND_CTL_POWER_D3|0x0000)
#define SND_CTL_POWER_D3cold    (SND_CTL_POWER_D3|0x0001)
#define SND_CTL_EVENT_MASK_REMOVE       (~0U)
#define SND_CTL_TLV_DB_GAIN_MUTE        -9999999
#define SND_CTL_POWER_D0        0x0000
#define SND_CTL_TLVT_CONTAINER  0x0000
#define SND_CTL_NONBLOCK        0x0001
#define SND_CTL_TLVT_DB_SCALE   0x0001
#define SND_SCTL_NOFREE 0x0001
#define SND_CTL_ASYNC   0x0002
#define SND_CTL_TLVT_DB_LINEAR  0x0002
#define SND_CTL_TLVT_DB_RANGE   0x0003
#define SND_CTL_READONLY        0x0004
#define SND_CTL_POWER_D1        0x0100
#define SND_CTL_POWER_D2        0x0200
#define SND_CTL_POWER_D3        0x0300
#define SND_CTL_POWER_MASK      0xff00

typedef struct snd_aes_iec958 {
    unsigned char status[24];
    unsigned char subcode[147];
    unsigned char pad;
    unsigned char dig_subframe[4];
} snd_aes_iec958_t;
```

```
typedef struct _snd_ctl_card_info snd_ctl_card_info_t;
typedef struct sndrv_ctl_elem_id snd_ctl_elem_id_t;
typedef enum _snd_ctl_elem_iface {
    SND_CTL_ELEM_IFACE_CARD,
    SND_CTL_ELEM_IFACE_HWDEP,
    SND_CTL_ELEM_IFACE_MIXER,
    SND_CTL_ELEM_IFACE_PCM,
    SND_CTL_ELEM_IFACE_RAWMIDI,
    SND_CTL_ELEM_IFACE_TIMER,
    SND_CTL_ELEM_IFACE_SEQUENCER,
    SND_CTL_ELEM_IFACE_LAST
} snd_ctl_elem_iface_t;
typedef struct _snd_ctl_elem_info snd_ctl_elem_info_t;
typedef struct sndrv_ctl_elem_list snd_ctl_elem_list_t;
typedef enum _snd_ctl_elem_type {
    SND_CTL_ELEM_TYPE_NONE,
    SND_CTL_ELEM_TYPE_BOOLEAN,
    SND_CTL_ELEM_TYPE_INTEGER,
    SND_CTL_ELEM_TYPE_ENUMERATED,
    SND_CTL_ELEM_TYPE_BYTES,
    SND_CTL_ELEM_TYPE_IEC958,
    SND_CTL_ELEM_TYPE_INTEGER64,
    SND_CTL_ELEM_TYPE_LAST
} snd_ctl_elem_type_t;
typedef struct _snd_ctl_elem_value snd_ctl_elem_value_t;
typedef struct sndrv_ctl_event snd_ctl_event_t;
typedef enum _snd_ctl_event_type {
    SND_CTL_EVENT_ELEM,
    SND_CTL_EVENT_LAST
} snd_ctl_event_type_t;
typedef struct _snd_ctl snd_ctl_t;
typedef enum _snd_ctl_type {
    SND_CTL_TYPE_HW,
    SND_CTL_TYPE_SHM,
    SND_CTL_TYPE_INET,
    SND_CTL_TYPE_EXT
} snd_ctl_type_t;
typedef struct _snd_hctl snd_hctl_t;
typedef struct _snd_sctl snd_sctl_t;
typedef struct _snd_hctl_elem snd_hctl_elem_t;
typedef int (*snd_hctl_compare_t) (void);
typedef int (*snd_hctl_callback_t) (void);
typedef int (*snd_hctl_elem_callback_t) (void);
extern  int  snd_async_add_ctl_handler(snd_async_handler_t  *  *,
snd_ctl_t *,
                                       snd_async_callback_t,    void
*);
extern  snd_ctl_t  *snd_async_handler_get_ctl(snd_async_handler_t
*);
extern int snd_card_get_index(const char *);
extern int snd_card_get_longname(int, char **);
extern int snd_card_get_name(int, char **);
extern int snd_card_load(int);
extern int snd_card_next(int *);
extern int snd_ctl_card_info(snd_ctl_t *, snd_ctl_card_info_t *);
extern void snd_ctl_card_info_clear(snd_ctl_card_info_t *);
extern void snd_ctl_card_info_copy(snd_ctl_card_info_t *,
                                   const snd_ctl_card_info_t *);
extern void snd_ctl_card_info_free(snd_ctl_card_info_t *);
extern const char *snd_ctl_card_info_get_components(const

snd_ctl_card_info_t *);
extern     const     char     *snd_ctl_card_info_get_driver(const
snd_ctl_card_info_t
                                                 *);
```

```
extern      const      char      *snd_ctl_card_info_get_id(const
snd_ctl_card_info_t *);
extern    const    char    *snd_ctl_card_info_get_longname(const
snd_ctl_card_info_t
                                                    *);
extern const char *snd_ctl_card_info_get_mixername(const

snd_ctl_card_info_t *);
extern     const      char     *snd_ctl_card_info_get_name(const
snd_ctl_card_info_t *);
extern int snd_ctl_card_info_malloc(snd_ctl_card_info_t * *);
extern size_t snd_ctl_card_info_sizeof(void);
extern int snd_ctl_close(snd_ctl_t *);
extern   int    snd_ctl_elem_add_boolean(snd_ctl_t    *,    const
snd_ctl_elem_id_t *,
                                    unsigned int);
extern    int    snd_ctl_elem_add_iec958(snd_ctl_t    *,    const
snd_ctl_elem_id_t *);
extern   int    snd_ctl_elem_add_integer(snd_ctl_t    *,    const
snd_ctl_elem_id_t *,
                                    unsigned int, long int, long
int,
                                    long int);
extern void snd_ctl_elem_id_clear(snd_ctl_elem_id_t *);
extern void snd_ctl_elem_id_copy(snd_ctl_elem_id_t *,
                                 const snd_ctl_elem_id_t *);
extern void snd_ctl_elem_id_free(snd_ctl_elem_id_t *);
extern     unsigned     int     snd_ctl_elem_id_get_device(const
snd_ctl_elem_id_t *);
extern     unsigned     int     snd_ctl_elem_id_get_index(const
snd_ctl_elem_id_t *);
extern snd_ctl_elem_iface_t snd_ctl_elem_id_get_interface(const

snd_ctl_elem_id_t
                                                        *);
extern      const      char      *snd_ctl_elem_id_get_name(const
snd_ctl_elem_id_t *);
extern     unsigned     int     snd_ctl_elem_id_get_numid(const
snd_ctl_elem_id_t *);
extern    unsigned    int    snd_ctl_elem_id_get_subdevice(const
snd_ctl_elem_id_t
                                                    *);
extern int snd_ctl_elem_id_malloc(snd_ctl_elem_id_t * *);
extern   void   snd_ctl_elem_id_set_device(snd_ctl_elem_id_t   *,
unsigned int);
extern   void   snd_ctl_elem_id_set_index(snd_ctl_elem_id_t   *,
unsigned int);
extern void snd_ctl_elem_id_set_interface(snd_ctl_elem_id_t *,
                                          snd_ctl_elem_iface_t);
extern  void  snd_ctl_elem_id_set_name(snd_ctl_elem_id_t  *,  const
char *);
extern   void   snd_ctl_elem_id_set_numid(snd_ctl_elem_id_t   *,
unsigned int);
extern void snd_ctl_elem_id_set_subdevice(snd_ctl_elem_id_t *,
                                          unsigned int);
extern size_t snd_ctl_elem_id_sizeof(void);
extern const char *snd_ctl_elem_iface_name(snd_ctl_elem_iface_t);
extern int snd_ctl_elem_info(snd_ctl_t *, snd_ctl_elem_info_t *);
extern void snd_ctl_elem_info_clear(snd_ctl_elem_info_t *);
extern void snd_ctl_elem_info_copy(snd_ctl_elem_info_t *,
                                   const snd_ctl_elem_info_t *);
extern void snd_ctl_elem_info_free(snd_ctl_elem_info_t *);
extern     unsigned     int     snd_ctl_elem_info_get_count(const
snd_ctl_elem_info_t
                                                      *);
extern void snd_ctl_elem_info_get_id(const snd_ctl_elem_info_t *,
```



```
                                        snd_ctl_elem_id_t *);
extern const char *snd_ctl_elem_info_get_item_name(const

snd_ctl_elem_info_t *);
extern     unsigned     int     snd_ctl_elem_info_get_items(const
snd_ctl_elem_info_t
                                                    *);
extern    long    long    int    snd_ctl_elem_info_get_max64(const
snd_ctl_elem_info_t
                                                     *);
extern       long       int       snd_ctl_elem_info_get_max(const
snd_ctl_elem_info_t *);
extern    long    long    int    snd_ctl_elem_info_get_min64(const
snd_ctl_elem_info_t
                                                     *);
extern       long       int       snd_ctl_elem_info_get_min(const
snd_ctl_elem_info_t *);
extern      const      char      *snd_ctl_elem_info_get_name(const
snd_ctl_elem_info_t *);
extern     unsigned     int     snd_ctl_elem_info_get_numid(const
snd_ctl_elem_info_t
                                                    *);
extern    long    long    int    snd_ctl_elem_info_get_step64(const
snd_ctl_elem_info_t
                                                      *);
extern       long       int       snd_ctl_elem_info_get_step(const
snd_ctl_elem_info_t *);
extern snd_ctl_elem_type_t snd_ctl_elem_info_get_type(const

snd_ctl_elem_info_t
                                                        *);
extern           int           snd_ctl_elem_info_is_inactive(const
snd_ctl_elem_info_t *);
extern  int  snd_ctl_elem_info_is_locked(const  snd_ctl_elem_info_t
*);
extern           int           snd_ctl_elem_info_is_readable(const
snd_ctl_elem_info_t *);
extern  int  snd_ctl_elem_info_is_user(const  snd_ctl_elem_info_t
*);
extern           int           snd_ctl_elem_info_is_volatile(const
snd_ctl_elem_info_t *);
extern           int           snd_ctl_elem_info_is_writable(const
snd_ctl_elem_info_t *);
extern int snd_ctl_elem_info_malloc(snd_ctl_elem_info_t * *);
extern void snd_ctl_elem_info_set_id(snd_ctl_elem_info_t *,
                                     const snd_ctl_elem_id_t *);
extern void snd_ctl_elem_info_set_item(snd_ctl_elem_info_t *,
                                       unsigned int);
extern size_t snd_ctl_elem_info_sizeof(void);
extern int snd_ctl_elem_list(snd_ctl_t *, snd_ctl_elem_list_t *);
extern int snd_ctl_elem_list_alloc_space(snd_ctl_elem_list_t *,
                                         unsigned int);
extern void snd_ctl_elem_list_clear(snd_ctl_elem_list_t *);
extern void snd_ctl_elem_list_copy(snd_ctl_elem_list_t *,
                                   const snd_ctl_elem_list_t *);
extern void snd_ctl_elem_list_free(snd_ctl_elem_list_t *);
extern void snd_ctl_elem_list_free_space(snd_ctl_elem_list_t *);
extern     unsigned     int     snd_ctl_elem_list_get_count(const
snd_ctl_elem_list_t
                                                    *);
extern void snd_ctl_elem_list_get_id(const snd_ctl_elem_list_t *,
                                     unsigned                int,
snd_ctl_elem_id_t *);
extern      const      char      *snd_ctl_elem_list_get_name(const
snd_ctl_elem_list_t *,
                                                unsigned int);
```

```
extern     unsigned     int     snd_ctl_elem_list_get_used(const
snd_ctl_elem_list_t
                                                *);
extern int snd_ctl_elem_list_malloc(snd_ctl_elem_list_t * *);
extern void snd_ctl_elem_list_set_offset(snd_ctl_elem_list_t *,
                                         unsigned int);
extern size_t snd_ctl_elem_list_sizeof(void);
extern  int  snd_ctl_elem_read(snd_ctl_t  *,  snd_ctl_elem_value_t
*);
extern int snd_ctl_elem_remove(snd_ctl_t *, snd_ctl_elem_id_t *);
extern const char *snd_ctl_elem_type_name(snd_ctl_elem_type_t);
extern void snd_ctl_elem_value_clear(snd_ctl_elem_value_t *);
extern void snd_ctl_elem_value_copy(snd_ctl_elem_value_t *,
                                    const     snd_ctl_elem_value_t
*);
extern void snd_ctl_elem_value_free(snd_ctl_elem_value_t *);
extern          int          snd_ctl_elem_value_get_boolean(const
snd_ctl_elem_value_t *,
                                         unsigned int);
extern     unsigned     char     snd_ctl_elem_value_get_byte(const
snd_ctl_elem_value_t
                                           *,       unsigned
int);
extern     const     void     *snd_ctl_elem_value_get_bytes(const
snd_ctl_elem_value_t
                                          *);
extern unsigned int snd_ctl_elem_value_get_enumerated(const

snd_ctl_elem_value_t
                                                *,  unsigned
int);
extern  void  snd_ctl_elem_value_get_id(const  snd_ctl_elem_value_t
*,
                                    snd_ctl_elem_id_t *);
extern          void          snd_ctl_elem_value_get_iec958(const
snd_ctl_elem_value_t *,
                                         snd_aes_iec958_t *);
extern long long int snd_ctl_elem_value_get_integer64(const

snd_ctl_elem_value_t
                                                *,  unsigned
int);
extern     long     int     snd_ctl_elem_value_get_integer(const
snd_ctl_elem_value_t
                                               *, unsigned int);
extern int snd_ctl_elem_value_malloc(snd_ctl_elem_value_t * *);
extern  void  snd_ctl_elem_value_set_boolean(snd_ctl_elem_value_t
*,
                                          unsigned   int,   long
int);
extern void snd_ctl_elem_value_set_byte(snd_ctl_elem_value_t *,
                                        unsigned   int,   unsigned
char);
extern                                                      void
snd_ctl_elem_value_set_enumerated(snd_ctl_elem_value_t *,
                                             unsigned       int,
unsigned int);
extern void snd_ctl_elem_value_set_id(snd_ctl_elem_value_t *,
                                      const snd_ctl_elem_id_t *);
extern void snd_ctl_elem_value_set_iec958(snd_ctl_elem_value_t *,
                                         const   snd_aes_iec958_t
*);
extern  void  snd_ctl_elem_value_set_integer64(snd_ctl_elem_value_t
*,
                                            unsigned   int,   long
long int);
```



```
extern  void  snd_ctl_elem_value_set_integer(snd_ctl_elem_value_t
*,
                                             unsigned   int,   long
int);
extern size_t snd_ctl_elem_value_sizeof(void);
extern  int  snd_ctl_elem_write(snd_ctl_t  *,  snd_ctl_elem_value_t
*);
extern void snd_ctl_event_clear(snd_ctl_event_t *);
extern    void    snd_ctl_event_copy(snd_ctl_event_t    *,    const
snd_ctl_event_t *);
extern void snd_ctl_event_elem_get_id(const snd_ctl_event_t *,
                                      snd_ctl_elem_id_t *);
extern     unsigned     int     snd_ctl_event_elem_get_mask(const
snd_ctl_event_t *);
extern void snd_ctl_event_free(snd_ctl_event_t *);
extern int snd_ctl_event_malloc(snd_ctl_event_t * *);
extern size_t snd_ctl_event_sizeof(void);
extern int snd_ctl_hwdep_info(snd_ctl_t *, snd_hwdep_info_t *);
extern int snd_ctl_hwdep_next_device(snd_ctl_t *, int *);
extern const char *snd_ctl_name(snd_ctl_t *);
extern int snd_ctl_nonblock(snd_ctl_t *, int);
extern int snd_ctl_open(snd_ctl_t * *, const char *, int);
extern int snd_ctl_pcm_info(snd_ctl_t *, snd_pcm_info_t *);
extern int snd_ctl_pcm_next_device(snd_ctl_t *, int *);
extern int snd_ctl_poll_descriptors(snd_ctl_t *, struct pollfd *,
                                    unsigned int);
extern int snd_ctl_poll_descriptors_count(snd_ctl_t *);
extern  int  snd_ctl_rawmidi_info(snd_ctl_t  *,  snd_rawmidi_info_t
*);
extern int snd_ctl_rawmidi_next_device(snd_ctl_t *, int *);
extern int snd_ctl_read(snd_ctl_t *, snd_ctl_event_t *);
extern int snd_ctl_subscribe_events(snd_ctl_t *, int);
extern int snd_hctl_close(snd_hctl_t *);
extern       void       *snd_hctl_elem_get_callback_private(const
snd_hctl_elem_t *);
extern void snd_hctl_elem_get_id(const snd_hctl_elem_t *,
                                 snd_ctl_elem_id_t *);
extern       int       snd_hctl_elem_info(snd_hctl_elem_t       *,
snd_ctl_elem_info_t *);
extern snd_hctl_elem_t *snd_hctl_elem_next(snd_hctl_elem_t *);
extern snd_hctl_elem_t *snd_hctl_elem_prev(snd_hctl_elem_t *);
extern       int       snd_hctl_elem_read(snd_hctl_elem_t       *,
snd_ctl_elem_value_t *);
extern void snd_hctl_elem_set_callback(snd_hctl_elem_t *,
                                       snd_hctl_elem_callback_t);
extern void snd_hctl_elem_set_callback_private(snd_hctl_elem_t *,
void *);
extern      int      snd_hctl_elem_write(snd_hctl_elem_t      *,
snd_ctl_elem_value_t *);
extern snd_hctl_elem_t *snd_hctl_find_elem(snd_hctl_t *,
                                           const
snd_ctl_elem_id_t *);
extern snd_hctl_elem_t *snd_hctl_first_elem(snd_hctl_t *);
extern int snd_hctl_free(snd_hctl_t *);
extern void *snd_hctl_get_callback_private(snd_hctl_t *);
extern snd_hctl_elem_t *snd_hctl_last_elem(snd_hctl_t *);
extern int snd_hctl_load(snd_hctl_t *);
extern int snd_hctl_open(snd_hctl_t * *, const char *, int);
extern       void       snd_hctl_set_callback(snd_hctl_t       *,
snd_hctl_callback_t);
extern void snd_hctl_set_callback_private(snd_hctl_t *, void *);
```

6.2.3 alsa/control_external.h

```
#define SND_CTL_EXT_VERSION      ((SND_CTL_EXT_VERSION_MAJOR<<16)
| (SND_CTL_EXT_VERSION_MINOR<<8) | (SND_CTL_EXT_VERSION_TINY))
#define SND_CTL_EXT_KEY_NOT_FOUND       (snd_ctl_ext_key_t)(-1)
#define SND_CTL_EXT_VERSION_MINOR       0
#define SND_CTL_EXT_VERSION_TINY        0
#define SND_CTL_EXT_VERSION_MAJOR       1

typedef struct snd_ctl_ext_callback {
    void (*close) (void);
    int (*elem_count) (void);
    int (*elem_list) (void);
     snd_ctl_ext_key_t(*find_elem) (void);
    void (*free_key) (void);
    int (*get_attribute) (void);
    int (*get_integer_info) (void);
    int (*get_integer64_info) (void);
    int (*get_enumerated_info) (void);
    int (*get_enumerated_name) (void);
    int (*read_integer) (void);
    int (*read_integer64) (void);
    int (*read_enumerated) (void);
    int (*read_bytes) (void);
    int (*read_iec958) (void);
    int (*write_integer) (void);
    int (*write_integer64) (void);
    int (*write_enumerated) (void);
    int (*write_bytes) (void);
    int (*write_iec958) (void);
    void (*subscribe_events) (void);
    int (*read_event) (void);
    int (*poll_descriptors_count) (void);
    int (*poll_descriptors) (void);
    int (*poll_revents) (void);
} snd_ctl_ext_callback_t;
typedef long unsigned int snd_ctl_ext_key_t;
typedef struct snd_ctl_ext {
    unsigned int version;
    int card_idx;
    char id[16];
    char driver[16];
    char name[32];
    char longname[80];
    char mixername[80];
    int poll_fd;
    const snd_ctl_ext_callback_t *callback;
    void *private_data;
    snd_ctl_t *handle;
    int nonblock;
    int subscribed;
} snd_ctl_ext_t;
```

6.2.4 alsa/error.h

```
#define SND_ERROR_INCOMPATIBLE_VERSION  (SND_ERROR_BEGIN+0)
#define SND_ERROR_ALISP_NIL     (SND_ERROR_BEGIN+1)
#define SND_ERROR_BEGIN 500000

typedef void (*snd_lib_error_handler_t) (void);
extern int snd_lib_error_set_handler(snd_lib_error_handler_t);
extern const char *snd_strerror(int);
```

6.2.5 alsa/global.h

```
typedef struct _snd_async_handler snd_async_handler_t;
```

```
typedef void (*snd_async_callback_t) (void);
typedef struct timespec snd_htimestamp_t;
typedef struct timeval snd_timestamp_t;
extern const char *snd_asoundlib_version(void);
extern int snd_async_add_handler(snd_async_handler_t * *, int,
                                 snd_async_callback_t, void *);
extern int snd_async_del_handler(snd_async_handler_t *);
extern                                                    void
*snd_async_handler_get_callback_private(snd_async_handler_t *);
```

6.2.6 alsa/hwdep.h

```
#define SND_HWDEP_OPEN_NONBLOCK (O_NONBLOCK)
#define SND_HWDEP_OPEN_READ     (O_RDONLY)
#define SND_HWDEP_OPEN_DUPLEX   (O_RDWR)
#define SND_HWDEP_OPEN_WRITE    (O_WRONLY)

typedef struct sndrv_hwdep_dsp_image snd_hwdep_dsp_image_t;
typedef struct sndrv_hwdep_dsp_status snd_hwdep_dsp_status_t;
typedef enum _snd_hwdep_iface {
    SND_HWDEP_IFACE_OPL2,
    SND_HWDEP_IFACE_OPL3,
    SND_HWDEP_IFACE_OPL4,
    SND_HWDEP_IFACE_SB16CSP,
    SND_HWDEP_IFACE_EMU10K1,
    SND_HWDEP_IFACE_YSS225,
    SND_HWDEP_IFACE_ICS2115,
    SND_HWDEP_IFACE_SSCAPE,
    SND_HWDEP_IFACE_VX,
    SND_HWDEP_IFACE_MIXART,
    SND_HWDEP_IFACE_USX2Y,
    SND_HWDEP_IFACE_EMUX_WAVETABLE,
    SND_HWDEP_IFACE_BLUETOOTH,
    SND_HWDEP_IFACE_USX2Y_PCM,
    SND_HWDEP_IFACE_PCXHR,
    SND_HWDEP_IFACE_SB_RC,
    SND_HWDEP_IFACE_LAST
} snd_hwdep_iface_t;
typedef struct sndrv_hwdep_info snd_hwdep_info_t;
typedef struct _snd_hwdep snd_hwdep_t;
typedef enum _snd_hwdep_type {
    SND_HWDEP_TYPE_HW,
    SND_HWDEP_TYPE_SHM,
    SND_HWDEP_TYPE_INET
} snd_hwdep_type_t;
extern int snd_hwdep_close(snd_hwdep_t *);
extern void snd_hwdep_dsp_image_copy(snd_hwdep_dsp_image_t *,
                                     const   snd_hwdep_dsp_image_t
*);
extern void snd_hwdep_dsp_image_free(snd_hwdep_dsp_image_t *);
extern const void *snd_hwdep_dsp_image_get_image(const

snd_hwdep_dsp_image_t *);
extern unsigned int snd_hwdep_dsp_image_get_index(const

snd_hwdep_dsp_image_t *);
extern         size_t         snd_hwdep_dsp_image_get_length(const
snd_hwdep_dsp_image_t
                                              *);
extern     const     char     *snd_hwdep_dsp_image_get_name(const
snd_hwdep_dsp_image_t
                                                    *);
extern int snd_hwdep_dsp_image_malloc(snd_hwdep_dsp_image_t * *);
extern   void   snd_hwdep_dsp_image_set_image(snd_hwdep_dsp_image_t
*, void *);
```

```
extern  void  snd_hwdep_dsp_image_set_index(snd_hwdep_dsp_image_t
*,
                                            unsigned int);
extern  void  snd_hwdep_dsp_image_set_length(snd_hwdep_dsp_image_t
*,
                                             size_t);
extern void snd_hwdep_dsp_image_set_name(snd_hwdep_dsp_image_t *,
                                         const char *);
extern size_t snd_hwdep_dsp_image_sizeof(void);
extern        int        snd_hwdep_dsp_load(snd_hwdep_t        *,
snd_hwdep_dsp_image_t *);
extern       int       snd_hwdep_dsp_status(snd_hwdep_t       *,
snd_hwdep_dsp_status_t *);
extern void snd_hwdep_dsp_status_copy(snd_hwdep_dsp_status_t *,
                                      const
snd_hwdep_dsp_status_t *);
extern void snd_hwdep_dsp_status_free(snd_hwdep_dsp_status_t *);
extern unsigned int snd_hwdep_dsp_status_get_chip_ready(const

snd_hwdep_dsp_status_t
                                                        *);
extern unsigned int snd_hwdep_dsp_status_get_dsp_loaded(const

snd_hwdep_dsp_status_t
                                                        *);
extern     const     char     *snd_hwdep_dsp_status_get_id(const
snd_hwdep_dsp_status_t
                                                       *);
extern unsigned int snd_hwdep_dsp_status_get_num_dsps(const

snd_hwdep_dsp_status_t
                                                         *);
extern unsigned int snd_hwdep_dsp_status_get_version(const

snd_hwdep_dsp_status_t
                                                       *);
extern  int  snd_hwdep_dsp_status_malloc(snd_hwdep_dsp_status_t  *
*);
extern size_t snd_hwdep_dsp_status_sizeof(void);
extern int snd_hwdep_info(snd_hwdep_t *, snd_hwdep_info_t *);
extern void snd_hwdep_info_copy(snd_hwdep_info_t *,
                                const snd_hwdep_info_t *);
extern void snd_hwdep_info_free(snd_hwdep_info_t *);
extern int snd_hwdep_info_get_card(const snd_hwdep_info_t *);
extern     unsigned     int     snd_hwdep_info_get_device(const
snd_hwdep_info_t *);
extern  const  char  *snd_hwdep_info_get_id(const  snd_hwdep_info_t
*);
extern      snd_hwdep_iface_t      snd_hwdep_info_get_iface(const
snd_hwdep_info_t
                                                     *);
extern const char *snd_hwdep_info_get_name(const snd_hwdep_info_t
*);
extern int snd_hwdep_info_malloc(snd_hwdep_info_t * *);
extern    void    snd_hwdep_info_set_device(snd_hwdep_info_t    *,
unsigned int);
extern size_t snd_hwdep_info_sizeof(void);
extern int snd_hwdep_ioctl(snd_hwdep_t *, unsigned int, void *);
extern int snd_hwdep_open(snd_hwdep_t * *, const char *, int);
extern   int   snd_hwdep_poll_descriptors(snd_hwdep_t   *,   struct
pollfd *,
                                          unsigned int);
extern ssize_t snd_hwdep_read(snd_hwdep_t *, void *, size_t);
extern  ssize_t  snd_hwdep_write(snd_hwdep_t  *,  const  void  *,
size_t);
```


6.2.7 alsa/iatomic.h

```
#define atomic_set(v,i) (((v)->counter) = (i))
#define atomic_read(v)  ((v)->counter)
#define ATOMIC_INIT(i)  { (i) }

typedef struct {
    unsigned int begin;
    unsigned int end;
} snd_atomic_write_t;
typedef struct {
    const volatile snd_atomic_write_t *write;
    unsigned int end;
} snd_atomic_read_t;
```

6.2.8 alsa/input.h

```
typedef struct _snd_input snd_input_t;
extern int snd_input_buffer_open(snd_input_t * *, const char *,
ssize_t);
extern int snd_input_close(snd_input_t *);
extern int snd_input_stdio_attach(snd_input_t * *, FILE *, int);
extern int snd_input_stdio_open(snd_input_t * *, const char *,
                                const char *);
```

6.2.9 alsa/instr.h

```
#define SND_SEQ_INSTR_ID_DLS1   "DLS1"
#define SND_SEQ_INSTR_ID_DLS2   "DLS2"
#define SND_SEQ_INSTR_ID_GUS_PATCH      "GUS Patch"
#define SND_SEQ_INSTR_ID_INTERWAVE      "Interwave FFFF"
#define SND_SEQ_INSTR_ID_OPL2_3 "OPL2/3 FM"
#define SND_SEQ_INSTR_ID_OPL4   "OPL4"
#define SND_SEQ_INSTR_ID_SIMPLE "Simple Wave"
#define SND_SEQ_INSTR_ID_SOUNDFONT      "SoundFont"
#define SND_SEQ_INSTR_QUERY_FOLLOW_ALIAS        (1<<0)
#define SND_SEQ_INSTR_TYPE0_DLS1        (1<<0)
#define SND_SEQ_INSTR_TYPE1_SIMPLE      (1<<0)
#define SND_SEQ_INSTR_TYPE2_OPL2_3      (1<<0)
#define SND_SEQ_INSTR_TYPE0_DLS2        (1<<1)
#define SND_SEQ_INSTR_TYPE1_SOUNDFONT   (1<<1)
#define SND_SEQ_INSTR_TYPE2_OPL4        (1<<1)
#define SND_SEQ_INSTR_TYPE1_GUS_PATCH   (1<<2)
#define SND_SEQ_INSTR_TYPE1_INTERWAVE   (1<<3)
#define SND_SEQ_INSTR_ATYPE_DATA        0
#define SND_SEQ_INSTR_FREE_CMD_ALL      0
#define SND_SEQ_INSTR_GET_CMD_FULL      0
#define SND_SEQ_INSTR_PUT_CMD_CREATE    0
#define SND_SEQ_INSTR_ATYPE_ALIAS       1
#define SND_SEQ_INSTR_FREE_CMD_PRIVATE  1
#define SND_SEQ_INSTR_GET_CMD_PARTIAL   1
#define SND_SEQ_INSTR_PUT_CMD_REPLACE   1
#define SND_SEQ_INSTR_FREE_CMD_CLUSTER  2
#define SND_SEQ_INSTR_PUT_CMD_MODIFY    2
#define SND_SEQ_INSTR_FREE_CMD_SINGLE   3
#define SND_SEQ_INSTR_PUT_CMD_ADD       3
#define SND_SEQ_INSTR_PUT_CMD_REMOVE    4

typedef void snd_instr_fm_t;
typedef struct sndrv_seq_instr_header snd_instr_header_t;
typedef void snd_instr_iwffff_t;
typedef void snd_instr_simple_t;
```

```
typedef struct _snd_iwffff_handle snd_iwffff_handle_t;
```

6.2.10 alsa/mixer.h

```
typedef struct _snd_mixer snd_mixer_t;
typedef struct _snd_mixer_elem snd_mixer_elem_t;
typedef enum _snd_mixer_elem_type {
    SND_MIXER_ELEM_SIMPLE,
    SND_MIXER_ELEM_LAST
} snd_mixer_elem_type_t;
typedef struct _snd_mixer_class snd_mixer_class_t;
typedef int (*snd_mixer_compare_t) (void);
typedef int (*snd_mixer_elem_callback_t) (void);
typedef int (*snd_mixer_callback_t) (void);
typedef int (*snd_mixer_event_t) (void);
typedef enum _snd_mixer_selem_channel_id {
    SND_MIXER_SCHN_UNKNOWN,
    SND_MIXER_SCHN_FRONT_LEFT,
    SND_MIXER_SCHN_FRONT_RIGHT,
    SND_MIXER_SCHN_REAR_LEFT,
    SND_MIXER_SCHN_REAR_RIGHT,
    SND_MIXER_SCHN_FRONT_CENTER,
    SND_MIXER_SCHN_WOOFER,
    SND_MIXER_SCHN_SIDE_LEFT,
    SND_MIXER_SCHN_SIDE_RIGHT,
    SND_MIXER_SCHN_REAR_CENTER,
    SND_MIXER_SCHN_LAST,
    SND_MIXER_SCHN_MONO
} snd_mixer_selem_channel_id_t;
typedef struct _snd_mixer_selem_id snd_mixer_selem_id_t;
enum snd_mixer_selem_regopt_abstract {
    SND_MIXER_SABSTRACT_NONE,
    SND_MIXER_SABSTRACT_BASIC
};
struct snd_mixer_selem_regopt {
    int ver;
    enum snd_mixer_selem_regopt_abstract abstract;
    const char *device;
    snd_pcm_t *playback_pcm;
    snd_pcm_t *capture_pcm;
};
extern int snd_mixer_attach(snd_mixer_t *, const char *);
extern int snd_mixer_close(snd_mixer_t *);
extern int snd_mixer_detach(snd_mixer_t *, const char *);
extern       void       *snd_mixer_elem_get_callback_private(const
snd_mixer_elem_t *);
extern     snd_mixer_elem_type_t     snd_mixer_elem_get_type(const
snd_mixer_elem_t
                                                          *);
extern snd_mixer_elem_t *snd_mixer_elem_next(snd_mixer_elem_t *);
extern snd_mixer_elem_t *snd_mixer_elem_prev(snd_mixer_elem_t *);
extern void snd_mixer_elem_set_callback(snd_mixer_elem_t *,

snd_mixer_elem_callback_t);
extern  void  snd_mixer_elem_set_callback_private(snd_mixer_elem_t
*,
                                                  void *);
extern snd_mixer_elem_t *snd_mixer_find_selem(snd_mixer_t *,
                                              const
snd_mixer_selem_id_t
                                              *);
extern snd_mixer_elem_t *snd_mixer_first_elem(snd_mixer_t *);
extern void snd_mixer_free(snd_mixer_t *);
extern void *snd_mixer_get_callback_private(const snd_mixer_t *);
extern unsigned int snd_mixer_get_count(const snd_mixer_t *);
```



```
extern int snd_mixer_handle_events(snd_mixer_t *);
extern snd_mixer_elem_t *snd_mixer_last_elem(snd_mixer_t *);
extern int snd_mixer_load(snd_mixer_t *);
extern int snd_mixer_open(snd_mixer_t * *, int);
extern  int  snd_mixer_poll_descriptors(snd_mixer_t  *,  struct
pollfd *,
                                        unsigned int);
extern int snd_mixer_poll_descriptors_count(snd_mixer_t *);
extern int snd_mixer_poll_descriptors_revents(snd_mixer_t *,
                                              struct pollfd *,
                                              unsigned int,
                                              short  unsigned  int
*);
extern const char
    *snd_mixer_selem_channel_name(snd_mixer_selem_channel_id_t);
extern int snd_mixer_selem_get_capture_group(snd_mixer_elem_t *);
extern int snd_mixer_selem_get_capture_switch(snd_mixer_elem_t *,

snd_mixer_selem_channel_id_t,
                                              int *);
extern int snd_mixer_selem_get_capture_volume(snd_mixer_elem_t *,

snd_mixer_selem_channel_id_t,
                                              long int *);
extern                                                    int
snd_mixer_selem_get_capture_volume_range(snd_mixer_elem_t *,
                                                    long int *,
                                                    long int *);
extern int snd_mixer_selem_get_enum_item(snd_mixer_elem_t *,

snd_mixer_selem_channel_id_t,
                                         unsigned int *);
extern int snd_mixer_selem_get_enum_item_name(snd_mixer_elem_t *,
                                              unsigned       int,
size_t,
                                              char *);
extern int snd_mixer_selem_get_enum_items(snd_mixer_elem_t *);
extern void snd_mixer_selem_get_id(snd_mixer_elem_t *,
                                   snd_mixer_selem_id_t *);
extern  unsigned  int  snd_mixer_selem_get_index(snd_mixer_elem_t
*);
extern const char *snd_mixer_selem_get_name(snd_mixer_elem_t *);
extern  int  snd_mixer_selem_get_playback_switch(snd_mixer_elem_t
*,

snd_mixer_selem_channel_id_t,
                                                 int *);
extern  int  snd_mixer_selem_get_playback_volume(snd_mixer_elem_t
*,

snd_mixer_selem_channel_id_t,
                                                 long int *);
extern                                                    int
snd_mixer_selem_get_playback_volume_range(snd_mixer_elem_t *,
                                                      long int *,
                                                      long int *);
extern  int  snd_mixer_selem_has_capture_channel(snd_mixer_elem_t
*,

snd_mixer_selem_channel_id_t);
extern   int   snd_mixer_selem_has_capture_switch(snd_mixer_elem_t
*);
extern                                                    int
snd_mixer_selem_has_capture_switch_exclusive(snd_mixer_elem_t
                                                        *);
```

```
extern                                                          int
snd_mixer_selem_has_capture_switch_joined(snd_mixer_elem_t *);
extern   int   snd_mixer_selem_has_capture_volume(snd_mixer_elem_t
*);
extern                                                          int
snd_mixer_selem_has_capture_volume_joined(snd_mixer_elem_t *);
extern int snd_mixer_selem_has_common_switch(snd_mixer_elem_t *);
extern int snd_mixer_selem_has_common_volume(snd_mixer_elem_t *);
extern  int  snd_mixer_selem_has_playback_channel(snd_mixer_elem_t
*,

snd_mixer_selem_channel_id_t);
extern  int  snd_mixer_selem_has_playback_switch(snd_mixer_elem_t
*);
extern                                                          int
snd_mixer_selem_has_playback_switch_joined(snd_mixer_elem_t *);
extern  int  snd_mixer_selem_has_playback_volume(snd_mixer_elem_t
*);
extern                                                          int
snd_mixer_selem_has_playback_volume_joined(snd_mixer_elem_t *);
extern void snd_mixer_selem_id_copy(snd_mixer_selem_id_t *,
                                    const    snd_mixer_selem_id_t
*);
extern void snd_mixer_selem_id_free(snd_mixer_selem_id_t *);
extern     unsigned     int     snd_mixer_selem_id_get_index(const
snd_mixer_selem_id_t
                                                *);
extern     const     char     *snd_mixer_selem_id_get_name(const
snd_mixer_selem_id_t
                                               *);
extern int snd_mixer_selem_id_malloc(snd_mixer_selem_id_t * *);
extern void snd_mixer_selem_id_set_index(snd_mixer_selem_id_t *,
                                         unsigned int);
extern void snd_mixer_selem_id_set_name(snd_mixer_selem_id_t *,
                                        const char *);
extern size_t snd_mixer_selem_id_sizeof(void);
extern int snd_mixer_selem_is_active(snd_mixer_elem_t *);
extern int snd_mixer_selem_is_capture_mono(snd_mixer_elem_t *);
extern int snd_mixer_selem_is_enum_capture(snd_mixer_elem_t *);
extern int snd_mixer_selem_is_enum_playback(snd_mixer_elem_t *);
extern int snd_mixer_selem_is_enumerated(snd_mixer_elem_t *);
extern int snd_mixer_selem_is_playback_mono(snd_mixer_elem_t *);
extern int snd_mixer_selem_register(snd_mixer_t *,
                                    struct  snd_mixer_selem_regopt
*,
                                    snd_mixer_class_t * *);
extern int snd_mixer_selem_set_capture_switch(snd_mixer_elem_t *,

snd_mixer_selem_channel_id_t,
                                              int);
extern                                                          int
snd_mixer_selem_set_capture_switch_all(snd_mixer_elem_t *, int);
extern int snd_mixer_selem_set_capture_volume(snd_mixer_elem_t *,

snd_mixer_selem_channel_id_t,
                                              long int);
extern                                                          int
snd_mixer_selem_set_capture_volume_all(snd_mixer_elem_t *,
                                                   long int);
extern                                                          int
snd_mixer_selem_set_capture_volume_range(snd_mixer_elem_t *,
                                                     long     int,
long int);
extern int snd_mixer_selem_set_enum_item(snd_mixer_elem_t *,

snd_mixer_selem_channel_id_t,
```



```
                                          unsigned int);
extern  int  snd_mixer_selem_set_playback_switch(snd_mixer_elem_t
*,

snd_mixer_selem_channel_id_t,
                                                int);
extern                                                    int
snd_mixer_selem_set_playback_switch_all(snd_mixer_elem_t *,
                                                  int);
extern  int  snd_mixer_selem_set_playback_volume(snd_mixer_elem_t
*,

snd_mixer_selem_channel_id_t,
                                                long int);
extern                                                    int
snd_mixer_selem_set_playback_volume_all(snd_mixer_elem_t *,
                                                  long int);
extern                                                    int
snd_mixer_selem_set_playback_volume_range(snd_mixer_elem_t *,
                                                    long    int,
long int);
extern      void      snd_mixer_set_callback(snd_mixer_t      *,
snd_mixer_callback_t);
extern  void  snd_mixer_set_callback_private(snd_mixer_t  *,  void
*);
extern int snd_mixer_wait(snd_mixer_t *, int);
```

6.2.11 alsa/mixer_abst.h

```
#define sm_selem(x)     ((sm_selem_t *)((x)->private_data))
#define sm_selem_ops(x) ((sm_selem_t *)((x)->private_data))->ops
#define SM_CAP_GVOLUME  (1<<1)
#define SM_CAP_CSWITCH_JOIN     (1<<10)
#define SM_CAP_CSWITCH_EXCL     (1<<11)
#define SM_CAP_PENUM    (1<<12)
#define SM_CAP_CENUM    (1<<13)
#define SM_CAP_GSWITCH  (1<<2)
#define SM_CAP_PVOLUME  (1<<3)
#define SM_CAP_PVOLUME_JOIN     (1<<4)
#define SM_CAP_PSWITCH  (1<<5)
#define SM_CAP_PSWITCH_JOIN     (1<<6)
#define SM_CAP_CVOLUME  (1<<7)
#define SM_CAP_CVOLUME_JOIN     (1<<8)
#define SM_CAP_CSWITCH  (1<<9)
#define SM_OPS_IS_ACTIVE        0
#define SM_OPS_IS_MONO  1
#define SM_OPS_IS_CHANNEL       2
#define SM_OPS_IS_ENUMERATED    3
#define SM_OPS_IS_ENUMCNT       4

typedef struct _sm_class_basic {
    char *device;
    snd_ctl_t *ctl;
    snd_hctl_t *hctl;
    snd_ctl_card_info_t *info;
} sm_class_basic_t;
```

6.2.12 alsa/output.h

```
typedef struct _snd_output snd_output_t;
extern int snd_output_buffer_open(snd_output_t * *);
extern size_t snd_output_buffer_string(snd_output_t *, char **);
extern int snd_output_close(snd_output_t *);
extern int snd_output_putc(snd_output_t *, int);
```

```
extern int snd_output_puts(snd_output_t *, const char *);
extern  int  snd_output_stdio_attach(snd_output_t  *  *,  FILE  *,
int);
extern int snd_output_stdio_open(snd_output_t * *, const char *,
                                 const char *);
```

6.2.13 alsa/pcm.h

```
#define SND_PCM_NONBLOCK        0x0001
#define SND_PCM_ASYNC   0x0002

typedef struct sndrv_mask snd_pcm_access_mask_t;
typedef enum _snd_pcm_access {
    SND_PCM_ACCESS_MMAP_INTERLEAVED,
    SND_PCM_ACCESS_MMAP_NONINTERLEAVED,
    SND_PCM_ACCESS_MMAP_COMPLEX,
    SND_PCM_ACCESS_RW_INTERLEAVED,
    SND_PCM_ACCESS_RW_NONINTERLEAVED,
    SND_PCM_ACCESS_LAST
} snd_pcm_access_t;
typedef struct _snd_pcm_channel_area {
    void *addr;
    unsigned int first;
    unsigned int step;
} snd_pcm_channel_area_t;
typedef enum _snd_pcm_class {
    SND_PCM_CLASS_GENERIC,
    SND_PCM_CLASS_MULTI,
    SND_PCM_CLASS_MODEM,
    SND_PCM_CLASS_DIGITIZER,
    SND_PCM_CLASS_LAST
} snd_pcm_class_t;
typedef struct sndrv_mask snd_pcm_format_mask_t;
typedef enum _snd_pcm_format {
    SND_PCM_FORMAT_UNKNOWN,
    SND_PCM_FORMAT_S8,
    SND_PCM_FORMAT_U8,
    SND_PCM_FORMAT_S16_LE,
    SND_PCM_FORMAT_S16_BE,
    SND_PCM_FORMAT_U16_LE,
    SND_PCM_FORMAT_U16_BE,
    SND_PCM_FORMAT_S24_LE,
    SND_PCM_FORMAT_S24_BE,
    SND_PCM_FORMAT_U24_LE,
    SND_PCM_FORMAT_U24_BE,
    SND_PCM_FORMAT_S32_LE,
    SND_PCM_FORMAT_S32_BE,
    SND_PCM_FORMAT_U32_LE,
    SND_PCM_FORMAT_U32_BE,
    SND_PCM_FORMAT_FLOAT_LE,
    SND_PCM_FORMAT_FLOAT_BE,
    SND_PCM_FORMAT_FLOAT64_LE,
    SND_PCM_FORMAT_FLOAT64_BE,
    SND_PCM_FORMAT_IEC958_SUBFRAME_LE,
    SND_PCM_FORMAT_IEC958_SUBFRAME_BE,
    SND_PCM_FORMAT_MU_LAW,
    SND_PCM_FORMAT_A_LAW,
    SND_PCM_FORMAT_IMA_ADPCM,
    SND_PCM_FORMAT_MPEG,
    SND_PCM_FORMAT_GSM,
    SND_PCM_FORMAT_SPECIAL,
    SND_PCM_FORMAT_S24_3LE,
    SND_PCM_FORMAT_S24_3BE,
    SND_PCM_FORMAT_U24_3LE,
    SND_PCM_FORMAT_U24_3BE,
```

```
    SND_PCM_FORMAT_S20_3LE,
    SND_PCM_FORMAT_S20_3BE,
    SND_PCM_FORMAT_U20_3LE,
    SND_PCM_FORMAT_U20_3BE,
    SND_PCM_FORMAT_S18_3LE,
    SND_PCM_FORMAT_S18_3BE,
    SND_PCM_FORMAT_U18_3LE,
    SND_PCM_FORMAT_U18_3BE,
    SND_PCM_FORMAT_LAST,
    SND_PCM_FORMAT_S16,
    SND_PCM_FORMAT_U16,
    SND_PCM_FORMAT_S24,
    SND_PCM_FORMAT_U24,
    SND_PCM_FORMAT_S32,
    SND_PCM_FORMAT_U32,
    SND_PCM_FORMAT_FLOAT,
    SND_PCM_FORMAT_FLOAT64,
    SND_PCM_FORMAT_IEC958_SUBFRAME
} snd_pcm_format_t;
typedef struct _snd_pcm_hook snd_pcm_hook_t;
typedef int (*snd_pcm_hook_func_t) (void);
typedef enum _snd_pcm_hook_type {
    SND_PCM_HOOK_TYPE_HW_PARAMS,
    SND_PCM_HOOK_TYPE_HW_FREE,
    SND_PCM_HOOK_TYPE_CLOSE,
    SND_PCM_HOOK_TYPE_LAST
} snd_pcm_hook_type_t;
typedef struct sndrv_pcm_hw_params snd_pcm_hw_params_t;
typedef struct sndrv_pcm_info snd_pcm_info_t;
typedef struct _snd_pcm_scope_ops {
    int (*enable) (void);
    void (*disable) (void);
    void (*start) (void);
    void (*stop) (void);
    void (*update) (void);
    void (*reset) (void);
    void (*close) (void);
} snd_pcm_scope_ops_t;
typedef struct _snd_pcm_scope snd_pcm_scope_t;
typedef long int snd_pcm_sframes_t;
typedef enum _snd_pcm_start {
    SND_PCM_START_DATA,
    SND_PCM_START_EXPLICIT,
    SND_PCM_START_LAST
} snd_pcm_start_t;
typedef enum _snd_pcm_state {
    SND_PCM_STATE_OPEN,
    SND_PCM_STATE_SETUP,
    SND_PCM_STATE_PREPARED,
    SND_PCM_STATE_RUNNING,
    SND_PCM_STATE_XRUN,
    SND_PCM_STATE_DRAINING,
    SND_PCM_STATE_PAUSED,
    SND_PCM_STATE_SUSPENDED,
    SND_PCM_STATE_DISCONNECTED,
    SND_PCM_STATE_LAST
} snd_pcm_state_t;
typedef struct sndrv_pcm_status snd_pcm_status_t;
typedef enum _snd_pcm_stream {
    SND_PCM_STREAM_PLAYBACK,
    SND_PCM_STREAM_CAPTURE,
    SND_PCM_STREAM_LAST
} snd_pcm_stream_t;
typedef enum _snd_pcm_subclass {
    SND_PCM_SUBCLASS_GENERIC_MIX,
    SND_PCM_SUBCLASS_MULTI_MIX,
```

```
    SND_PCM_SUBCLASS_LAST
} snd_pcm_subclass_t;
typedef struct sndrv_mask snd_pcm_subformat_mask_t;
typedef enum _snd_pcm_subformat {
    SND_PCM_SUBFORMAT_STD,
    SND_PCM_SUBFORMAT_LAST
} snd_pcm_subformat_t;
typedef struct sndrv_pcm_sw_params snd_pcm_sw_params_t;
typedef union _snd_pcm_sync_id {
    unsigned char id[16];
    short unsigned int id16[8];
    unsigned int id32[4];
} snd_pcm_sync_id_t;
typedef struct _snd_pcm snd_pcm_t;
typedef enum _snd_pcm_tstamp {
    SND_PCM_TSTAMP_NONE,
    SND_PCM_TSTAMP_MMAP,
    SND_PCM_TSTAMP_LAST
} snd_pcm_tstamp_t;
typedef enum _snd_pcm_type {
    SND_PCM_TYPE_HW,
    SND_PCM_TYPE_HOOKS,
    SND_PCM_TYPE_MULTI,
    SND_PCM_TYPE_FILE,
    SND_PCM_TYPE_NULL,
    SND_PCM_TYPE_SHM,
    SND_PCM_TYPE_INET,
    SND_PCM_TYPE_COPY,
    SND_PCM_TYPE_LINEAR,
    SND_PCM_TYPE_ALAW,
    SND_PCM_TYPE_MULAW,
    SND_PCM_TYPE_ADPCM,
    SND_PCM_TYPE_RATE,
    SND_PCM_TYPE_ROUTE,
    SND_PCM_TYPE_PLUG,
    SND_PCM_TYPE_SHARE,
    SND_PCM_TYPE_METER,
    SND_PCM_TYPE_MIX,
    SND_PCM_TYPE_DROUTE,
    SND_PCM_TYPE_LBSERVER,
    SND_PCM_TYPE_LINEAR_FLOAT,
    SND_PCM_TYPE_LADSPA,
    SND_PCM_TYPE_DMIX,
    SND_PCM_TYPE_JACK,
    SND_PCM_TYPE_DSNOOP,
    SND_PCM_TYPE_DSHARE,
    SND_PCM_TYPE_IEC958,
    SND_PCM_TYPE_SOFTVOL,
    SND_PCM_TYPE_IOPLUG,
    SND_PCM_TYPE_EXTPLUG,
    SND_PCM_TYPE_LAST
} snd_pcm_type_t;
typedef long unsigned int snd_pcm_uframes_t;
typedef enum _snd_pcm_xrun {
    SND_PCM_XRUN_NONE,
    SND_PCM_XRUN_STOP,
    SND_PCM_XRUN_LAST
} snd_pcm_xrun_t;
typedef enum _snd_spcm_duplex_type {
    SND_SPCM_DUPLEX_LIBERAL,
    SND_SPCM_DUPLEX_PEDANTIC
} snd_spcm_duplex_type_t;
typedef enum _snd_spcm_latency {
    SND_SPCM_LATENCY_STANDARD,
    SND_SPCM_LATENCY_MEDIUM,
    SND_SPCM_LATENCY_REALTIME
```

```
} snd_spcm_latency_t;
typedef enum _snd_spcm_xrun_type {
    SND_SPCM_XRUN_IGNORE,
    SND_SPCM_XRUN_STOP
} snd_spcm_xrun_type_t;
extern  int  snd_async_add_pcm_handler(snd_async_handler_t  *  *,
snd_pcm_t *,
                                       snd_async_callback_t,    void
*);
extern  snd_pcm_t  *snd_async_handler_get_pcm(snd_async_handler_t
*);
extern void snd_pcm_access_mask_any(snd_pcm_access_mask_t *);
extern void snd_pcm_access_mask_copy(snd_pcm_access_mask_t *,
                                       const   snd_pcm_access_mask_t
*);
extern void snd_pcm_access_mask_free(snd_pcm_access_mask_t *);
extern int snd_pcm_access_mask_malloc(snd_pcm_access_mask_t * *);
extern void snd_pcm_access_mask_none(snd_pcm_access_mask_t *);
extern void snd_pcm_access_mask_set(snd_pcm_access_mask_t *,
                                       snd_pcm_access_t);
extern size_t snd_pcm_access_mask_sizeof(void);
extern  int  snd_pcm_access_mask_test(const  snd_pcm_access_mask_t
*,
                                       snd_pcm_access_t);
extern const char *snd_pcm_access_name(snd_pcm_access_t);
extern int snd_pcm_area_copy(const snd_pcm_channel_area_t *,
                             snd_pcm_uframes_t,
                             const snd_pcm_channel_area_t *,
                             snd_pcm_uframes_t, unsigned int,
                             snd_pcm_format_t);
extern int snd_pcm_area_silence(const snd_pcm_channel_area_t *,
                                snd_pcm_uframes_t, unsigned int,
                                snd_pcm_format_t);
extern int snd_pcm_areas_copy(const snd_pcm_channel_area_t *,
                              snd_pcm_uframes_t,
                              const snd_pcm_channel_area_t *,
                              snd_pcm_uframes_t, unsigned int,
                              snd_pcm_uframes_t,
snd_pcm_format_t);
extern int snd_pcm_areas_silence(const snd_pcm_channel_area_t *,
                                 snd_pcm_uframes_t, unsigned int,
                                 snd_pcm_uframes_t,
snd_pcm_format_t);
extern snd_pcm_sframes_t snd_pcm_avail_update(snd_pcm_t *);
extern  snd_pcm_format_t  snd_pcm_build_linear_format(int,  int,
int, int);
extern  snd_pcm_sframes_t  snd_pcm_bytes_to_frames(snd_pcm_t  *,
ssize_t);
extern long int snd_pcm_bytes_to_samples(snd_pcm_t *, ssize_t);
extern int snd_pcm_close(snd_pcm_t *);
extern int snd_pcm_delay(snd_pcm_t *, snd_pcm_sframes_t *);
extern int snd_pcm_drain(snd_pcm_t *);
extern int snd_pcm_drop(snd_pcm_t *);
extern int snd_pcm_dump(snd_pcm_t *, snd_output_t *);
extern int snd_pcm_format_big_endian(snd_pcm_format_t);
extern int snd_pcm_format_cpu_endian(snd_pcm_format_t);
extern const char *snd_pcm_format_description(snd_pcm_format_t);
extern int snd_pcm_format_float(snd_pcm_format_t);
extern int snd_pcm_format_linear(snd_pcm_format_t);
extern int snd_pcm_format_little_endian(snd_pcm_format_t);
extern void snd_pcm_format_mask_any(snd_pcm_format_mask_t *);
extern void snd_pcm_format_mask_copy(snd_pcm_format_mask_t *,
                                       const   snd_pcm_format_mask_t
*);
extern void snd_pcm_format_mask_free(snd_pcm_format_mask_t *);
extern int snd_pcm_format_mask_malloc(snd_pcm_format_mask_t * *);
```

```
extern void snd_pcm_format_mask_none(snd_pcm_format_mask_t *);
extern void snd_pcm_format_mask_set(snd_pcm_format_mask_t *,
                                    snd_pcm_format_t);
extern size_t snd_pcm_format_mask_sizeof(void);
extern  int  snd_pcm_format_mask_test(const  snd_pcm_format_mask_t
*,
                                    snd_pcm_format_t);
extern const char *snd_pcm_format_name(snd_pcm_format_t);
extern int snd_pcm_format_physical_width(snd_pcm_format_t);
extern int snd_pcm_format_set_silence(snd_pcm_format_t, void *,
                                     unsigned int);
extern int snd_pcm_format_signed(snd_pcm_format_t);
extern ssize_t snd_pcm_format_size(snd_pcm_format_t, size_t);
extern int snd_pcm_format_unsigned(snd_pcm_format_t);
extern snd_pcm_format_t snd_pcm_format_value(const char *);
extern int snd_pcm_format_width(snd_pcm_format_t);
extern      ssize_t      snd_pcm_frames_to_bytes(snd_pcm_t      *,
snd_pcm_sframes_t);
extern int snd_pcm_hw_free(snd_pcm_t *);
extern int snd_pcm_hw_params(snd_pcm_t *, snd_pcm_hw_params_t *);
extern int snd_pcm_hw_params_any(snd_pcm_t *, snd_pcm_hw_params_t
*);
extern int snd_pcm_hw_params_can_mmap_sample_resolution(const

snd_pcm_hw_params_t
                                                        *);
extern  int  snd_pcm_hw_params_can_pause(const  snd_pcm_hw_params_t
*);
extern int snd_pcm_hw_params_can_resume(const snd_pcm_hw_params_t
*);
extern          int          snd_pcm_hw_params_can_sync_start(const
snd_pcm_hw_params_t *);
extern void snd_pcm_hw_params_copy(snd_pcm_hw_params_t *,
                                   const snd_pcm_hw_params_t *);
extern       int       snd_pcm_hw_params_current(snd_pcm_t       *,
snd_pcm_hw_params_t *);
extern    int    snd_pcm_hw_params_dump(snd_pcm_hw_params_t    *,
snd_output_t *);
extern void snd_pcm_hw_params_free(snd_pcm_hw_params_t *);
extern  int  snd_pcm_hw_params_get_access_mask(snd_pcm_hw_params_t
*,

snd_pcm_access_mask_t *);
extern  void  snd_pcm_hw_params_get_format_mask(snd_pcm_hw_params_t
*,

snd_pcm_format_mask_t *);
extern         int         snd_pcm_hw_params_get_rate_numden(const
snd_pcm_hw_params_t *,
                                            unsigned int *,
                                            unsigned int *);
extern int snd_pcm_hw_params_get_rate_resample(snd_pcm_t *,

snd_pcm_hw_params_t *,
                                              unsigned int *);
extern  int  snd_pcm_hw_params_get_sbits(const  snd_pcm_hw_params_t
*);
extern  int  snd_pcm_hw_params_is_double(const  snd_pcm_hw_params_t
*);
extern          int          snd_pcm_hw_params_is_half_duplex(const
snd_pcm_hw_params_t *);
extern         int         snd_pcm_hw_params_is_joint_duplex(const
snd_pcm_hw_params_t *);
extern int snd_pcm_hw_params_malloc(snd_pcm_hw_params_t * *);
extern      int      snd_pcm_hw_params_set_access(snd_pcm_t      *,
snd_pcm_hw_params_t *,
```

```
                                            snd_pcm_access_t);
extern int snd_pcm_hw_params_set_access_mask(snd_pcm_t *,
                                             snd_pcm_hw_params_t
*,

snd_pcm_access_mask_t *);
extern int snd_pcm_hw_params_set_buffer_size(snd_pcm_t *,
                                             snd_pcm_hw_params_t
*,
                                             snd_pcm_uframes_t);
extern int snd_pcm_hw_params_set_buffer_time(snd_pcm_t *,
                                             snd_pcm_hw_params_t
*,
                                             unsigned int, int);
extern int snd_pcm_hw_params_set_channels(snd_pcm_t *,
                                          snd_pcm_hw_params_t *,
                                          unsigned int);
extern     int     snd_pcm_hw_params_set_format(snd_pcm_t     *,
snd_pcm_hw_params_t *,
                                            snd_pcm_format_t);
extern int snd_pcm_hw_params_set_format_mask(snd_pcm_t *,
                                             snd_pcm_hw_params_t
*,

snd_pcm_format_mask_t *);
extern int snd_pcm_hw_params_set_period_size(snd_pcm_t *,
                                             snd_pcm_hw_params_t
*,
                                             snd_pcm_uframes_t,
int);
extern int snd_pcm_hw_params_set_period_time(snd_pcm_t *,
                                             snd_pcm_hw_params_t
*,
                                             unsigned int, int);
extern int snd_pcm_hw_params_set_periods(snd_pcm_t *,
                                         snd_pcm_hw_params_t *,
                                         unsigned int, int);
extern int snd_pcm_hw_params_set_periods_integer(snd_pcm_t *,

snd_pcm_hw_params_t *);
extern      int      snd_pcm_hw_params_set_rate(snd_pcm_t      *,
snd_pcm_hw_params_t *,
                                       unsigned int, int);
extern int snd_pcm_hw_params_set_rate_resample(snd_pcm_t *,

snd_pcm_hw_params_t *,
                                                unsigned int);
extern size_t snd_pcm_hw_params_sizeof(void);
extern int snd_pcm_hw_params_test_access(snd_pcm_t *,
                                         snd_pcm_hw_params_t *,
                                         snd_pcm_access_t);
extern int snd_pcm_hw_params_test_buffer_size(snd_pcm_t *,
                                              snd_pcm_hw_params_t
*,
                                              snd_pcm_uframes_t);
extern int snd_pcm_hw_params_test_buffer_time(snd_pcm_t *,
                                              snd_pcm_hw_params_t
*,
                                              unsigned int, int);
extern int snd_pcm_hw_params_test_channels(snd_pcm_t *,
                                           snd_pcm_hw_params_t *,
                                           unsigned int);
extern int snd_pcm_hw_params_test_format(snd_pcm_t *,
                                         snd_pcm_hw_params_t *,
                                         snd_pcm_format_t);
extern int snd_pcm_hw_params_test_period_size(snd_pcm_t *,
```

```
                                                    snd_pcm_hw_params_t
*,
                                                    snd_pcm_uframes_t,
int);
extern int snd_pcm_hw_params_test_period_time(snd_pcm_t *,
                                                    snd_pcm_hw_params_t
*,
                                                    unsigned int, int);
extern int snd_pcm_hw_params_test_periods(snd_pcm_t *,
                                            snd_pcm_hw_params_t *,
                                            unsigned int, int);
extern      int      snd_pcm_hw_params_test_rate(snd_pcm_t      *,
snd_pcm_hw_params_t *,
                                          unsigned int, int);
extern int snd_pcm_hwsync(snd_pcm_t *);
extern int snd_pcm_info(snd_pcm_t *, snd_pcm_info_t *);
extern    void    snd_pcm_info_copy(snd_pcm_info_t    *,    const
snd_pcm_info_t *);
extern void snd_pcm_info_free(snd_pcm_info_t *);
extern int snd_pcm_info_get_card(const snd_pcm_info_t *);
extern        snd_pcm_class_t        snd_pcm_info_get_class(const
snd_pcm_info_t *);
extern  unsigned  int  snd_pcm_info_get_device(const  snd_pcm_info_t
*);
extern const char *snd_pcm_info_get_id(const snd_pcm_info_t *);
extern const char *snd_pcm_info_get_name(const snd_pcm_info_t *);
extern       snd_pcm_stream_t       snd_pcm_info_get_stream(const
snd_pcm_info_t *);
extern      unsigned      int      snd_pcm_info_get_subdevice(const
snd_pcm_info_t *);
extern    const    char    *snd_pcm_info_get_subdevice_name(const
snd_pcm_info_t *);
extern   unsigned   int   snd_pcm_info_get_subdevices_avail(const
snd_pcm_info_t
                                                          *);
extern   unsigned   int   snd_pcm_info_get_subdevices_count(const
snd_pcm_info_t
                                                          *);
extern int snd_pcm_info_malloc(snd_pcm_info_t * *);
extern  void  snd_pcm_info_set_device(snd_pcm_info_t  *,  unsigned
int);
extern     void     snd_pcm_info_set_stream(snd_pcm_info_t     *,
snd_pcm_stream_t);
extern void snd_pcm_info_set_subdevice(snd_pcm_info_t *, unsigned
int);
extern size_t snd_pcm_info_sizeof(void);
extern int snd_pcm_link(snd_pcm_t *, snd_pcm_t *);
extern int snd_pcm_mmap_begin(snd_pcm_t *,
                              const snd_pcm_channel_area_t * *,
                              snd_pcm_uframes_t                *,
snd_pcm_uframes_t *);
extern snd_pcm_sframes_t snd_pcm_mmap_commit(snd_pcm_t *,
                                             snd_pcm_uframes_t,
                                             snd_pcm_uframes_t);
extern snd_pcm_sframes_t snd_pcm_mmap_readi(snd_pcm_t *, void *,
                                            snd_pcm_uframes_t);
extern snd_pcm_sframes_t snd_pcm_mmap_readn(snd_pcm_t *, void **,
                                            snd_pcm_uframes_t);
extern  snd_pcm_sframes_t  snd_pcm_mmap_writei(snd_pcm_t  *,  const
void *,
                                             snd_pcm_uframes_t);
extern  snd_pcm_sframes_t  snd_pcm_mmap_writen(snd_pcm_t  *,  void
**,
                                             snd_pcm_uframes_t);
extern const char *snd_pcm_name(snd_pcm_t *);
extern int snd_pcm_nonblock(snd_pcm_t *, int);
```

```
extern   int   snd_pcm_open(snd_pcm_t   *   *,   const   char   *,
snd_pcm_stream_t,
                        int);
extern int snd_pcm_open_lconf(snd_pcm_t * *, const char *,
                              snd_pcm_stream_t, int, snd_config_t
*);
extern int snd_pcm_pause(snd_pcm_t *, int);
extern int snd_pcm_poll_descriptors(snd_pcm_t *, struct pollfd *,
                                    unsigned int);
extern int snd_pcm_poll_descriptors_count(snd_pcm_t *);
extern  int  snd_pcm_poll_descriptors_revents(snd_pcm_t  *,  struct
pollfd *,
                                              unsigned int,
                                              short   unsigned   int
*);
extern int snd_pcm_prepare(snd_pcm_t *);
extern snd_pcm_sframes_t snd_pcm_readi(snd_pcm_t *, void *,
                                       snd_pcm_uframes_t);
extern snd_pcm_sframes_t snd_pcm_readn(snd_pcm_t *, void **,
                                       snd_pcm_uframes_t);
extern int snd_pcm_recover(snd_pcm_t *, int, int);
extern int snd_pcm_reset(snd_pcm_t *);
extern int snd_pcm_resume(snd_pcm_t *);
extern      snd_pcm_sframes_t      snd_pcm_rewind(snd_pcm_t      *,
snd_pcm_uframes_t);
extern ssize_t snd_pcm_samples_to_bytes(snd_pcm_t *, long int);
extern int snd_pcm_start(snd_pcm_t *);
extern snd_pcm_state_t snd_pcm_state(snd_pcm_t *);
extern const char *snd_pcm_state_name(snd_pcm_state_t);
extern int snd_pcm_status(snd_pcm_t *, snd_pcm_status_t *);
extern void snd_pcm_status_copy(snd_pcm_status_t *,
                                const snd_pcm_status_t *);
extern  int  snd_pcm_status_dump(snd_pcm_status_t  *,  snd_output_t
*);
extern void snd_pcm_status_free(snd_pcm_status_t *);
extern      snd_pcm_uframes_t      snd_pcm_status_get_avail(const
snd_pcm_status_t
                                                        *);
extern snd_pcm_uframes_t snd_pcm_status_get_avail_max(const

snd_pcm_status_t *);
extern      snd_pcm_sframes_t      snd_pcm_status_get_delay(const
snd_pcm_status_t
                                                        *);
extern       snd_pcm_state_t       snd_pcm_status_get_state(const
snd_pcm_status_t *);
extern        void        snd_pcm_status_get_trigger_tstamp(const
snd_pcm_status_t *,
                                                      snd_timestamp_t *);
extern void snd_pcm_status_get_tstamp(const snd_pcm_status_t *,
                                      snd_timestamp_t *);
extern int snd_pcm_status_malloc(snd_pcm_status_t * *);
extern size_t snd_pcm_status_sizeof(void);
extern snd_pcm_stream_t snd_pcm_stream(snd_pcm_t *);
extern const char *snd_pcm_stream_name(snd_pcm_stream_t);
extern int snd_pcm_sw_params(snd_pcm_t *, snd_pcm_sw_params_t *);
extern void snd_pcm_sw_params_copy(snd_pcm_sw_params_t *,
                                   const snd_pcm_sw_params_t *);
extern      int      snd_pcm_sw_params_current(snd_pcm_t      *,
snd_pcm_sw_params_t *);
extern    int    snd_pcm_sw_params_dump(snd_pcm_sw_params_t    *,
snd_output_t *);
extern void snd_pcm_sw_params_free(snd_pcm_sw_params_t *);
extern          int          snd_pcm_sw_params_get_boundary(const
snd_pcm_sw_params_t *,
                                             snd_pcm_uframes_t *);
```

```
extern int snd_pcm_sw_params_malloc(snd_pcm_sw_params_t * *);
extern int snd_pcm_sw_params_set_avail_min(snd_pcm_t *,
                                           snd_pcm_sw_params_t *,
                                           snd_pcm_uframes_t);
extern int snd_pcm_sw_params_set_silence_size(snd_pcm_t *,
                                              snd_pcm_sw_params_t
*,
                                              snd_pcm_uframes_t);
extern int snd_pcm_sw_params_set_silence_threshold(snd_pcm_t *,

snd_pcm_sw_params_t *,

snd_pcm_uframes_t);
extern int snd_pcm_sw_params_set_start_threshold(snd_pcm_t *,

snd_pcm_sw_params_t *,

snd_pcm_uframes_t);
extern int snd_pcm_sw_params_set_stop_threshold(snd_pcm_t *,

snd_pcm_sw_params_t *,

snd_pcm_uframes_t);
extern int snd_pcm_sw_params_set_tstamp_mode(snd_pcm_t *,
                                             snd_pcm_sw_params_t
*,
                                             snd_pcm_tstamp_t);
extern int snd_pcm_sw_params_set_xfer_align(snd_pcm_t *,
                                            snd_pcm_sw_params_t
*,
                                            snd_pcm_uframes_t);
extern size_t snd_pcm_sw_params_sizeof(void);
extern snd_pcm_type_t snd_pcm_type(snd_pcm_t *);
extern int snd_pcm_unlink(snd_pcm_t *);
extern int snd_pcm_wait(snd_pcm_t *, int);
extern  snd_pcm_sframes_t  snd_pcm_writei(snd_pcm_t  *,  const  void
*,
                                          snd_pcm_uframes_t);
extern snd_pcm_sframes_t snd_pcm_writen(snd_pcm_t *, void **,
                                          snd_pcm_uframes_t);
extern          int          snd_pcm_hw_params_get_rate_min(const
snd_pcm_hw_params_t *,
                                            unsigned int *, int *);
extern          int          snd_pcm_sw_params_get_avail_min(const
snd_pcm_sw_params_t *,
                                           snd_pcm_uframes_t *);
extern         int         snd_pcm_hw_params_get_period_time(const
snd_pcm_hw_params_t *,
                                             unsigned  int  *,  int
*);
extern int snd_pcm_hw_params_set_buffer_time_near(snd_pcm_t *,

snd_pcm_hw_params_t *,
                                                  unsigned int *,
int *);
extern int snd_pcm_hw_params_get_format(const snd_pcm_hw_params_t
*,
                                          snd_pcm_format_t *);
extern        int        snd_pcm_hw_params_get_channels_min(const
snd_pcm_hw_params_t *,
                                             unsigned int *);
extern       int       snd_pcm_sw_params_get_start_threshold(const
snd_pcm_sw_params_t
                                                    *,
snd_pcm_uframes_t *);
extern int snd_pcm_hw_params_set_period_time_near(snd_pcm_t *,
```



```
snd_pcm_hw_params_t *,
                                                    unsigned int *,
int *);
extern int snd_pcm_hw_params_set_channels_near(snd_pcm_t *,

snd_pcm_hw_params_t *,
                                                  unsigned int *);
extern          int          snd_pcm_hw_params_get_rate_max(const
snd_pcm_hw_params_t *,
                                               unsigned int *, int *);
extern         int         snd_pcm_hw_params_get_period_size(const
snd_pcm_hw_params_t *,
                                                 snd_pcm_uframes_t  *,
int *);
extern       int       snd_pcm_hw_params_get_period_time_max(const
snd_pcm_hw_params_t
                                                    *,  unsigned  int
*, int *);
extern         int         snd_pcm_hw_params_get_buffer_size(const
snd_pcm_hw_params_t *,
                                                 snd_pcm_uframes_t
*);
extern int snd_pcm_sw_params_get_silence_threshold(const

snd_pcm_sw_params_t *,

snd_pcm_uframes_t *);
extern int snd_pcm_hw_params_get_access(const snd_pcm_hw_params_t
*,
                                         snd_pcm_access_t *);
extern       int       snd_pcm_hw_params_get_buffer_time_min(const
snd_pcm_hw_params_t
                                                    *,  unsigned  int
*, int *);
extern         int         snd_pcm_hw_params_get_buffer_time(const
snd_pcm_hw_params_t *,
                                                 unsigned  int  *,  int
*);
extern  int  snd_pcm_hw_params_get_rate(const  snd_pcm_hw_params_t
*,
                                       unsigned int *, int *);
extern       int       snd_pcm_sw_params_get_stop_threshold(const
snd_pcm_sw_params_t
                                                   *,
snd_pcm_uframes_t *);
extern       int       snd_pcm_hw_params_get_period_size_max(const
snd_pcm_hw_params_t
                                                    *,
snd_pcm_uframes_t *,
                                                    int *);
extern       int       snd_pcm_hw_params_get_buffer_size_min(const
snd_pcm_hw_params_t
                                                    *,
snd_pcm_uframes_t *);
extern const char *snd_pcm_type_name(snd_pcm_type_t);
extern        int        snd_pcm_sw_params_get_silence_size(const
snd_pcm_sw_params_t *,
                                                  snd_pcm_uframes_t
*);
extern          int          snd_pcm_hw_params_get_periods(const
snd_pcm_hw_params_t *,
                                         unsigned int *, int *);
extern       int       snd_pcm_hw_params_get_buffer_size_max(const
snd_pcm_hw_params_t
```

```
                                                    *,
snd_pcm_uframes_t *);
extern         int         snd_pcm_sw_params_get_tstamp_mode(const
snd_pcm_sw_params_t *,
                                                  snd_pcm_tstamp_t *);
extern int snd_pcm_hw_params_set_period_size_near(snd_pcm_t *,

snd_pcm_hw_params_t *,

snd_pcm_uframes_t *,
                                                   int *);
extern       int       snd_pcm_hw_params_get_buffer_time_max(const
snd_pcm_hw_params_t
                                                   *,  unsigned  int
*, int *);
extern       int       snd_pcm_hw_params_get_period_time_min(const
snd_pcm_hw_params_t
                                                   *,  unsigned  int
*, int *);
extern int snd_pcm_hw_params_set_periods_near(snd_pcm_t *,
                                                  snd_pcm_hw_params_t
*,
                                                  unsigned int *, int
*);
extern     snd_pcm_sframes_t     snd_pcm_forward(snd_pcm_t     *,
snd_pcm_uframes_t);
extern int snd_pcm_hw_params_set_rate_near(snd_pcm_t *,
                                           snd_pcm_hw_params_t *,
                                           unsigned  int  *,  int
*);
extern       int       snd_pcm_hw_params_get_period_size_min(const
snd_pcm_hw_params_t
                                                   *,
snd_pcm_uframes_t *,
                                                   int *);
extern         int         snd_pcm_hw_params_get_periods_min(const
snd_pcm_hw_params_t *,
                                                 unsigned  int  *,  int
*);
extern          int          snd_pcm_hw_params_get_channels(const
snd_pcm_hw_params_t *,
                                               unsigned int *);
extern         int         snd_pcm_hw_params_get_channels_max(const
snd_pcm_hw_params_t *,
                                                  unsigned int *);
extern         int         snd_pcm_hw_params_get_periods_max(const
snd_pcm_hw_params_t *,
                                                 unsigned  int  *,  int
*);
extern int snd_pcm_hw_params_set_buffer_size_near(snd_pcm_t *,

snd_pcm_hw_params_t *,

snd_pcm_uframes_t *);
```

6.2.14 alsa/pcm_extplug.h

```
#define                                    SND_PCM_EXTPLUG_VERSION
((SND_PCM_EXTPLUG_VERSION_MAJOR<<16)                              |
(SND_PCM_EXTPLUG_VERSION_MINOR<<8)                                |
(SND_PCM_EXTPLUG_VERSION_TINY))
#define SND_PCM_EXTPLUG_VERSION_MINOR   0
#define SND_PCM_EXTPLUG_VERSION_MAJOR   1
#define SND_PCM_EXTPLUG_VERSION_TINY    1
```



```
typedef struct snd_pcm_extplug_callback {
    snd_pcm_sframes_t(*transfer) (void);
    int (*close) (void);
    int (*hw_params) (void);
    int (*hw_free) (void);
    void (*dump) (void);
    int (*init) (void);
} snd_pcm_extplug_callback_t;
typedef struct snd_pcm_extplug {
    unsigned int version;
    const char *name;
    const snd_pcm_extplug_callback_t *callback;
    void *private_data;
    snd_pcm_t *pcm;
    snd_pcm_stream_t stream;
    snd_pcm_format_t format;
    snd_pcm_subformat_t subformat;
    unsigned int channels;
    unsigned int rate;
    snd_pcm_format_t slave_format;
    snd_pcm_subformat_t slave_subformat;
    unsigned int slave_channels;
} snd_pcm_extplug_t;
```

6.2.15 alsa/pcm_plugin.h

```
#define SND_PCM_PLUGIN_ROUTE_HALF       0.5
#define SND_PCM_PLUGIN_ROUTE_FLOAT      1
#define SND_PCM_PLUGIN_ROUTE_FULL       1.0
#define SND_PCM_PLUGIN_ROUTE_RESOLUTION 16
#define SND_PCM_PLUGIN_RATE_MAX 192000
#define SND_PCM_PLUGIN_RATE_MIN 4000

typedef float snd_pcm_route_ttable_entry_t;
```

6.2.16 alsa/rawmidi.h

```
#define SND_RAWMIDI_APPEND      0x0001
#define SND_RAWMIDI_NONBLOCK    0x0002
#define SND_RAWMIDI_SYNC        0x0004

typedef struct sndrv_rawmidi_info snd_rawmidi_info_t;
typedef struct sndrv_rawmidi_params snd_rawmidi_params_t;
typedef struct sndrv_rawmidi_status snd_rawmidi_status_t;
typedef enum _snd_rawmidi_stream {
    SND_RAWMIDI_STREAM_OUTPUT,
    SND_RAWMIDI_STREAM_INPUT,
    SND_RAWMIDI_STREAM_LAST
} snd_rawmidi_stream_t;
typedef struct _snd_rawmidi snd_rawmidi_t;
typedef enum _snd_rawmidi_type {
    SND_RAWMIDI_TYPE_HW,
    SND_RAWMIDI_TYPE_SHM,
    SND_RAWMIDI_TYPE_INET,
    SND_RAWMIDI_TYPE_VIRTUAL
} snd_rawmidi_type_t;
extern int snd_rawmidi_close(snd_rawmidi_t *);
extern int snd_rawmidi_drain(snd_rawmidi_t *);
extern int snd_rawmidi_drop(snd_rawmidi_t *);
extern void snd_rawmidi_info_free(snd_rawmidi_info_t *);
extern       const       char       *snd_rawmidi_info_get_id(const
snd_rawmidi_info_t *);
extern      const      char      *snd_rawmidi_info_get_name(const
snd_rawmidi_info_t *);
```

```
extern unsigned int snd_rawmidi_info_get_subdevices_count(const

snd_rawmidi_info_t
                                                          *);
extern int snd_rawmidi_info_malloc(snd_rawmidi_info_t * *);
extern void snd_rawmidi_info_set_device(snd_rawmidi_info_t *,
                                        unsigned int);
extern void snd_rawmidi_info_set_stream(snd_rawmidi_info_t *,
                                        snd_rawmidi_stream_t);
extern void snd_rawmidi_info_set_subdevice(snd_rawmidi_info_t *,
                                           unsigned int);
extern int snd_rawmidi_nonblock(snd_rawmidi_t *, int);
extern int snd_rawmidi_open(snd_rawmidi_t * *, snd_rawmidi_t * *,
                            const char *, int);
extern  int  snd_rawmidi_poll_descriptors(snd_rawmidi_t  *,  struct
pollfd *,
                                        unsigned int);
extern int snd_rawmidi_poll_descriptors_count(snd_rawmidi_t *);
extern int snd_rawmidi_poll_descriptors_revents(snd_rawmidi_t *,
                                                struct pollfd *,
                                                unsigned int,
                                                short     unsigned
int *);
extern ssize_t snd_rawmidi_read(snd_rawmidi_t *, void *, size_t);
extern  ssize_t  snd_rawmidi_write(snd_rawmidi_t  *,  const  void  *,
size_t);
```

6.2.17 alsa/seq.h

```
#define snd_seq_ev_is_prior(ev)  \
        (((ev)->flags       &       SND_SEQ_PRIORITY_MASK)       ==
SND_SEQ_PRIORITY_HIGH)
#define snd_seq_ev_length_type(ev)       \
        ((ev)->flags & SND_SEQ_EVENT_LENGTH_MASK)
#define snd_seq_ev_timemode_type(ev)     \
        ((ev)->flags & SND_SEQ_TIME_MODE_MASK)
#define snd_seq_ev_timestamp_type(ev)    \
        ((ev)->flags & SND_SEQ_TIME_STAMP_MASK)
#define snd_seq_ev_is_channel_type(ev)   \
        (snd_seq_event_types[(ev)->type]                          &
(_SND_SEQ_TYPE(SND_SEQ_EVFLG_NOTE) \
        | _SND_SEQ_TYPE(SND_SEQ_EVFLG_CONTROL)))
#define snd_seq_type_check(ev,x)         \
        (snd_seq_event_types[(ev)->type] & _SND_SEQ_TYPE(x))
#define snd_seq_ev_is_fixed(ev)  \
        (snd_seq_ev_length_type(ev)                               ==
SND_SEQ_EVENT_LENGTH_FIXED)
#define snd_seq_ev_is_variable(ev)       \
        (snd_seq_ev_length_type(ev)                               ==
SND_SEQ_EVENT_LENGTH_VARIABLE)
#define snd_seq_ev_is_varusr(ev)         \
        (snd_seq_ev_length_type(ev)                               ==
SND_SEQ_EVENT_LENGTH_VARUSR)
#define snd_seq_ev_is_abstime(ev)        \
        (snd_seq_ev_timemode_type(ev) == SND_SEQ_TIME_MODE_ABS)
#define snd_seq_ev_is_reltime(ev)        \
        (snd_seq_ev_timemode_type(ev) == SND_SEQ_TIME_MODE_REL)
#define snd_seq_ev_is_real(ev)   \
        (snd_seq_ev_timestamp_type(ev)                            ==
SND_SEQ_TIME_STAMP_REAL)
#define snd_seq_ev_is_tick(ev)   \
        (snd_seq_ev_timestamp_type(ev)                            ==
SND_SEQ_TIME_STAMP_TICK)
#define snd_seq_ev_is_subscribe_type(ev)         \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_CONNECTION)
```

```
#define snd_seq_ev_is_control_type(ev)   \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_CONTROL)
#define snd_seq_ev_is_fixed_type(ev)     \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_FIXED)
#define snd_seq_ev_is_instr_type(ev)     \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_INSTR)
#define snd_seq_ev_is_message_type(ev)   \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_MESSAGE)
#define snd_seq_ev_is_note_type(ev)      \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_NOTE)
#define snd_seq_ev_is_queue_type(ev)     \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_QUEUE)
#define snd_seq_ev_is_result_type(ev)    \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_RESULT)
#define snd_seq_ev_is_sample_type(ev)    \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_SAMPLE)
#define snd_seq_ev_is_user_type(ev)      \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_USERS)
#define snd_seq_ev_is_variable_type(ev)  \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_VARIABLE)
#define snd_seq_ev_is_varusr_type(ev)    \
        snd_seq_type_check(ev, SND_SEQ_EVFLG_VARUSR)
#define    snd_seq_ev_is_reserved(ev)                          (!
snd_seq_event_types[(ev)->type])
#define  snd_seq_ev_is_direct(ev)               ((ev)->queue  ==
SND_SEQ_QUEUE_DIRECT)
#define _SND_SEQ_TYPE_OPT(x)    ((x)<<24)
#define _SND_SEQ_TYPE(x)        (1<<(x))
#define SND_SEQ_PORT_CAP_READ   (1<<0)
#define SND_SEQ_PORT_TYPE_SPECIFIC      (1<<0)
#define SND_SEQ_REMOVE_INPUT    (1<<0)
#define SND_SEQ_PORT_CAP_WRITE  (1<<1)
#define SND_SEQ_PORT_TYPE_MIDI_GENERIC  (1<<1)
#define SND_SEQ_REMOVE_OUTPUT   (1<<1)
#define SND_SEQ_PORT_TYPE_SYNTH (1<<10)
#define SND_SEQ_PORT_TYPE_DIRECT_SAMPLE (1<<11)
#define SND_SEQ_PORT_TYPE_SAMPLE        (1<<12)
#define SND_SEQ_PORT_TYPE_HARDWARE      (1<<16)
#define SND_SEQ_PORT_TYPE_SOFTWARE      (1<<17)
#define SND_SEQ_PORT_TYPE_SYNTHESIZER   (1<<18)
#define SND_SEQ_PORT_TYPE_PORT  (1<<19)
#define SND_SEQ_PORT_CAP_SYNC_READ      (1<<2)
#define SND_SEQ_PORT_TYPE_MIDI_GM       (1<<2)
#define SND_SEQ_REMOVE_DEST     (1<<2)
#define SND_SEQ_PORT_TYPE_APPLICATION   (1<<20)
#define SND_SEQ_PORT_CAP_SYNC_WRITE     (1<<3)
#define SND_SEQ_PORT_TYPE_MIDI_GS       (1<<3)
#define SND_SEQ_REMOVE_DEST_CHANNEL     (1<<3)
#define SND_SEQ_PORT_CAP_DUPLEX (1<<4)
#define SND_SEQ_PORT_TYPE_MIDI_XG       (1<<4)
#define SND_SEQ_REMOVE_TIME_BEFORE      (1<<4)
#define SND_SEQ_PORT_CAP_SUBS_READ      (1<<5)
#define SND_SEQ_PORT_TYPE_MIDI_MT32     (1<<5)
#define SND_SEQ_REMOVE_TIME_AFTER       (1<<5)
#define SND_SEQ_PORT_CAP_SUBS_WRITE     (1<<6)
#define SND_SEQ_PORT_TYPE_MIDI_GM2      (1<<6)
#define SND_SEQ_REMOVE_TIME_TICK        (1<<6)
#define SND_SEQ_PORT_CAP_NO_EXPORT      (1<<7)
#define SND_SEQ_REMOVE_EVENT_TYPE       (1<<7)
#define SND_SEQ_REMOVE_IGNORE_OFF       (1<<8)
#define SND_SEQ_REMOVE_TAG_MATCH        (1<<9)
#define                                       SND_SEQ_OPEN_DUPLEX
(SND_SEQ_OPEN_OUTPUT|SND_SEQ_OPEN_INPUT)
#define SND_SEQ_CLIENT_SYSTEM   0
#define SND_SEQ_PORT_SYSTEM_TIMER       0
#define SND_SEQ_NONBLOCK        0x0001
```

```
#define SND_SEQ_OPEN_OUTPUT     1
#define SND_SEQ_PORT_SYSTEM_ANNOUNCE    1
#define SND_SEQ_OPEN_INPUT      2
#define SND_SEQ_ADDRESS_UNKNOWN 253
#define SND_SEQ_QUEUE_DIRECT    253
#define SND_SEQ_ADDRESS_SUBSCRIBERS     254
#define SND_SEQ_ADDRESS_BROADCAST       255

typedef struct sndrv_seq_client_info snd_seq_client_info_t;
typedef struct sndrv_seq_client_pool snd_seq_client_pool_t;
typedef enum snd_seq_client_type {
    SND_SEQ_USER_CLIENT,
    SND_SEQ_KERNEL_CLIENT
} snd_seq_client_type_t;
typedef struct sndrv_seq_port_info snd_seq_port_info_t;
typedef struct sndrv_seq_port_subscribe snd_seq_port_subscribe_t;
typedef enum {
    SND_SEQ_QUERY_SUBS_READ,
    SND_SEQ_QUERY_SUBS_WRITE
} snd_seq_query_subs_type_t;
typedef struct sndrv_seq_query_subs snd_seq_query_subscribe_t;
typedef struct sndrv_seq_queue_info snd_seq_queue_info_t;
typedef struct sndrv_seq_queue_status snd_seq_queue_status_t;
typedef struct sndrv_seq_queue_tempo snd_seq_queue_tempo_t;
typedef struct sndrv_seq_queue_timer snd_seq_queue_timer_t;
typedef enum {
    SND_SEQ_TIMER_ALSA,
    SND_SEQ_TIMER_MIDI_CLOCK,
    SND_SEQ_TIMER_MIDI_TICK
} snd_seq_queue_timer_type_t;
typedef struct sndrv_seq_remove_events snd_seq_remove_events_t;
typedef struct sndrv_seq_system_info snd_seq_system_info_t;
typedef struct _snd_seq snd_seq_t;
typedef enum _snd_seq_type {
    SND_SEQ_TYPE_HW,
    SND_SEQ_TYPE_SHM,
    SND_SEQ_TYPE_INET
} snd_seq_type_t;
extern int snd_seq_alloc_named_queue(snd_seq_t *, const char *);
extern int snd_seq_alloc_queue(snd_seq_t *);
extern int snd_seq_client_id(snd_seq_t *);
extern void snd_seq_client_info_copy(snd_seq_client_info_t *,
                                     const   snd_seq_client_info_t
*);
extern void snd_seq_client_info_free(snd_seq_client_info_t *);
extern          int          snd_seq_client_info_get_client(const
snd_seq_client_info_t *);
extern                          const                           char
*snd_seq_client_info_get_name(snd_seq_client_info_t *);
extern         int         snd_seq_client_info_get_num_ports(const
snd_seq_client_info_t
                                                *);
extern snd_seq_client_type_t snd_seq_client_info_get_type(const

snd_seq_client_info_t
                                                          *);
extern int snd_seq_client_info_malloc(snd_seq_client_info_t * *);
extern  void  snd_seq_client_info_set_client(snd_seq_client_info_t
*, int);
extern void snd_seq_client_info_set_name(snd_seq_client_info_t *,
                                         const char *);
extern size_t snd_seq_client_info_sizeof(void);
extern int snd_seq_close(snd_seq_t *);
extern  int  snd_seq_create_port(snd_seq_t  *,  snd_seq_port_info_t
*);
extern int snd_seq_delete_port(snd_seq_t *, int);
```



```
extern int snd_seq_drain_output(snd_seq_t *);
extern int snd_seq_drop_output(snd_seq_t *);
extern int snd_seq_drop_output_buffer(snd_seq_t *);
extern int snd_seq_event_input(snd_seq_t *, snd_seq_event_t * *);
extern int snd_seq_event_input_pending(snd_seq_t *, int);
extern ssize_t snd_seq_event_length(snd_seq_event_t *);
extern int snd_seq_event_output(snd_seq_t *, snd_seq_event_t *);
extern      int      snd_seq_event_output_direct(snd_seq_t      *,
snd_seq_event_t *);
extern int snd_seq_free_event(snd_seq_event_t *);
extern int snd_seq_free_queue(snd_seq_t *, int);
extern int snd_seq_get_any_client_info(snd_seq_t *, int,
                                       snd_seq_client_info_t *);
extern int snd_seq_get_any_port_info(snd_seq_t *, int, int,
                                     snd_seq_port_info_t *);
extern       int       snd_seq_get_client_info(snd_seq_t       *,
snd_seq_client_info_t *);
extern size_t snd_seq_get_input_buffer_size(snd_seq_t *);
extern size_t snd_seq_get_output_buffer_size(snd_seq_t *);
extern     int     snd_seq_get_port_info(snd_seq_t     *,     int,
snd_seq_port_info_t *);
extern int snd_seq_get_port_subscription(snd_seq_t *,
                                         snd_seq_port_subscribe_t
*);
extern int snd_seq_get_queue_status(snd_seq_t *, int,
                                    snd_seq_queue_status_t *);
extern int snd_seq_get_queue_tempo(snd_seq_t *, int,
                                   snd_seq_queue_tempo_t *);
extern int snd_seq_nonblock(snd_seq_t *, int);
extern int snd_seq_open(snd_seq_t * *, const char *, int, int);
extern int snd_seq_poll_descriptors(snd_seq_t *, struct pollfd *,
                                    unsigned int, short int);
extern  int  snd_seq_poll_descriptors_count(snd_seq_t  *,  short
int);
extern  int  snd_seq_poll_descriptors_revents(snd_seq_t  *,  struct
pollfd *,
                                              unsigned int,
                                              short   unsigned   int
*);
extern void snd_seq_port_info_copy(snd_seq_port_info_t *,
                                   const snd_seq_port_info_t *);
extern void snd_seq_port_info_free(snd_seq_port_info_t *);
extern const snd_seq_addr_t *snd_seq_port_info_get_addr(const

snd_seq_port_info_t
                                                        *);
extern unsigned int snd_seq_port_info_get_capability(const

snd_seq_port_info_t
                                                     *);
extern int snd_seq_port_info_get_client(const snd_seq_port_info_t
*);
extern      const      char      *snd_seq_port_info_get_name(const
snd_seq_port_info_t *);
extern  int  snd_seq_port_info_get_port(const  snd_seq_port_info_t
*);
extern     unsigned     int     snd_seq_port_info_get_type(const
snd_seq_port_info_t
                                                       *);
extern int snd_seq_port_info_malloc(snd_seq_port_info_t * *);
extern  void  snd_seq_port_info_set_capability(snd_seq_port_info_t
*,
                                               unsigned int);
extern  void  snd_seq_port_info_set_client(snd_seq_port_info_t  *,
int);
```

```
extern                                                          void
snd_seq_port_info_set_midi_channels(snd_seq_port_info_t *,
                                                      int);
extern void snd_seq_port_info_set_name(snd_seq_port_info_t *,
                                       const char *);
extern   void   snd_seq_port_info_set_port(snd_seq_port_info_t   *,
int);
extern                                                          void
snd_seq_port_info_set_port_specified(snd_seq_port_info_t *,
                                                       int);
extern                                                          void
snd_seq_port_info_set_timestamp_queue(snd_seq_port_info_t *,
                                                        int);
extern                                                          void
snd_seq_port_info_set_timestamp_real(snd_seq_port_info_t *,
                                                       int);
extern                                                          void
snd_seq_port_info_set_timestamping(snd_seq_port_info_t *, int);
extern void snd_seq_port_info_set_type(snd_seq_port_info_t *,
                                       unsigned int);
extern size_t snd_seq_port_info_sizeof(void);
extern  void  snd_seq_port_subscribe_copy(snd_seq_port_subscribe_t
*,
                                          const
snd_seq_port_subscribe_t *);
extern  void  snd_seq_port_subscribe_free(snd_seq_port_subscribe_t
*);
extern                     const                     snd_seq_addr_t
*snd_seq_port_subscribe_get_dest(const

snd_seq_port_subscribe_t
                                                                *);
extern int snd_seq_port_subscribe_get_exclusive(const

snd_seq_port_subscribe_t
                                                       *);
extern         int         snd_seq_port_subscribe_get_queue(const
snd_seq_port_subscribe_t
                                                  *);
extern                     const                     snd_seq_addr_t
*snd_seq_port_subscribe_get_sender(const

snd_seq_port_subscribe_t

*);
extern int snd_seq_port_subscribe_get_time_real(const

snd_seq_port_subscribe_t
                                                       *);
extern int snd_seq_port_subscribe_get_time_update(const

snd_seq_port_subscribe_t
                                                         *);
extern  int  snd_seq_port_subscribe_malloc(snd_seq_port_subscribe_t
* *);
extern                                                          void
snd_seq_port_subscribe_set_dest(snd_seq_port_subscribe_t *,
                                                 const   snd_seq_addr_t
*);
extern                                                          void
snd_seq_port_subscribe_set_exclusive(snd_seq_port_subscribe_t
                                                       *, int);
extern                                                          void
snd_seq_port_subscribe_set_queue(snd_seq_port_subscribe_t *,
                                                  int);
```



```
extern                                                          void
snd_seq_port_subscribe_set_sender(snd_seq_port_subscribe_t *,
                                                  const
snd_seq_addr_t *);
extern                                                          void
snd_seq_port_subscribe_set_time_real(snd_seq_port_subscribe_t
                                                    *, int);
extern                                                          void
snd_seq_port_subscribe_set_time_update(snd_seq_port_subscribe_t
                                                      *, int);
extern size_t snd_seq_port_subscribe_sizeof(void);
extern      int      snd_seq_query_next_client(snd_seq_t      *,
snd_seq_client_info_t *);
extern       int       snd_seq_query_next_port(snd_seq_t       *,
snd_seq_port_info_t *);
extern int snd_seq_query_port_subscribers(snd_seq_t *,

snd_seq_query_subscribe_t *);
extern                                                          void
snd_seq_query_subscribe_copy(snd_seq_query_subscribe_t *,
                                        const
snd_seq_query_subscribe_t
                                        *);
extern                                                          void
snd_seq_query_subscribe_free(snd_seq_query_subscribe_t *);
extern                    const                    snd_seq_addr_t
*snd_seq_query_subscribe_get_addr(const

snd_seq_query_subscribe_t
                                                                *);
extern int snd_seq_query_subscribe_get_exclusive(const

snd_seq_query_subscribe_t
                                                    *);
extern int snd_seq_query_subscribe_get_index(const

snd_seq_query_subscribe_t *);
extern int snd_seq_query_subscribe_get_queue(const

snd_seq_query_subscribe_t *);
extern                    const                    snd_seq_addr_t
*snd_seq_query_subscribe_get_root(const

snd_seq_query_subscribe_t
                                                                *);
extern int snd_seq_query_subscribe_get_time_real(const

snd_seq_query_subscribe_t
                                                    *);
extern int snd_seq_query_subscribe_get_time_update(const

snd_seq_query_subscribe_t
                                                      *);
extern                                                           int
snd_seq_query_subscribe_malloc(snd_seq_query_subscribe_t * *);
extern                                                          void
snd_seq_query_subscribe_set_index(snd_seq_query_subscribe_t *,
                                                  int);
extern                                                          void
snd_seq_query_subscribe_set_root(snd_seq_query_subscribe_t *,
                                                const  snd_seq_addr_t
*);
extern                                                          void
snd_seq_query_subscribe_set_type(snd_seq_query_subscribe_t *,

snd_seq_query_subs_type_t);
```

```
extern size_t snd_seq_query_subscribe_sizeof(void);
extern void snd_seq_queue_status_copy(snd_seq_queue_status_t *,
                                      const
snd_seq_queue_status_t *);
extern void snd_seq_queue_status_free(snd_seq_queue_status_t *);
extern                  const                  snd_seq_real_time_t
*snd_seq_queue_status_get_real_time(const

snd_seq_queue_status_t

*);
extern                                         snd_seq_tick_time_t
snd_seq_queue_status_get_tick_time(const

snd_seq_queue_status_t
                                                                  *);
extern  int  snd_seq_queue_status_malloc(snd_seq_queue_status_t  *
*);
extern size_t snd_seq_queue_status_sizeof(void);
extern void snd_seq_queue_tempo_copy(snd_seq_queue_tempo_t *,
                                     const   snd_seq_queue_tempo_t
*);
extern void snd_seq_queue_tempo_free(snd_seq_queue_tempo_t *);
extern            int            snd_seq_queue_tempo_get_ppq(const
snd_seq_queue_tempo_t *);
extern unsigned int snd_seq_queue_tempo_get_tempo(const

snd_seq_queue_tempo_t *);
extern int snd_seq_queue_tempo_malloc(snd_seq_queue_tempo_t * *);
extern  void  snd_seq_queue_tempo_set_ppq(snd_seq_queue_tempo_t  *,
int);
extern   void   snd_seq_queue_tempo_set_tempo(snd_seq_queue_tempo_t
*,
                                              unsigned int);
extern size_t snd_seq_queue_tempo_sizeof(void);
extern       int       snd_seq_set_client_info(snd_seq_t       *,
snd_seq_client_info_t *);
extern int snd_seq_set_input_buffer_size(snd_seq_t *, size_t);
extern int snd_seq_set_output_buffer_size(snd_seq_t *, size_t);
extern     int     snd_seq_set_port_info(snd_seq_t     *,     int,
snd_seq_port_info_t *);
extern int snd_seq_set_queue_tempo(snd_seq_t *, int,
                                   snd_seq_queue_tempo_t *);
extern       int       snd_seq_subscribe_port(snd_seq_t       *,
snd_seq_port_subscribe_t *);
extern int snd_seq_system_info(snd_seq_t *, snd_seq_system_info_t
*);
extern void snd_seq_system_info_copy(snd_seq_system_info_t *,
                                     const   snd_seq_system_info_t
*);
extern void snd_seq_system_info_free(snd_seq_system_info_t *);
extern          int          snd_seq_system_info_get_clients(const
snd_seq_system_info_t *);
extern           int           snd_seq_system_info_get_ports(const
snd_seq_system_info_t *);
extern          int          snd_seq_system_info_get_queues(const
snd_seq_system_info_t *);
extern int snd_seq_system_info_malloc(snd_seq_system_info_t * *);
extern size_t snd_seq_system_info_sizeof(void);
extern int snd_seq_unsubscribe_port(snd_seq_t *,
                                    snd_seq_port_subscribe_t *);
extern const unsigned int snd_seq_event_types[];
```

6.2.18 alsa/seq_event.h

```
#define SND_SEQ_TIME_STAMP_TICK (0<<0)
#define SND_SEQ_TIME_MODE_ABS   (0<<1)
#define SND_SEQ_EVENT_LENGTH_FIXED      (0<<2)
#define SND_SEQ_PRIORITY_NORMAL (0<<4)
#define SND_SEQ_TIME_STAMP_MASK (1<<0)
#define SND_SEQ_TIME_STAMP_REAL (1<<0)
#define SND_SEQ_TIME_MODE_MASK  (1<<1)
#define SND_SEQ_TIME_MODE_REL   (1<<1)
#define SND_SEQ_EVENT_LENGTH_VARIABLE   (1<<2)
#define SND_SEQ_PRIORITY_HIGH   (1<<4)
#define SND_SEQ_PRIORITY_MASK   (1<<4)
#define SND_SEQ_EVENT_LENGTH_VARUSR     (2<<2)
#define SND_SEQ_EVENT_LENGTH_MASK       (3<<2)

typedef struct snd_seq_addr {
    unsigned char client;
    unsigned char port;
} snd_seq_addr_t;
typedef struct snd_seq_connect {
    snd_seq_addr_t sender;
    snd_seq_addr_t dest;
} snd_seq_connect_t;
typedef struct snd_seq_ev_cluster {
    snd_seq_instr_cluster_t cluster;
} snd_seq_ev_cluster_t;
typedef struct snd_seq_ev_ctrl {
    unsigned char channel;
    unsigned char unused[3];
    unsigned int param;
    int value;
} snd_seq_ev_ctrl_t;
typedef struct snd_seq_ev_ext {
    unsigned int len;
    void *ptr;
} __attribute__ (packed)
    snd_seq_ev_ext_t;
typedef struct snd_seq_ev_instr_begin {
    int timeout;
} snd_seq_ev_instr_begin_t;
typedef struct snd_seq_ev_loop {
    unsigned int start;
    unsigned int end;
} snd_seq_ev_loop_t;
typedef struct snd_seq_ev_note {
    unsigned char channel;
    unsigned char note;
    unsigned char velocity;
    unsigned char off_velocity;
    unsigned int duration;
} snd_seq_ev_note_t;
typedef struct snd_seq_ev_queue_control {
    unsigned char queue;
    unsigned char unused[3];
    union {
        int value;
        snd_seq_timestamp_t time;
        unsigned int position;
        snd_seq_queue_skew_t skew;
        unsigned int d32[2];
        unsigned char d8[8];
    } param;
} snd_seq_ev_queue_control_t;
typedef struct snd_seq_ev_raw32 {
    unsigned int d[3];
} snd_seq_ev_raw32_t;
typedef struct snd_seq_ev_raw8 {
```

```
    unsigned char d[12];
} snd_seq_ev_raw8_t;
typedef struct snd_seq_ev_sample_control {
    unsigned char channel;
    unsigned char unused[3];
    union {
        snd_seq_ev_sample_t sample;
        snd_seq_ev_cluster_t cluster;
        snd_seq_position_t position;
        snd_seq_stop_mode_t stop_mode;
        snd_seq_frequency_t frequency;
        snd_seq_ev_volume_t volume;
        snd_seq_ev_loop_t loop;
        unsigned char raw8[8];
    } param;
} snd_seq_ev_sample_control_t;
typedef struct snd_seq_ev_sample {
    unsigned int std;
    short unsigned int bank;
    short unsigned int prg;
} snd_seq_ev_sample_t;
typedef struct snd_seq_ev_volume {
    short int volume;
    short int lr;
    short int fr;
    short int du;
} snd_seq_ev_volume_t;
typedef struct snd_seq_event {
    snd_seq_event_type_t type;
    unsigned char flags;
    unsigned char tag;
    unsigned char queue;
    snd_seq_timestamp_t time;
    snd_seq_addr_t source;
    snd_seq_addr_t dest;
    union {
        snd_seq_ev_note_t note;
        snd_seq_ev_ctrl_t control;
        snd_seq_ev_raw8_t raw8;
        snd_seq_ev_raw32_t raw32;
        snd_seq_ev_ext_t ext;
        snd_seq_ev_queue_control_t queue;
        snd_seq_timestamp_t time;
        snd_seq_addr_t addr;
        snd_seq_connect_t connect;
        snd_seq_result_t result;
        snd_seq_ev_instr_begin_t instr_begin;
        snd_seq_ev_sample_control_t sample;
    } data;
} snd_seq_event_t;
typedef unsigned char snd_seq_event_type_t;
typedef int snd_seq_frequency_t;
typedef unsigned int snd_seq_instr_cluster_t;
typedef struct snd_seq_instr {
    snd_seq_instr_cluster_t cluster;
    unsigned int std;
    short unsigned int bank;
    short unsigned int prg;
} snd_seq_instr_t;
typedef unsigned int snd_seq_position_t;
typedef struct snd_seq_queue_skew {
    unsigned int value;
    unsigned int base;
} snd_seq_queue_skew_t;
union snd_seq_timestamp {
    snd_seq_tick_time_t tick;
```

```
    struct snd_seq_real_time time;
};
typedef struct snd_seq_real_time {
    unsigned int tv_sec;
    unsigned int tv_nsec;
} snd_seq_real_time_t;
typedef struct snd_seq_result {
    int event;
    int result;
} snd_seq_result_t;
typedef enum snd_seq_stop_mode {
    SND_SEQ_SAMPLE_STOP_IMMEDIATELY,
    SND_SEQ_SAMPLE_STOP_VENVELOPE,
    SND_SEQ_SAMPLE_STOP_LOOP
} snd_seq_stop_mode_t;
typedef unsigned int snd_seq_tick_time_t;
typedef union snd_seq_timestamp {
    snd_seq_tick_time_t tick;
    struct snd_seq_real_time time;
} snd_seq_timestamp_t;
```

6.2.19 alsa/seq_midi_event.h

```
typedef struct snd_midi_event snd_midi_event_t;
extern   long   int   snd_midi_event_decode(snd_midi_event_t   *,
unsigned char *,
                                      long       int,       const
snd_seq_event_t *);
extern long int snd_midi_event_encode(snd_midi_event_t *,
                                      const unsigned char *, long
int,
                                      snd_seq_event_t *);
extern int snd_midi_event_encode_byte(snd_midi_event_t *, int,
                                      snd_seq_event_t *);
extern void snd_midi_event_free(snd_midi_event_t *);
extern void snd_midi_event_init(snd_midi_event_t *);
extern int snd_midi_event_new(size_t, snd_midi_event_t * *);
extern void snd_midi_event_reset_decode(snd_midi_event_t *);
extern void snd_midi_event_reset_encode(snd_midi_event_t *);
```

6.2.20 alsa/seqmid.h

```
#define snd_seq_ev_set_dest(ev,c,p)      \
        ((ev)->dest.client = (c), (ev)->dest.port = (p))
#define snd_seq_ev_set_broadcast(ev)     \
        ((ev)->dest.client   =   SND_SEQ_ADDRESS_BROADCAST,   (ev)-
>dest.port = \
        SND_SEQ_ADDRESS_BROADCAST)
#define snd_seq_ev_set_subs(ev)  \
        ((ev)->dest.client  =  SND_SEQ_ADDRESS_SUBSCRIBERS,  (ev)-
>dest.port = \
        SND_SEQ_ADDRESS_UNKNOWN)
#define snd_seq_ev_set_fixed(ev)         \
        ((ev)->flags  &=  ~SND_SEQ_EVENT_LENGTH_MASK,  (ev)->flags
|= \
        SND_SEQ_EVENT_LENGTH_FIXED)
#define snd_seq_ev_set_chanpress(ev,ch,val)      \
        ((ev)->type           =           SND_SEQ_EVENT_CHANPRESS,
snd_seq_ev_set_fixed(ev), \
        (ev)->data.control.channel       =       (ch),       (ev)-
>data.control.value = (val))
#define snd_seq_ev_set_controller(ev,ch,cc,val)  \
        ((ev)->type          =          SND_SEQ_EVENT_CONTROLLER,
snd_seq_ev_set_fixed(ev), \
```

```
        (ev)->data.control.channel      =       (ch),       (ev)-
>data.control.param = (cc), \
        (ev)->data.control.value = (val))
#define snd_seq_ev_set_keypress(ev,ch,key,vel)   \
        ((ev)->type           =           SND_SEQ_EVENT_KEYPRESS,
snd_seq_ev_set_fixed(ev), \
        (ev)->data.note.channel  =  (ch),  (ev)->data.note.note  =
(key), \
        (ev)->data.note.velocity = (vel))
#define snd_seq_ev_set_pgmchange(ev,ch,val)      \
        ((ev)->type           =           SND_SEQ_EVENT_PGMCHANGE,
snd_seq_ev_set_fixed(ev), \
        (ev)->data.control.channel      =       (ch),       (ev)-
>data.control.value = (val))
#define snd_seq_ev_set_pitchbend(ev,ch,val)      \
        ((ev)->type           =           SND_SEQ_EVENT_PITCHBEND,
snd_seq_ev_set_fixed(ev), \
        (ev)->data.control.channel      =       (ch),       (ev)-
>data.control.value = (val))
#define  snd_seq_ev_set_direct(ev)               ((ev)->queue  =
SND_SEQ_QUEUE_DIRECT)
#define snd_seq_ev_set_source(ev,p)     ((ev)->source.port = (p))
#define snd_seq_ev_set_tag(ev,t)        ((ev)->tag = (t))
#define    snd_seq_ev_clear(ev)                 memset(ev,    0,
sizeof(snd_seq_event_t))

extern int snd_seq_connect_from(snd_seq_t *, int, int, int);
extern int snd_seq_connect_to(snd_seq_t *, int, int, int);
extern int snd_seq_control_queue(snd_seq_t *, int, int, int,
                                 snd_seq_event_t *);
extern int snd_seq_create_simple_port(snd_seq_t *, const char *,
                                      unsigned    int,    unsigned
int);
extern int snd_seq_delete_simple_port(snd_seq_t *, int);
extern int snd_seq_disconnect_from(snd_seq_t *, int, int, int);
extern int snd_seq_disconnect_to(snd_seq_t *, int, int, int);
extern int snd_seq_parse_address(snd_seq_t *, snd_seq_addr_t *,
                                 const char *);
extern int snd_seq_set_client_name(snd_seq_t *, const char *);
extern int snd_seq_sync_output_queue(snd_seq_t *);
```

6.2.21 alsa/timer.h

```
#define SND_TIMER_OPEN_NONBLOCK (1<<0)
#define SND_TIMER_OPEN_TREAD    (1<<1)
#define SND_TIMER_GLOBAL_SYSTEM 0
#define SND_TIMER_GLOBAL_RTC    1
#define SND_TIMER_GLOBAL_HPET   2

typedef struct sndrv_timer_ginfo snd_timer_ginfo_t;
typedef struct sndrv_timer_gparams snd_timer_gparams_t;
typedef struct sndrv_timer_gstatus snd_timer_gstatus_t;
typedef struct sndrv_timer_id snd_timer_id_t;
typedef struct sndrv_timer_info snd_timer_info_t;
typedef struct sndrv_timer_params snd_timer_params_t;
typedef struct _snd_timer_query snd_timer_query_t;
typedef struct sndrv_timer_status snd_timer_status_t;
typedef struct _snd_timer snd_timer_t;
typedef enum _snd_timer_type {
    SND_TIMER_TYPE_HW,
    SND_TIMER_TYPE_SHM,
    SND_TIMER_TYPE_INET
} snd_timer_type_t;
extern int snd_timer_close(snd_timer_t *);
extern int snd_timer_continue(snd_timer_t *);
```

```
extern    void    snd_timer_id_copy(snd_timer_id_t    *,    const
snd_timer_id_t *);
extern void snd_timer_id_free(snd_timer_id_t *);
extern int snd_timer_id_get_card(snd_timer_id_t *);
extern int snd_timer_id_get_class(snd_timer_id_t *);
extern int snd_timer_id_get_device(snd_timer_id_t *);
extern int snd_timer_id_get_sclass(snd_timer_id_t *);
extern int snd_timer_id_get_subdevice(snd_timer_id_t *);
extern int snd_timer_id_malloc(snd_timer_id_t * *);
extern void snd_timer_id_set_card(snd_timer_id_t *, int);
extern void snd_timer_id_set_class(snd_timer_id_t *, int);
extern void snd_timer_id_set_device(snd_timer_id_t *, int);
extern void snd_timer_id_set_sclass(snd_timer_id_t *, int);
extern void snd_timer_id_set_subdevice(snd_timer_id_t *, int);
extern size_t snd_timer_id_sizeof(void);
extern int snd_timer_info(snd_timer_t *, snd_timer_info_t *);
extern void snd_timer_info_copy(snd_timer_info_t *,
                                const snd_timer_info_t *);
extern void snd_timer_info_free(snd_timer_info_t *);
extern int snd_timer_info_get_card(snd_timer_info_t *);
extern const char *snd_timer_info_get_id(snd_timer_info_t *);
extern const char *snd_timer_info_get_name(snd_timer_info_t *);
extern  long  int  snd_timer_info_get_resolution(snd_timer_info_t
*);
extern int snd_timer_info_malloc(snd_timer_info_t * *);
extern size_t snd_timer_info_sizeof(void);
extern int snd_timer_open(snd_timer_t * *, const char *, int);
extern int snd_timer_params(snd_timer_t *, snd_timer_params_t *);
extern long int snd_timer_params_get_ticks(snd_timer_params_t *);
extern int snd_timer_params_malloc(snd_timer_params_t * *);
extern  int  snd_timer_params_set_auto_start(snd_timer_params_t  *,
int);
extern void snd_timer_params_set_ticks(snd_timer_params_t *, long
int);
extern  int   snd_timer_poll_descriptors(snd_timer_t   *,   struct
pollfd *,
                                         unsigned int);
extern int snd_timer_poll_descriptors_count(snd_timer_t *);
extern ssize_t snd_timer_read(snd_timer_t *, void *, size_t);
extern int snd_timer_start(snd_timer_t *);
extern int snd_timer_status(snd_timer_t *, snd_timer_status_t *);
extern int snd_timer_stop(snd_timer_t *);
```

III Trial Use Module

7 Trial Use Module

7.1 Introduction

The Trial Use Module decribes components in Trial Use status. Trial Use Specifications are non-mandatory components of the Linux Standard Base.

7.2 Xdg-utils

Xdg-utils is a set of command line utilities that assist applications with a variety of desktop integration tasks. Some of the utilities focus on tasks commonly required during the installation of a desktop application. The remainder focus on integration with the desktop environment while the application is running.

These utilities operate as described at xdg-utils reference

7.2.1 Xdg-utils Commands

An LSB conforming implementation shall provide the commands and utilities as described in Table 7-1, with at least the behavior described as mandatory in the referenced underlying specification, with the following exceptions:

1. If any operand (except one which follows `--`) starts with a hyphen, the behavior is unspecified.

Rationale (Informative): Applications should place options before operands, or use `--`, as needed. This text is needed because, by default, GNU option parsing differs from POSIX, unless the environment variable POSIXLY_CORRECT is set. For example, **ls . -a** in GNU **ls** means to list the current directory, showing all files (that is, `"."` is an operand and `-a` is an option). In POSIX, `"."` and `-a` are both operands, and the command means to list the current directory, and also the file named `-a`. Suggesting that applications rely on the setting of the POSIXLY_CORRECT environment variable, or try to set it, seems worse than just asking the applications to invoke commands in ways which work with either the POSIX or GNU behaviors.

Table 7-1 Commands And Utilities

xdg-desktop-icon [1]	xdg-email [1]	xdg-mime [1]	xdg-screensaver [1]	
xdg-desktop-menu [1]	xdg-icon-resource [1]	xdg-open [1]		

Referenced Specification(s)

[1]. xdg-utils reference

Annex A GNU Free Documentation License (Informative)

This specification is published under the terms of the GNU Free Documentation License, Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

A.1 PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.2 APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A.3 VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.4 COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.5 MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.6 COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the

name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

A.7 COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.8 AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

A.9 TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

A.10 TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or

rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.11 FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

A.12 How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.