Speech Processing interface

v1.0.0

Generated by Doxygen 1.6.3

Fri Jun 29 10:03:15 2012

Contents

1 Main Page **1**

2 Data Structure Documentation **10**

2.1 speech_proc_frame_info_t Struct Reference 10

2.1.1 Detailed Description 10

2.1.2 Field Documentation 10

2.2 speech_proc_portsetting_t Struct Reference 10

2.2.1 Detailed Description 10

2.2.2 Field Documentation 10

2.3 speech_proc_setting_t Struct Reference 11

2.3.1 Detailed Description 11

2.3.2 Field Documentation 11

2.4 SpeechProcLibrary Class Reference 11

2.4.1 Detailed Description 12

2.4.2 Constructor & Destructor Documentation 12

2.4.3 Member Function Documentation 12

3 File Documentation **17**

3.1 speech_proc_interface.h File Reference 17

3.1.1 Detailed Description 18

3.1.2 Typedef Documentation 18

3.1.3 Enumeration Type Documentation 19

3.1.4 Function Documentation 20

3.2 speech_proc_interface.txt File Reference 25

3.3 speech_proc_omx_interface.h File Reference 25

3.3.1 Detailed Description 25

3.3.2 Enumeration Type Documentation 25

3.3.3 Function Documentation 26

1 Main Page

Introduction

This documentation aims to describe both interfaces (Processing and OMX) to be impleted by one library in order to be used as processing library inside ST-Ericsson OMX Speech Processing component.

Context

The speech proc library will be used as the algorithm part of ST-Ericsson's implementattion of OMX Speech Processing Component.

This OMX component (OMX.ST.AFM.speech_proc) contains 5 audio ports and is made of 2 main parts (as almost all ST-Ericsson OMX components):

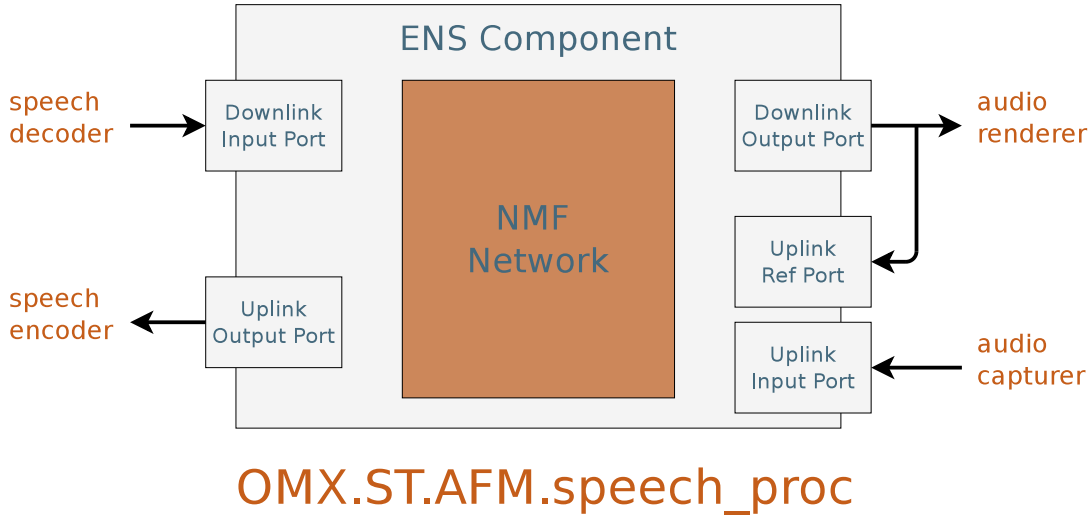


Figure 1: speech_proc OMX component

- An ENS Component :

  This part implements all the OMX stuff (State machine, command ...). For complete details refer to ENS documentation.

- A Processing Component :

  This, as implied by its name, implements the processing part based on a network of NMF components.

  In our case, this NMF network is made of 3 main NMF components (plus other miscellaneous ones, not mentioned here for clarity. Refer to speech_proc documentation for more detail).

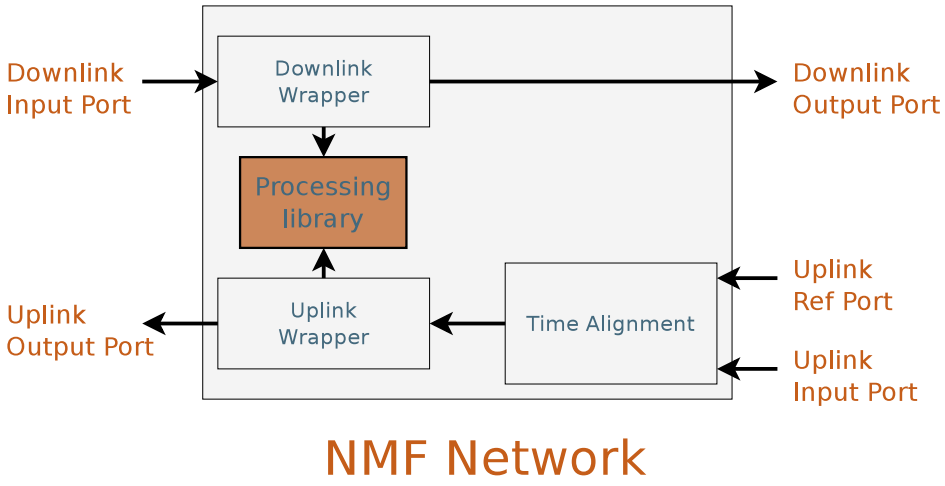


Figure 2: NMF network of speech_proc component

- Uplink Wrapper :
  This component reads data from time alignment component, call speech processing library and write output buffer on uplink output port.
- Downlink wrapper :
  This component reads data from downlink input port, call speech processing library and write output buffer on downlink output port.
- Time alignment :
  This component reads data from uplink input port and uplink reference port, align them temporally and give them to uplink wrapper.
  It will also resample the reference port data to the uplink port frequency, and "pack" both input (uplink + reference) into one buffer.

Shared library

Both interfaces have to be implemented in a .so file called **libspeech_processing.so**.

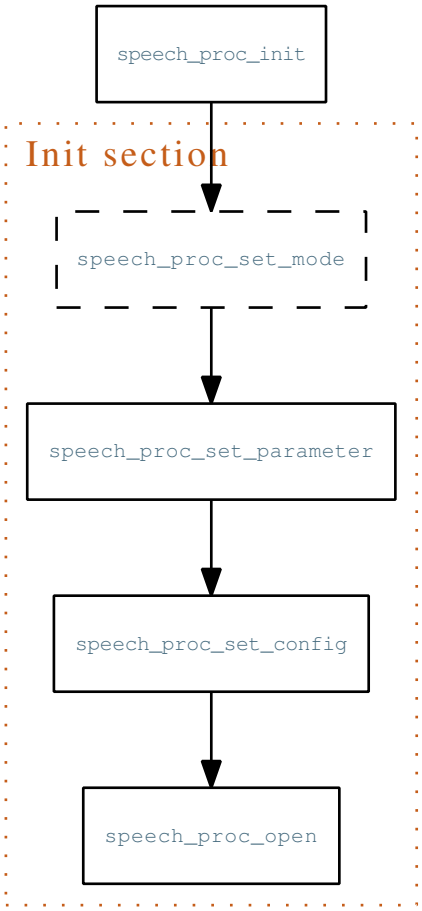
This library will be loaded when the speech_proc_component is loaded.

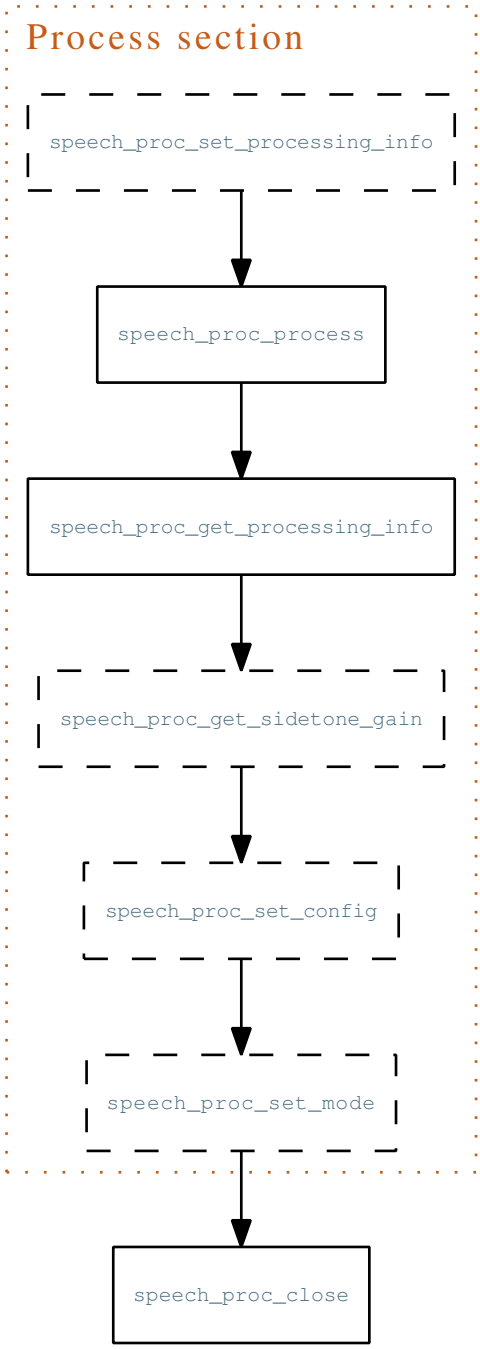
Processing Interface description

The processing interface is composed of several C functions. The init function is responsible to allocate the library instance context, and then all other interface functions will be called with this context as parameter.

Call sequence

The call sequence of interface functions is as follow (click in the graph to get detailled description of one function):


speech_proc_init
Init section
speech_proc_set_mode
speech_proc_set_parameter
speech_proc_set_config
speech_proc_open



Init section

- speech_proc_set_mode is not call during init if port are not enabled (and then not configured) when going to idle.

Process section

- The process section is repeated for each input frame.

- The speech_proc_set_processing_info function may be called or not and not necessarily in the same order.

  Nevertheless when call is established the sequence should be respected.

- speech_proc_set_mode will be called eacht time port (at OMX level) are reconfigured.
- speech_proc_set_config may be called at any time an undetermined number of times (including zero).
- speech_proc_get_sidetone_gain will no longer be called if library return SP_ERROR_UNSUPPORTED.

For a complete description of the processing API, check here.

OMX Interface Description

As different speech processing library will require different configuration paramaters, a second interface has been added. This interface is at OMX level and is supposed to "convert" OMX indexes and configuration structures into processing library structures. This interface is used for OMX configuration calls (i.e. OMX_SetConfig, OMX_SetParameter, OMX_GetConfig and OMX_GetParameter) and when OMX component is passing from OMX_StateLoaded to OMX_StateIdle, since speech_proc library is actually instantiated when leaving OMX_StateLoaded state.

SpeechProcLibrary Class

As all proxy code is written in C++, OMX interface is defined as an abstract class.

Full details on the class definition is available here

Factory Method

In order to keep proxy completely independant, the library should also provide a function speech_proc_getOMXInterface that will return a pointer of the class implementing the OMX interface. Thus proxy does not have to know the definition of the subclass and then changing the libspeech_processing.so is enough to change the processing library.

Call sequence of OMX interface

You can find here after call sequence of the OMX interface.

Before reading the following graphs please note the following points:

- do NOT except respect of some graph standard (sorry for that, i don't know them), nevertheless I use mine that is :
  - normal arrows are method calls.
  - dotted arrows are returned of method.
  - double arrows are function calls trough NMF interfaces.
  - UL & DL means that method is called twice once for Downlink algorithms and once for Uplink ones.
  - When same parameter name is used for 2 method (or NMF itf) calls it means that this is the same value.

- SpeechProcComponent and SpeechProcNmf_ProcessincComp are the two main classes that compose what we call ENS Component here
- There is **NO** direct connection between SpeechProcNmf_ProcessincComp and speech_proc_library. Even if graphs do not show it, connection actually pass through NMF components before reaching processing library. They have been omitted for clarity.

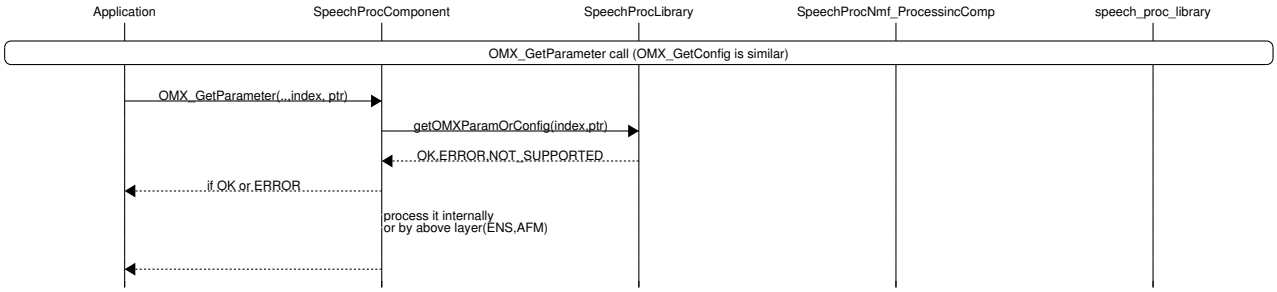


Figure 3: sequence for OMX_GetParameter and OMX_GetConfig call

OMX_GetParameter and OMX_GetConfig

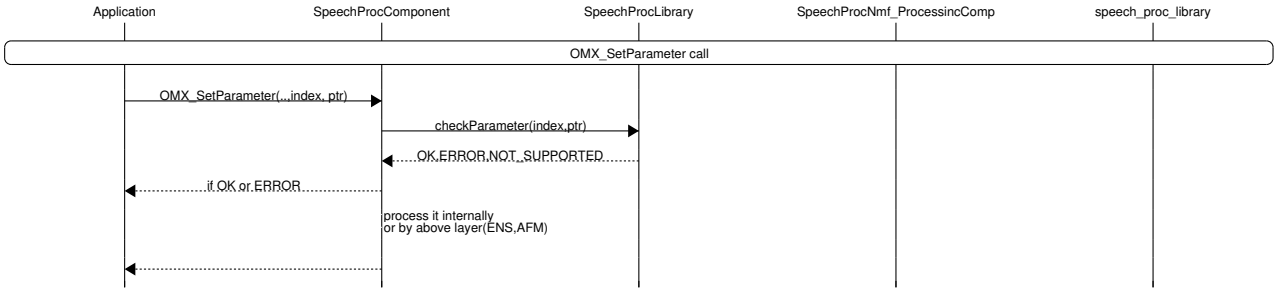


Figure 4: sequence for OMX_SetParameter call

OMX_SetParameter

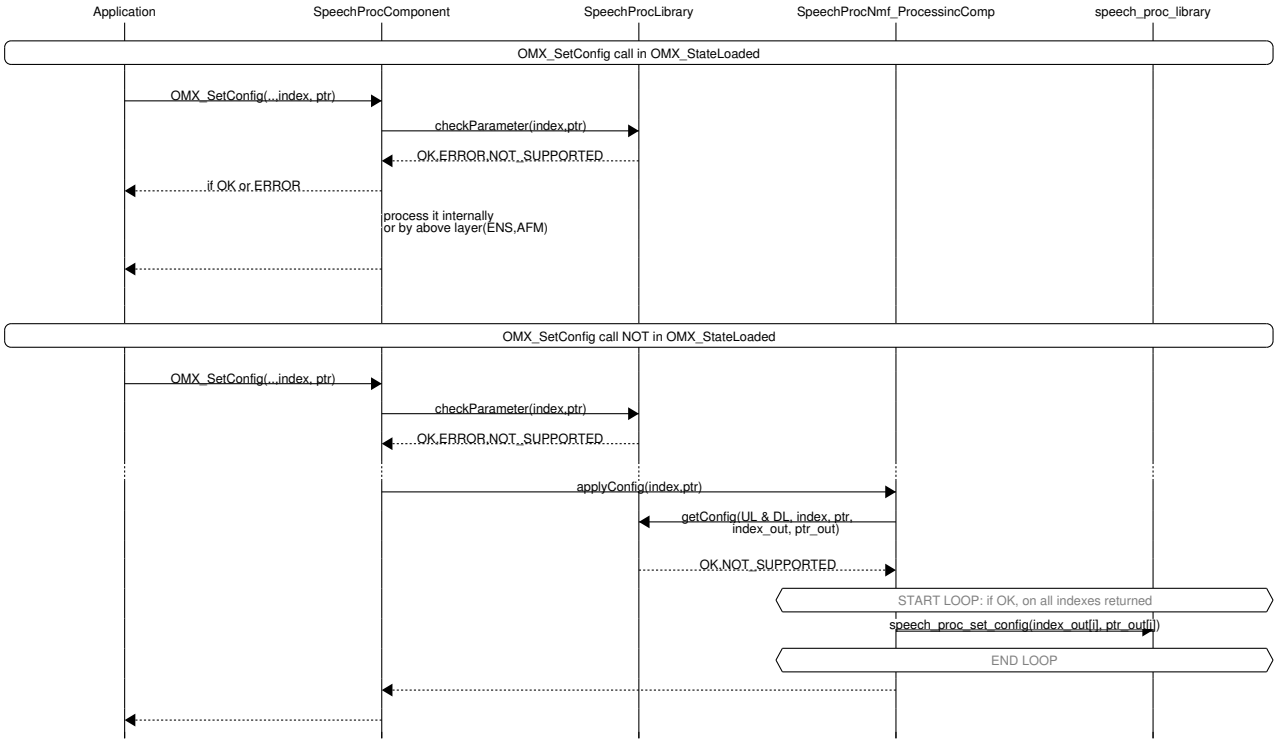


Figure 5: sequence for OMX_SetConfig call

OMX_SetConfig

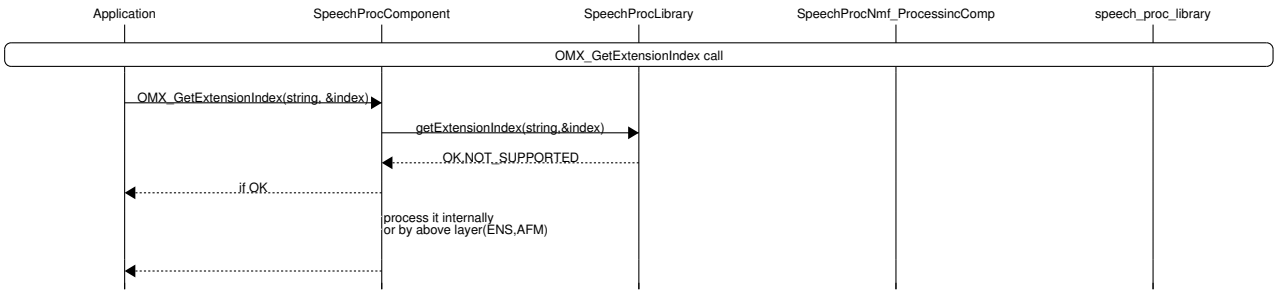


Figure 6: sequence forOMX_GetExtensionIndex call

OMX_GetExtensionIndex

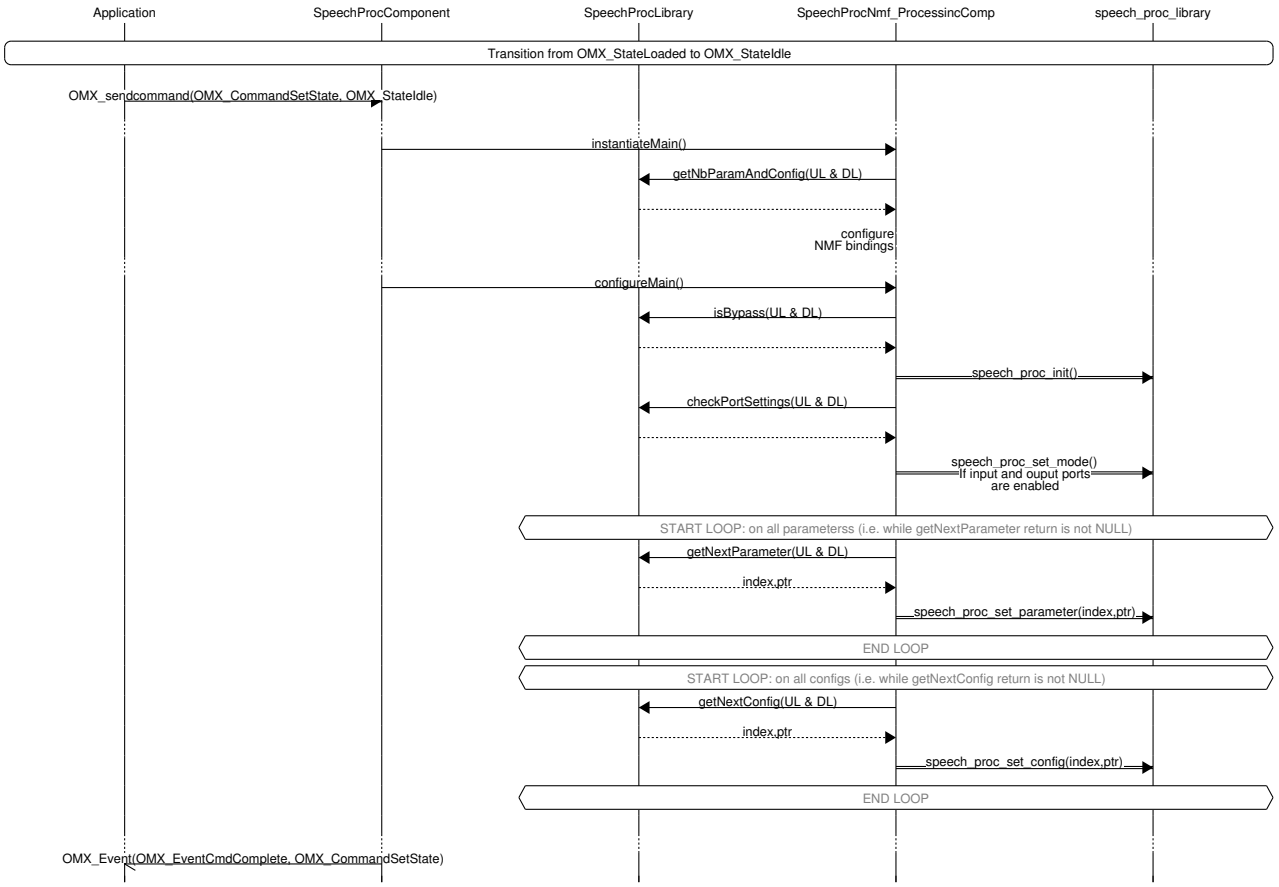


Figure 7: sequence for transition form OMX_StateLoaded to OMX_StateIdle

Transition from OMX_StateLoaded to OMX_StateIdle

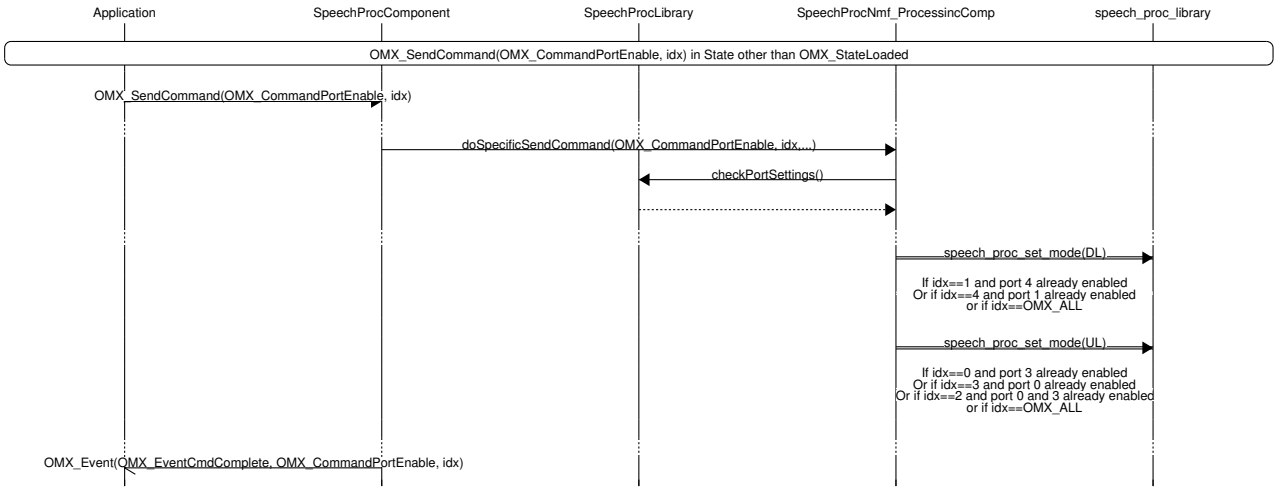


Figure 8: sequence for port enabling

Enabling port in state other than OMX_StateIdle

2 Data Structure Documentation

2.1 speech_proc_frame_info_t Struct Reference

`#include <speech_proc_interface.h>`

Data Fields

- unsigned char FrameInfo [SPEECH_PROC_INFO_ELEMENT_NUMBER_OF]

2.1.1 Detailed Description

Frame information

2.1.2 Field Documentation

2.1.2.1 unsigned char speech_proc_frame_info_t::FrameInfo[SPEECH_PROC_INFO_ELEMENT_NUMBER_OF]

Sequence of setting

2.2 speech_proc_portsetting_t Struct Reference

`#include <speech_proc_interface.h>`

Data Fields

- struct {
  unsigned int enabled
  unsigned int interleaved
  unsigned int nb_channels
  unsigned int samplerate
  } port [2]

- unsigned int nb_ports

2.2.1 Detailed Description

Define settings

2.2.2 Field Documentation

2.2.2.1 unsigned int speech_proc_portsetting_t::enabled

If not set, then library should not take into account next info

2.2.2.2 unsigned int speech_proc_portsetting_t::interleaved

0 means flat, 1 means interleaved. (Only flat is currently supported)

2.2.2.3 unsigned int speech_proc_portsetting_t::nb_channels

Number of channels

2.2.2.4 unsigned int speech_proc_portsetting_t::nb_ports

2.2.2.5 struct { ... } speech_proc_portsetting_t::port[2]

2.2.2.6 unsigned int speech_proc_portsetting_t::samplerate

8000 or 16000

2.3 speech_proc_setting_t Struct Reference

`#include <speech_proc_interface.h>`

Data Fields

- unsigned int resolution
- unsigned int framesize

2.3.1 Detailed Description

Define buffer size, sample size and sample rate

2.3.2 Field Documentation

2.3.2.1 unsigned int speech_proc_setting_t::framesize

Number of samples in each execution. May be 1,2,4,5,10 or 20 ms (only 10ms for currently) Each 1 ms contains samplerate/1000 so 10 ms in 8kHz is 80 and in 16kHz 160

2.3.2.2 unsigned int speech_proc_setting_t::resolution

bit per sample, only 16 currently supported

2.4 SpeechProcLibrary Class Reference

`#include <speech_proc_omx_interface.h>`

Public Member Functions

- SpeechProcLibrary ()
- virtual ~SpeechProcLibrary ()
- virtual OMX_U32 getNbParamAndConfig (speech_proc_algo_type_t algo)=0
- virtual bool isBypass (speech_proc_algo_type_t algo)=0
- virtual OMX_ERRORTYPE checkParameter (OMX_U32 index, OMX_PTR param)=0
- virtual OMX_PTR getNextParameter (speech_proc_algo_type_t algo, OMX_U32 previous_index, OMX_U32 *current_index)=0
- virtual OMX_ERRORTYPE checkConfig (OMX_U32 index, OMX_PTR config)=0
- virtual OMX_PTR getNextConfig (speech_proc_algo_type_t algo, OMX_U32 previous_index, OMX_U32 *current_index)=0
- virtual OMX_ERRORTYPE getConfig (speech_proc_algo_type_t algo, OMX_U32 index, OMX_-PTR *config_table, OMX_U32 *index_table, OMX_U32 *nb_config)=0
- virtual OMX_ERRORTYPE getOMXParamOrConfig (OMX_U32 index, OMX_PTR config)=0
- virtual OMX_ERRORTYPE checkPortSettings (OMX_AUDIO_PARAM_PCMMODETYPE *portsettings[SPEECH_PROC_NB_AUDIO_PORT], OMX_U32 portenabled)=0
- virtual OMX_ERRORTYPE getExtensionIndex (OMX_STRING extension, OMX_U32 *index)=0
- void setLibCtx (void *ctx, speech_proc_algo_type_t algo)

Protected Member Functions

- void * getLibCtx (speech_proc_algo_type_t algo)

2.4.1 Detailed Description

Abstract Class

2.4.2 Constructor & Destructor Documentation

2.4.2.1 SpeechProcLibrary::SpeechProcLibrary () `[inline]`

2.4.2.2 virtual SpeechProcLibrary::~SpeechProcLibrary () `[inline, virtual]`

2.4.3 Member Function Documentation

2.4.3.1 virtual OMX_ERRORTYPE SpeechProcLibrary::checkConfig (OMX_U32 *index*, OMX_PTR *config*) `[pure virtual]`

This function is called on each call to OMX_SetConfig interface of the OMX.ST.AFM.speech_proc component.

- If the index is not related to speech processing library configuration, the function must return OMX_-ErrorUnsupportedIndex and the configuration will be handled by above layers.

- If the index is related to speech processing configuration, the function should check the content of the structure, return OMX_ErrorUnsupportedSetting if it is not correct or save it locally and return OMX_ErrorNone.

Parameters

← ***index*** OMX index

← ***config*** pointer on the configuration structure related to index

Returns

OMX_ErrorNone, OMX_ErrorUnsupportedIndex or OMX_ErrorUnsupportedSetting

2.4.3.2 virtual OMX_ERRORTYPE SpeechProcLibrary::checkParameter (OMX_U32 *index*, OMX_PTR *param*) `[pure virtual]`

This function is called on each call to OMX_SetParameter interface of the OMX.ST.AFM.speech_proc component.

- If the index is not related to speech processing library configuration, the function must return OMX_-ErrorUnsupportedIndex and the configuration will be handled by above layers.
- If the index is related to speech processing configuration, the function should check the content of the structure, return OMX_ErrorUnsupportedSetting if it is not correct or save it locally and return OMX_ErrorNone.

Parameters

← ***index*** OMX index

← ***param*** pointer on the configuration structure related to index

Returns

OMX_ErrorNone, OMX_ErrorUnsupportedIndex or OMX_ErrorUnsupportedSetting

2.4.3.3 virtual OMX_ERRORTYPE SpeechProcLibrary::checkPortSettings (OMX_AUDIO_-PARAM_PCMMODETYPE ∗ *portsettings*[SPEECH_PROC_NB_AUDIO_PORT], OMX_U32 *portenabled*) `[pure virtual]`

Function called each time the OMX component goes from OMX_LoadedState to OMX_IdleState (i.e. when we create the speech_proc library) and when a port is enabled. As all ports may not be enabled (i.e. not configured) when this function is called, the library should only check port marked as enabled with the **portenabled** bitfield. This function has been added because it is not easy to return an error to the application in speech_proc_set_mode() function.

NOTE : Parameters pass to this function do NOT take into account the samplerate converter embedded inside time alignment.

Parameters

← ***portsettings*** settings for all audio ports. (port_settings[INPUT_UPLINK_PORT] ... port_-settings[OUTPUT_DOWNLINK_PORT])

← ***portenabled*** Bitfield indicating if a port is enabled. (e.g (portenable & (1<<INPUT_UPLINK_-PORT)) means that INPUT_UPLINK_PORT is enabled)

Returns

OMX_ErrorNone if settings is OK, and OMX_ErrorBadParameter otherwise

2.4.3.4 virtual OMX_ERRORTYPE SpeechProcLibrary::getConfig (speech_proc_algo_type_t *algo*, OMX_U32 *index*, OMX_PTR * *config_table*, OMX_U32 * *index_table*, OMX_U32 * *nb_config*) [pure virtual]

Function called on each call of the OMX_SetConfig interface of the OMX.ST.AFM.speech_proc component, if the component is not in OMX_LoadedState.

Because it is not easy to get the result of checkConfig when this function is called NULL should be returned if the index is not related to configuraion of the specify type of algo (downlink or uplink).

As one OMX index may result in several library configuration, the fuction should return a table of structure and indexes. (Tables have been allocated by proxy and the size is the one returned by getNbParamAnd-Config()).

Parameters

← ***algo*** type of Algo (Uplink or Downlink)

← ***index*** OMX index

→ ***config_table*** table of configuration structure that will be send to library

→ ***index_table*** table of index that will be send to library

→ ***nb_config*** number of config to send to library

Returns

OMX_ErrorNone, OMX_ErrorUnsupportedIndex

2.4.3.5 virtual OMX_ERRORTYPE SpeechProcLibrary::getExtensionIndex (OMX_STRING *extension*, OMX_U32 * *index*) [pure virtual]

Function called on each call to the OMX_GetExtensionIndex interface of the OMX.ST.AFM.speech_proc component.

Parameters

← ***extension*** String containing extension name

→ ***index*** OMX index corresponding to the extension

Returns

OMX_ErrorUnsupportedIndex if extension is not supported, OMX_ErrorNone otherwise

2.4.3.6 void* SpeechProcLibrary::getLibCtx (speech_proc_algo_type_t *algo*) [inline, protected]

Function to retrieve library context, if (for very specific case) you need to directly call the library. Typically in function getOMXParamOrConfig you may need to get information directly form the library.

Attention

Directly calling the library may not be safe: You have not guarantee that it is not currently running, unless you specificallky add some synchro.

Parameters

← ***algo*** Library instance (Uplink or Downlink)

Returns

library context pointer

2.4.3.7 virtual OMX_U32 SpeechProcLibrary::getNbParamAndConfig (speech_proc_algo_type_t *algo*) [pure virtual]

Return the number of different config and parameter indexes supported by the processing library. This is not necessarly the same as OMX indexes. This value will be used to size the NMF fifo between proxy and NMF components.

Parameters

← ***algo*** type of Algo (Uplink or Downlink)

Returns

number of indexes (Parameter and Config) supported by the processing library

2.4.3.8 virtual OMX_PTR SpeechProcLibrary::getNextConfig (speech_proc_algo_type_t *algo*, OMX_U32 *previous_index*, OMX_U32 * *current_index*) [pure virtual]

Function called to retrieve ALL configs to set on speech_proc library one by one. This function is called each time the OMX component goes from OMX_LoadedState to OMX_IdleState (i.e. when we create the speech_proc library). The struture returned should always be available (eg not allocated in the stack for example).

This function will be used like this :

```
OMX_U32 previous_index = 0;
OMX_U32 current_index = 0;
OMX_PTR param = NULL

param = getNextConfig(algo,previous_index, &current_index);
while(param)
{
  // Actually this is NOT a direct call to speech_proc library
  // but it will end to this.
  speech_proc_set_config(ctx,current_index,param)
  previous_index = current_index;
  param = getNextConfig(algo,previous_index, &current_index);
}
```

Parameters

← ***algo*** type of Algo (Uplink or Downlink)

← ***previous_index*** last index retrieved (0 on first call)

→ ***current_index*** value of the current index retreived

Returns

pointer of config structure to be passed to speech_proc library. NULL when all config have been provided (i.e. when previous_index is the last index supported).

2.4.3.9 virtual OMX_PTR SpeechProcLibrary::getNextParameter (speech_proc_algo_type_t *algo*, OMX_U32 *previous_index*, OMX_U32 * *current_index*) [pure virtual]

Function called to retrieve ALL parameters to set on speech_proc library one by one. This function is called each time the OMX component goes from OMX_LoadedState to OMX_IdleState (i.e. when we create the speech_proc library). The struture returned should always be available (e.g. not allocated in the stack for example).

This function will be used like this :

```
OMX_U32 previous_index = 0;
OMX_U32 current_index = 0;
OMX_PTR param = NULL

param = getNextParameter(algo,previous_index, &current_index);
while(param)
{
  // Actually this is NOT a direct call to speech_proc library
  // but it will end to this.
  speech_proc_set_parameter(ctx,current_index,param)
  previous_index = current_index;
  param = getNextParameter(algo,previous_index, &current_index);
}
```

Parameters

← ***algo*** type of Algo (Uplink or Downlink)

← ***previous_index*** last index retrieved (0 on first call)

→ ***current_index*** value of the current index retreived

Returns

pointer of parameter structure to be passed to speech_proc library. NULL when all parameters have been provided (i.e. when previous_index is the last index supported).

2.4.3.10 virtual OMX_ERRORTYPE SpeechProcLibrary::getOMXParamOrConfig (OMX_U32 *index*, OMX_PTR *config*) [pure virtual]

Function called on each call to the OMX_GetParameter/OMX_GetConfig interface of the OMX.ST.AFM.speech_proc component. Function should copy its local config into provided pointer. (This is above layer resposability to allocate enough memory)

Parameters

← ***index*** OMX index

→ ***config*** pointer on free memory to be filled with required config. (class should consider that memory size is big enough to write the requested config)

Returns

OMX_ErrorNone, OMX_ErrorUnsupportedIndex

2.4.3.11 virtual bool SpeechProcLibrary::isBypass (speech_proc_algo_type_t *algo*) [pure virtual]

Return true if bypass mode should be activated for this type of algo. (For debug purpose only)

Parameters

← ***algo*** type of Algo (Uplink or Downlink)

Returns

true to activate bypass mode

2.4.3.12 void SpeechProcLibrary::setLibCtx (void * *ctx*, speech_proc_algo_type_t *algo*) [inline]

Function called by proxy when lib is instanciated

Parameters

← ***ctx*** Library context pointer

← ***algo*** Library instance (Uplink or Downlink)

3 File Documentation

3.1 speech_proc_interface.h File Reference

Interface between NMF wrappers of OMX Speechproc component and the actual processing library.

Data Structures

- struct speech_proc_frame_info_t
- struct speech_proc_setting_t
- struct speech_proc_portsetting_t

Typedefs

- typedef struct speech_proc_frame_info_t speech_proc_frame_info_t
- typedef struct speech_proc_setting_t speech_proc_settings_t
- typedef struct speech_proc_portsetting_t speech_proc_port_settings_t

Enumerations

- enum speech_proc_algo_t { SP_UPLINK_NO_ALGO, SP_DOWNLINK_NO_ALGO, SP_UPLINK_ALGO, SP_DOWNLINK_ALGO }
- enum speech_proc_port_index_t { SP_MAIN_PORT, SP_REFERENCE_PORT }
- enum speech_proc_frame_info_element_t {
  SPEECH_PROC_INFO_ELEMENT_SYSTEM, SPEECH_PROC_INFO_ELEMENT_SPEECH_CODEC, SPEECH_PROC_INFO_ELEMENT_RXTYPE_OR_SID, SPEECH_PROC_INFO_ELEMENT_BITRATE_OR_TAF,
  SPEECH_PROC_INFO_ELEMENT_FQI_OR_BFI, SPEECH_PROC_INFO_ELEMENT_EXTRA, SPEECH_PROC_INFO_ELEMENT_NUMBER_OF }
- enum speech_proc_error_t {
  SP_ERROR_NONE, SP_ERROR_NO_MORE_MEMORY, SP_ERROR_UNSUPPORTED, SP_ERROR,
  SP_ERROR_SPECIFIC_START = 0x1000 }

Functions

- speech_proc_error_t speech_proc_init (speech_proc_algo_t algo, void **ctx, unsigned int *version)
- speech_proc_error_t speech_proc_close (void *ctx)
- speech_proc_error_t speech_proc_set_parameter (void *ctx, int index, void *config_struct)
- speech_proc_error_t speech_proc_set_config (void *ctx, int index, void *config_struct)
- speech_proc_error_t speech_proc_get_config (void *ctx, int index, void *config_struct)
- speech_proc_error_t speech_proc_set_mode (void *ctx, const speech_proc_settings_t *mode, const speech_proc_port_settings_t *inports, const speech_proc_port_settings_t *outports)
- speech_proc_error_t speech_proc_open (void *ctx)
- speech_proc_error_t speech_proc_process (void *ctx, short **input, int nb_input, short **output, int nb_output, const speech_proc_frame_info_t *frame_info)
- void * speech_proc_get_processing_info (void *ctx)
- speech_proc_error_t speech_proc_set_processing_info (void *ctx, void *info)
- speech_proc_error_t speech_proc_get_sidetone_gain (void *ctx, int *gain, int *updated)

3.1.1 Detailed Description

Interface between NMF wrappers of OMX Speechproc component and the actual processing library.

Author

ST-Ericsson

Version

v1.0.0

3.1.2 Typedef Documentation

3.1.2.1 typedef struct speech_proc_frame_info_t speech_proc_frame_info_t

Frame information

3.1.2.2 typedef struct speech_proc_portsetting_t speech_proc_port_settings_t

Define settings

3.1.2.3 typedef struct speech_proc_setting_t speech_proc_settings_t

Define buffer size, sample size and sample rate

3.1.3 Enumeration Type Documentation

3.1.3.1 enum speech_proc_algo_t

Used at init to tell which kind of algorithm the instantiation of the library will have to handle

Enumerator:

SP_UPLINK_NO_ALGO Bypass Mode uplink

SP_DOWNLINK_NO_ALGO Bypass Mode downlink

SP_UPLINK_ALGO Uplink Algorithms

SP_DOWNLINK_ALGO Downlink Algorithms

3.1.3.2 enum speech_proc_error_t

Error type

Enumerator:

SP_ERROR_NONE No error

SP_ERROR_NO_MORE_MEMORY Not enough memory available

SP_ERROR_UNSUPPORTED Feature not supported

SP_ERROR Any other Kind of error, not specified

SP_ERROR_SPECIFIC_START Library can define all necessary error code starting from this point

3.1.3.3 enum speech_proc_frame_info_element_t

Used to describe speech data for downlink.

	Bad Frame	Speech
AMR-WB 3G	FQI (0=Bad, 1=Good)	Rate 1-8(speech) 9,15(no speech), 14 see below
AMR-NB 3G	FQI (0=Bad, 1=Good)	Rate 1-7(speech) 8,15(no speech)
AMR-WB 2G	RX type 1,3,6 (Bad), 0,4,5,7 (Good), 2 see below	Rate 1-8(speech) 9,15(no speech)
AMR-NB 2G	RX type 1,3,6 (Bad), 0,4,5,7 (Good), 2 see below	Rate 1-7(speech) 8,15(no speech)
FR,HR,EFR 2G	BFI (0=Good, 1=Bad)	SID 0 (speech), 2 (no speech)

Speech Lost (2) for 2G AMR (NB and WB) we should idealy use previous frame. The same is true for AMR-WB speech lost in 3G

Previous frame	Bad Frame	Speech
Speech	true	true
No speech	false	false

Enumerator:

SPEECH_PROC_INFO_ELEMENT_SYSTEM Current system, 1 = 2G, 2 = 3G

SPEECH_PROC_INFO_ELEMENT_SPEECH_CODEC Which speech codec used to decode downlink frame. Half rate (1), Full Rate (2), Enhanced Full Rate (3), AMR-NB (4), AMR-WB (5)

SPEECH_PROC_INFO_ELEMENT_RXTYPE_OR_SID For AMR-NB the field contains RX type as found in downlink message. For Half Rate, Full Rate, and Enhanced Full Rate the field contains silence descriptor. Both field only valid for 2G

SPEECH_PROC_INFO_ELEMENT_BITRATE_OR_TAF For AMR-NB the field contains bit rate as defined in downlink message. For Half Rate, Full Rate, and Enhanced Full Rate the field contains TAF flag as defined in downlink message

SPEECH_PROC_INFO_ELEMENT_FQI_OR_BFI BFI is valid only for Half Rate, Full Rate, and Enhanced Full Rate and is based on BFI flag in downlink message. For Half Rate it is True if either BFI or UFI is true, otherwise False. For Full Rate it is True if either BFI or DFI is true, otherwise False.

SPEECH_PROC_INFO_ELEMENT_EXTRA Additional information that can be used for other purpose

SPEECH_PROC_INFO_ELEMENT_NUMBER_OF

3.1.3.4 enum speech_proc_port_index_t

Index used to distinguish various input buffers

Enumerator:

SP_MAIN_PORT input port for all kind of algorithms

SP_REFERENCE_PORT observation port, only for uplink algorithms

3.1.4 Function Documentation

3.1.4.1 speech_proc_error_t speech_proc_close (void * *ctx*)

Called once to close the library.

This function should free all memory used by the library.

Parameters

ctx pointer on local context

Returns

SP_ERROR_NONE if no error
SP_ERROR otherwise

3.1.4.2 speech_proc_error_t speech_proc_get_config (void * *ctx*, int *index*, void * *config_struct*)

This function is NOT called by NMF wrapper. Nevertheless, the OMX layer may need to retrieve information directly from the library (like internal algo variable values for example). If so, this function will be called from SpeechProcLibrary derived class, and then if you don't call it you don't need to implement it.

Parameters

ctx pointer on local context

index specifies which algo is to be retrieved

config_struct configuration structure for this index (to be casted into right structure)

Returns

SP_ERROR_NONE if no error
SP_ERROR_XXX otherwise

3.1.4.3 void* speech_proc_get_processing_info (void * *ctx*)

Called after each call to process.

This function should return usefull information for the other algorithms group. The buffer returned should have be allocated by the library itself.

Parameters

ctx pointer on local context

Returns

pointer on information area
NULL if no information is to be shared

3.1.4.4 speech_proc_error_t speech_proc_get_sidetone_gain (void * *ctx*, int * *gain*, int * *updated*)

Called after each call to process.

This function should return the gain to apply for sidetone. The value should be specify in mB (millibels = 1/100 dB). Although it is called after each frame, sidetone gain is only "used" to update sinks & sources gain if updated flag is set.

Parameters

ctx pointer on local context

gain pointer on gain.

updated pointer on flag to be set if sidetone gain value has been modified compared to previous call

Returns

SP_ERROR_NONE if the gain pointer has been updated
SP_ERROR_UNSUPPORTED if sidetone gain is not supported by the lib (it will prevent further calls to this function)

3.1.4.5 speech_proc_error_t speech_proc_init (speech_proc_algo_t *algo*, void ** *ctx*, unsigned int * *version*)

Called once to initialize the library.

This function should allocate and initialize all the necessary data structure, and then return a pointer on its context. This context will be passed as first argument in all other interface functions. In case of error *ctx **MUST** also be set to NULL.

Parameters

algo specify the kind of algorithms this instance will process

ctx pointer of pointer to return address of the context

version pointer to be filled with library version (if any)

Returns

SP_ERROR_NONE if no error
SP_ERROR_XXX otherwise

3.1.4.6 speech_proc_error_t speech_proc_open (void * *ctx*)

Called once before first call to speech_proc_process and after call to speech_proc_set_parameter function.

This function finalize memory allocation (if depending of static parameters), and library initialization.

Parameters

ctx pointer on local context

Returns

SP_ERROR_NONE if no error
SP_ERROR_XXX otherwise

3.1.4.7 speech_proc_error_t speech_proc_process (void * *ctx*, short ** *input*, int *nb_input*, short ** *output*, int *nb_output*, const speech_proc_frame_info_t * *frame_info*)

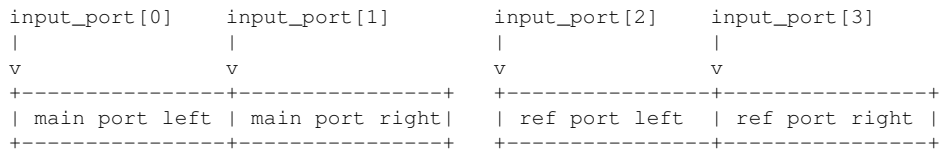
Called each time a new input buffer is available.

This function should always generate nb_output (always 1 currently) output buffers. Size, resolution, sample rate and channels layout of input/output buffers have already been set with speech_proc_set_mode function.

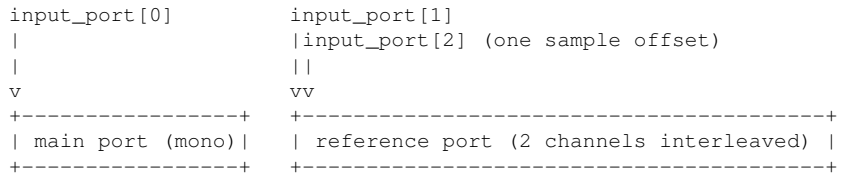
Additional frame info can also be given as parameter. For downlink it contains information received on downlink with speech codec specific information such as BFI. If no information exists then parameter is set to NULL. NOTICE : What we call input/output is one channel and not all channels of one port. For example, uplink has two input ports (SP_MAIN_PORT and SP_REFERENCE_PORT) but MAIN_PORT may be mono and reference port may be stereo. In this case nb_input will be 3.

Layout of input/output buffers will be organized so that all channels of one port will be in one continuous buffer.

Example for 2 input ports, both stereo and not interleaved:



Example for 2 input ports, one mono one stereo and interleaved:



Parameters

ctx pointer on local context

input table of input buffers

nb_input number of input channels

output table of output buffers

nb_output number of output channels

frame_info contains frame information. Can be NULL in which case no frame info exists.

Returns

SP_ERROR_NONE if no error
SP_ERROR_XXX otherwise

3.1.4.8 speech_proc_error_t speech_proc_set_config (void * *ctx*, int *index*, void * *config_struct*)

Called at any time before or during process.

This function is called to set dynamic parameters. Use OMX configuration prototype. Index filtering will be done in above layers and then library will only received indexes that it is supposed to support.

Parameters

ctx pointer on local context

index specifies which algo is to be configured

config_struct configuration structure for this index (to be casted into right structure)

Returns

SP_ERROR_NONE if no error
SP_ERROR_XXX otherwise

3.1.4.9 speech_proc_error_t speech_proc_set_mode (void * *ctx*, const speech_proc_settings_t * *mode*, const speech_proc_port_settings_t * *inports*, const speech_proc_port_settings_t * *outports*)

Called to set the working sample frequency.

Necessarily called once before first process but can be re-called during process. Called when both input and ouput port are enabled (but not necessarily reference port for uplink. It will be recalled if reference port is enabled after)

Parameters

ctx pointer on local context

mode global settings

inports settings of input ports

outports settings of output ports

Returns

SP_ERROR_NONE if no error
SP_ERROR_XXX otherwise (for example if settings asked are not supported)

3.1.4.10 speech_proc_error_t speech_proc_set_parameter (void * *ctx*, int *index*, void * *config_struct*)

Called before speech_proc_open.

This function is called to set static parameters. Use OMX configuration prototype. Index filtering will be done in above layers and then library will only received indexes that it is supposed to support.

Parameters

ctx pointer on local context

index specifies which algo is to be configured

config_struct configuration structure for this index (to be casted into right structure)

Returns

SP_ERROR_NONE if no error
SP_ERROR_XXX otherwise

3.1.4.11 speech_proc_error_t speech_proc_set_processing_info (void * *ctx*, void * *info*)

Called by the other algorithms group (i.e. downlink for uplink, and uplink for downlink) wrapper after processing one frame.

This function provides information about the other path

Parameters

ctx pointer on local context

info pointer on information from other path

Returns

SP_ERROR_NONE if no error
SP_ERROR_XXX otherwise

3.2 speech_proc_interface.txt File Reference

3.3 speech_proc_omx_interface.h File Reference

Interface between OMX configuration and the actual processing library.

`#include <OMX_Audio.h>`

Data Structures

- class SpeechProcLibrary

Enumerations

- enum speech_proc_port_name_t {
  INPUT_UPLINK_PORT, INPUT_DOWNLINK_PORT, INPUT_REFERENCE_PORT, OUTPUT_UPLINK_PORT,
  OUTPUT_DOWNLINK_PORT, SPEECH_PROC_NB_AUDIO_PORT, OUTPUT_CONTROL_PORT = SPEECH_PROC_NB_AUDIO_PORT, SPEECH_PROC_NB_PORT }
- enum speech_proc_algo_type_t { UPLINK_ALGORITHMS = 2, DOWNLINK_ALGORITHMS = 3 }

Functions

- SpeechProcLibrary * speech_proc_getOMXInterface (void)

3.3.1 Detailed Description

Interface between OMX configuration and the actual processing library.

Author

ST-Ericsson

3.3.2 Enumeration Type Documentation

3.3.2.1 enum speech_proc_algo_type_t

Enum to identify the type of algo at OMX level (inline with speech_proc_algo_t)

Enumerator:

UPLINK_ALGORITHMS

DOWNLINK_ALGORITHMS

3.3.2.2 enum speech_proc_port_name_t

Enum to identify port at OMX level

Enumerator:

INPUT_UPLINK_PORT
INPUT_DOWNLINK_PORT
INPUT_REFERENCE_PORT
OUTPUT_UPLINK_PORT
OUTPUT_DOWNLINK_PORT
SPEECH_PROC_NB_AUDIO_PORT
OUTPUT_CONTROL_PORT
SPEECH_PROC_NB_PORT

3.3.3 Function Documentation

3.3.3.1 SpeechProcLibrary* speech_proc_getOMXInterface (void)

Factory method so that Speech Proc Proxy is completely independant of the processing library used.

Returns

pointer on a class implementing SpeechProcLibrary interface

Index

~SpeechProcLibrary
SpeechProcLibrary, 11

checkConfig
SpeechProcLibrary, 11
checkParameter
SpeechProcLibrary, 12
checkPortSettings
SpeechProcLibrary, 12

DOWNLINK_ALGORITHMS
speech_proc_omx_interface.h, 24

enabled
speech_proc_portsetting_t, 9

FrameInfo
speech_proc_frame_info_t, 9
framesize
speech_proc_setting_t, 10

getConfig
SpeechProcLibrary, 13
getExtensionIndex
SpeechProcLibrary, 13
getLibCtx
SpeechProcLibrary, 13
getNbParamAndConfig
SpeechProcLibrary, 14
getNextConfig
SpeechProcLibrary, 14
getNextParameter
SpeechProcLibrary, 15
getOMXParamOrConfig
SpeechProcLibrary, 15

INPUT_DOWNLINK_PORT
speech_proc_omx_interface.h, 25
INPUT_REFERENCE_PORT
speech_proc_omx_interface.h, 25
INPUT_UPLINK_PORT
speech_proc_omx_interface.h, 25
interleaved
speech_proc_portsetting_t, 9
isBypass
SpeechProcLibrary, 16

nb_channels
speech_proc_portsetting_t, 10
nb_ports
speech_proc_portsetting_t, 10

OUTPUT_CONTROL_PORT
speech_proc_omx_interface.h, 25
OUTPUT_DOWNLINK_PORT
speech_proc_omx_interface.h, 25
OUTPUT_UPLINK_PORT
speech_proc_omx_interface.h, 25

port
speech_proc_portsetting_t, 10

resolution
speech_proc_setting_t, 10

samplerate
speech_proc_portsetting_t, 10
setLibCtx
SpeechProcLibrary, 16
SP_DOWNLINK_ALGO
speech_proc_interface.h, 18
SP_DOWNLINK_NO_ALGO
speech_proc_interface.h, 18
SP_ERROR
speech_proc_interface.h, 18
SP_ERROR_NO_MORE_MEMORY
speech_proc_interface.h, 18
SP_ERROR_NONE
speech_proc_interface.h, 18
SP_ERROR_SPECIFIC_START
speech_proc_interface.h, 18
SP_ERROR_UNSUPPORTED
speech_proc_interface.h, 18
SP_MAIN_PORT
speech_proc_interface.h, 19
SP_REFERENCE_PORT
speech_proc_interface.h, 19
SP_UPLINK_ALGO
speech_proc_interface.h, 18
SP_UPLINK_NO_ALGO
speech_proc_interface.h, 18
SPEECH_PROC_INFO_ELEMENT_BITRATE_-
OR_TAF
speech_proc_interface.h, 19
SPEECH_PROC_INFO_ELEMENT_EXTRA
speech_proc_interface.h, 19
SPEECH_PROC_INFO_ELEMENT_FQI_OR_-
BFI
speech_proc_interface.h, 19
SPEECH_PROC_INFO_ELEMENT_NUMBER_-
OF
speech_proc_interface.h, 19

SPEECH_PROC_INFO_ELEMENT_RXTYPE_-
OR_SID
speech_proc_interface.h, 19
SPEECH_PROC_INFO_ELEMENT_SPEECH_-
CODEC
speech_proc_interface.h, 19
SPEECH_PROC_INFO_ELEMENT_SYSTEM
speech_proc_interface.h, 19
speech_proc_interface.h
SP_DOWNLINK_ALGO, 18
SP_DOWNLINK_NO_ALGO, 18
SP_ERROR, 18
SP_ERROR_NO_MORE_MEMORY, 18
SP_ERROR_NONE, 18
SP_ERROR_SPECIFIC_START, 18
SP_ERROR_UNSUPPORTED, 18
SP_MAIN_PORT, 19
SP_REFERENCE_PORT, 19
SP_UPLINK_ALGO, 18
SP_UPLINK_NO_ALGO, 18
SPEECH_PROC_INFO_ELEMENT_-
BITRATE_OR_TAF, 19
SPEECH_PROC_INFO_ELEMENT_-
EXTRA, 19
SPEECH_PROC_INFO_ELEMENT_FQI_-
OR_BFI, 19
SPEECH_PROC_INFO_ELEMENT_-
NUMBER_OF, 19
SPEECH_PROC_INFO_ELEMENT_-
RXTYPE_OR_SID, 19
SPEECH_PROC_INFO_ELEMENT_-
SPEECH_CODEC, 19
SPEECH_PROC_INFO_ELEMENT_-
SYSTEM, 19
SPEECH_PROC_NB_AUDIO_PORT
speech_proc_omx_interface.h, 25
SPEECH_PROC_NB_PORT
speech_proc_omx_interface.h, 25
speech_proc_omx_interface.h
DOWNLINK_ALGORITHMS, 24
INPUT_DOWNLINK_PORT, 25
INPUT_REFERENCE_PORT, 25
INPUT_UPLINK_PORT, 25
OUTPUT_CONTROL_PORT, 25
OUTPUT_DOWNLINK_PORT, 25
OUTPUT_UPLINK_PORT, 25
SPEECH_PROC_NB_AUDIO_PORT, 25
SPEECH_PROC_NB_PORT, 25
UPLINK_ALGORITHMS, 24
speech_proc_algo_t
speech_proc_interface.h, 18
speech_proc_algo_type_t
speech_proc_omx_interface.h, 24
speech_proc_close
speech_proc_interface.h, 19
speech_proc_error_t
speech_proc_interface.h, 18
speech_proc_frame_info_element_t
speech_proc_interface.h, 18
speech_proc_frame_info_t, 9
FrameInfo, 9
speech_proc_interface.h, 17
speech_proc_get_config
speech_proc_interface.h, 19
speech_proc_get_processing_info
speech_proc_interface.h, 20
speech_proc_get_sidetone_gain
speech_proc_interface.h, 20
speech_proc_getOMXInterface
speech_proc_omx_interface.h, 25
speech_proc_init
speech_proc_interface.h, 20
speech_proc_interface.h, 16
speech_proc_algo_t, 18
speech_proc_close, 19
speech_proc_error_t, 18
speech_proc_frame_info_element_t, 18
speech_proc_frame_info_t, 17
speech_proc_get_config, 19
speech_proc_get_processing_info, 20
speech_proc_get_sidetone_gain, 20
speech_proc_init, 20
speech_proc_open, 21
speech_proc_port_index_t, 19
speech_proc_port_settings_t, 17
speech_proc_process, 21
speech_proc_set_config, 22
speech_proc_set_mode, 22
speech_proc_set_parameter, 23
speech_proc_set_processing_info, 23
speech_proc_settings_t, 18
speech_proc_interface.txt, 24
speech_proc_omx_interface.h, 24
speech_proc_algo_type_t, 24
speech_proc_getOMXInterface, 25
speech_proc_port_name_t, 24
speech_proc_open
speech_proc_interface.h, 21
speech_proc_port_index_t
speech_proc_interface.h, 19
speech_proc_port_name_t
speech_proc_omx_interface.h, 24
speech_proc_port_settings_t
speech_proc_interface.h, 17
speech_proc_portsetting_t, 9
enabled, 9
interleaved, 9
nb_channels, 10

nb_ports, 10
port, 10
samplerate, 10
speech_proc_process
speech_proc_interface.h, 21
speech_proc_set_config
speech_proc_interface.h, 22
speech_proc_set_mode
speech_proc_interface.h, 22
speech_proc_set_parameter
speech_proc_interface.h, 23
speech_proc_set_processing_info
speech_proc_interface.h, 23
speech_proc_setting_t, 10
framesize, 10
resolution, 10
speech_proc_settings_t
speech_proc_interface.h, 18
SpeechProcLibrary, 10
∼SpeechProcLibrary, 11
checkConfig, 11
checkParameter, 12
checkPortSettings, 12
getConfig, 13
getExtensionIndex, 13
getLibCtx, 13
getNbParamAndConfig, 14
getNextConfig, 14
getNextParameter, 15
getOMXParamOrConfig, 15
isBypass, 16
setLibCtx, 16
SpeechProcLibrary, 11

UPLINK_ALGORITHMS
speech_proc_omx_interface.h, 24