Speech Processing Implementation

v2.7.3

Generated by Doxygen 1.6.3

Thu Jun 28 19:33:29 2012

Contents

1 Table of contents

This documentation aims to briefly describe implementation of OMX speech Proc component (OMX.ST.AFM.speech_proc), both hybrid and full host version.

- Overview
- UL & DL wrappers description
- Time alignment
- Configuration
- How to interpret speech_proc traces
- Differnces between hybrid and full host version

2 Overview

Introduction

This OMX component (OMX.ST.AFM.speech_proc) contains 5 audio ports and is made to be used in an OMX graph that look like :

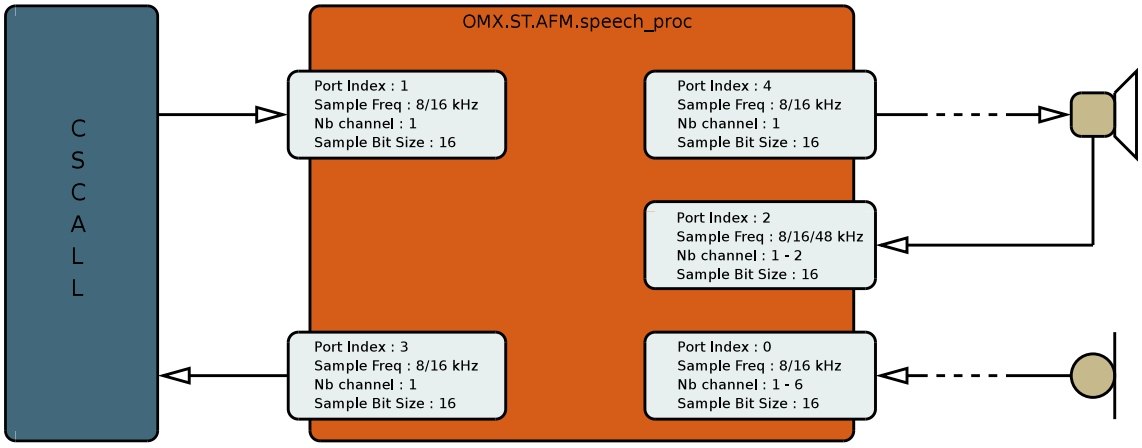

Figure 1: Speech Proc OMX component

This component is made of 2 main parts :

- a NMF network : This is the implementation of the processing part.
- an OMX proxy : This part is responsible to provide OMX API, and forwards (resp. receives) command/data to(resp. from) NMF network

It exists two NMF network implementation:

- Hybrid version : contains both MPC and HOST nmf component (This is the main version)
- Host version : contians only HOST nmf version (developed for product whitout mmdsp, is it still needed ?)

Both version have the same "design" (i.e. same main NMF components). The differences between those two versions are highligted Differences Explanation here.

To understand the following schemes please refer to the following caption.

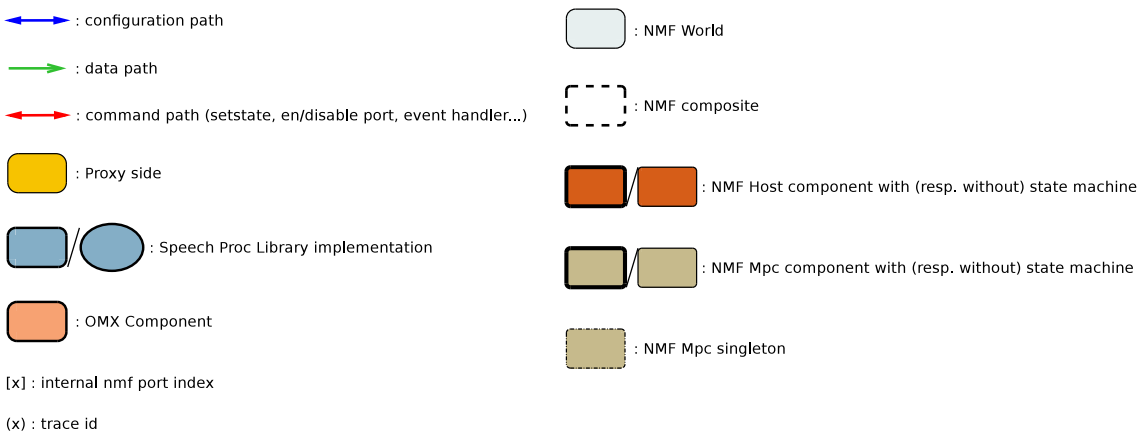

Figure 2: caption for detailled graphs

Hybrid Version

The hybrid version is described in the following figure :

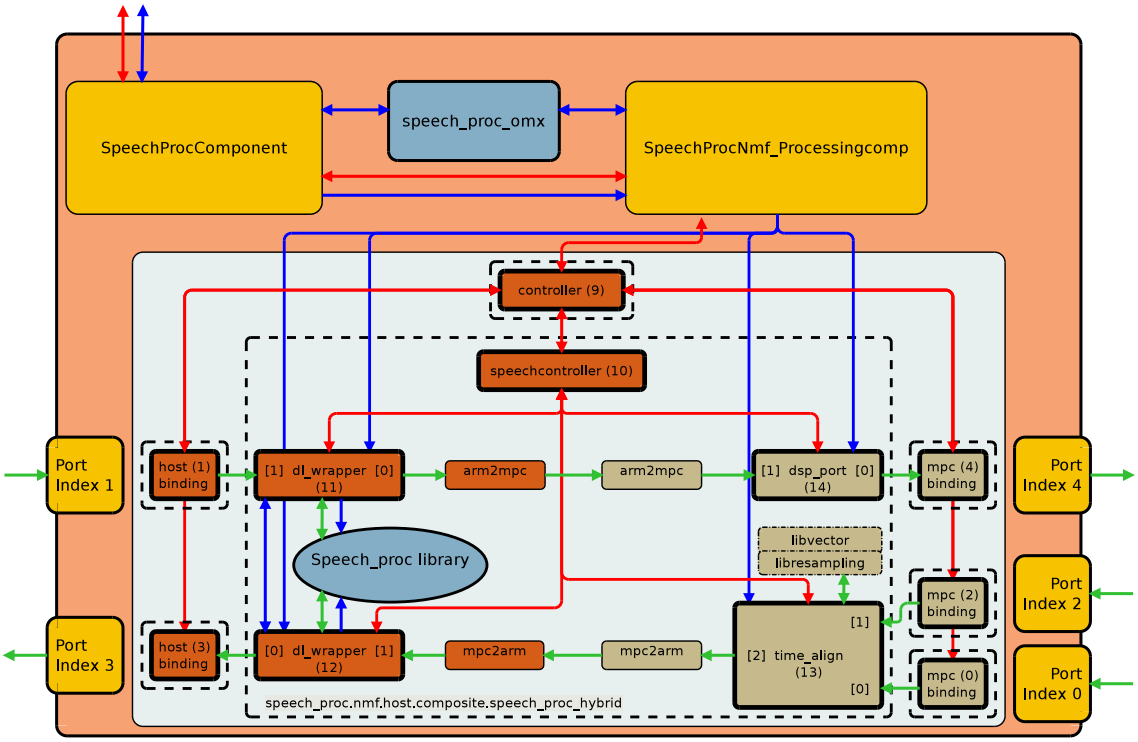

Figure 3: Speech Proc hybrid, detailed graph

Full Host version

The full host version is described in the following figure :

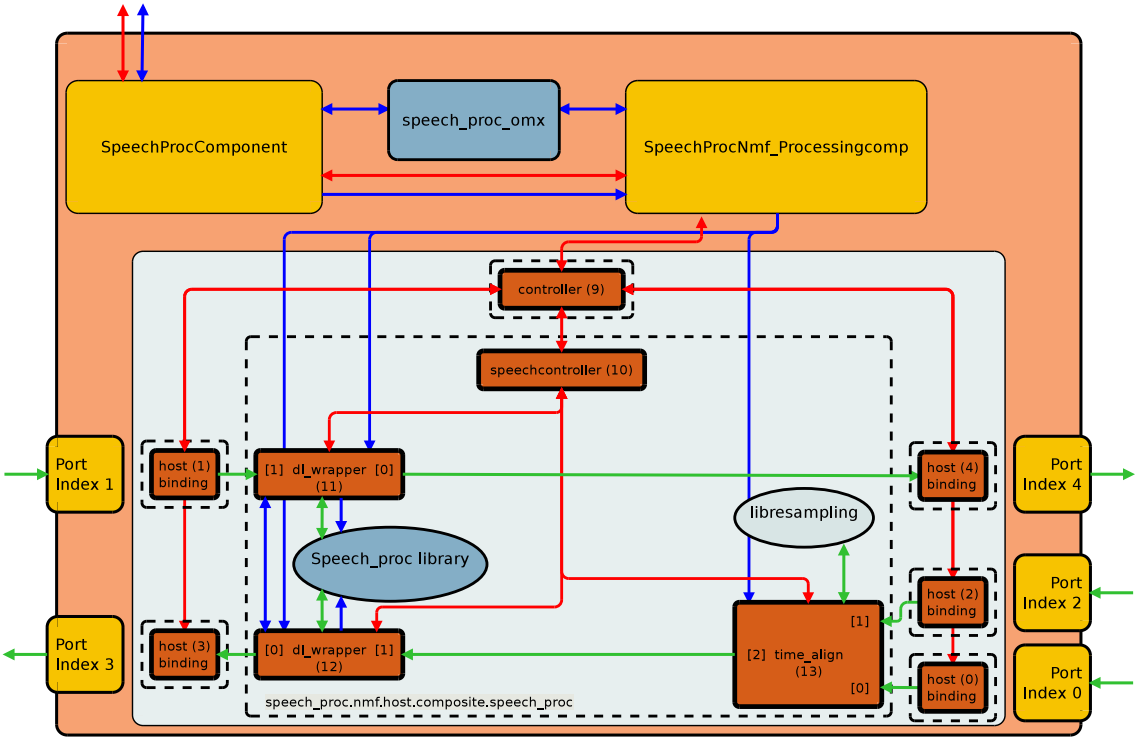

Figure 4: Speech Proc host, detailed graph

3 UL & DL wrappers

Those 2 components are only here to handle the state machine and the buffer transfer. They do nothing more than calling the processing library with the correct buffers. All the processing is done by the processing library, and the interface between wrappers and the library is fully detailled in speech_proc_itf component.

4 Time Alignment

The goal of this component is to temporarily align the data coming from the source (microphone) with the reference ones received from the sink (speaker). This alignment is done in order to optimize echo cancellation algorithm, by keeping a constant delay between uplink and reference data.

It will also adapt the sample frequency of reference data to the sample frequency of uplink processing.

Principle

The alignment is done based on timestamp set inside buffers. Those timestamps are set by the sink (resp. source) component and represent the date (in us) when the first sample of the buffer has been send to (resp. received from) MSP transfer.

When a buffer is received on the reference port (port index 2), time alignmennt will adapt its sample frequency (using libresampling) and store it in its ring buffer. Then when a buffer is received on the uplink port (port index 0), time alignment will check its reference ring buffer to extract the samples with the same timestamp, packetize both signal into a single buffer and send it to ul_wrapper.

Remarks

If timestamps are not set on buffers received by time alignment (whether there are not updated by source/sink components or not propagated by component between source and speech_proc), time alignment will no longer align the data. In this case, for each uplink buffer it will simply extract the oldest samples from its reference ring buffer.

Timestamp

It is important to understand the meaning of the timestamp included buffers.

For reference port buffer, the timestamp (t0) indicates the date (in us) at which sink started to transfer this buffer via MSP. The sink will send this buffer to reference port before the beginning of transfer, so at worst at date=t0.

For uplink port buffer, the timestamp (t0) indicates the date (in us) at which source started to captured this buffer via MSP. The time_align can only received this buffer when it is completly acquired, so at best at date=t0+5ms (if we assume transfer size is 5ms).

Thus for a same timestamp (t0), buffer will always be received on reference port before uplink port.

Even if timestamp are expressed in us, since MSP transfer unit is 1ms the "real" (understand usefull) unit for timestamp is ms. That's why in the code, and in the graphs after, we mainly work with ms rather than us.

Special cases

The following graphs are designed like this :

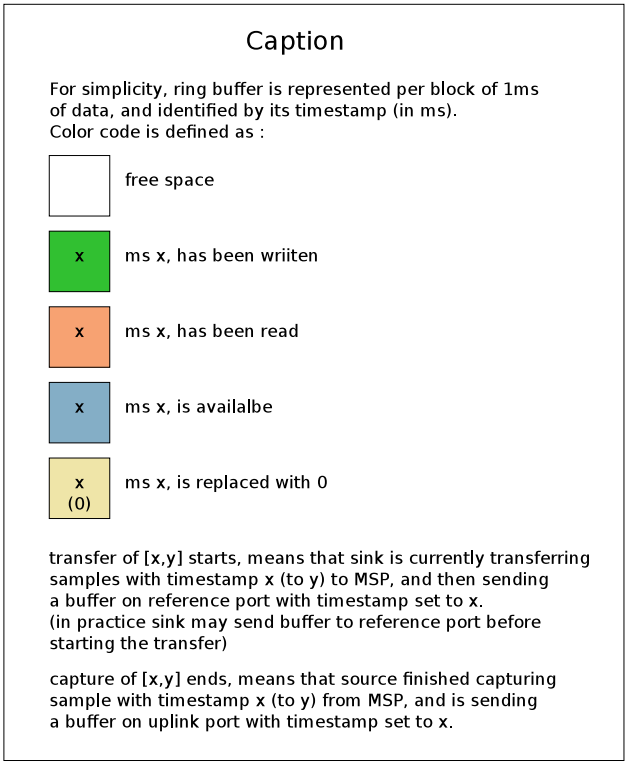

Figure 5: definiton for time alignment graphs

Start

At start, the ring buffer may not contains the reference samples for the first uplink buffers. In this case, missing samples are replaced by 0.

Let's take a more concrete example :

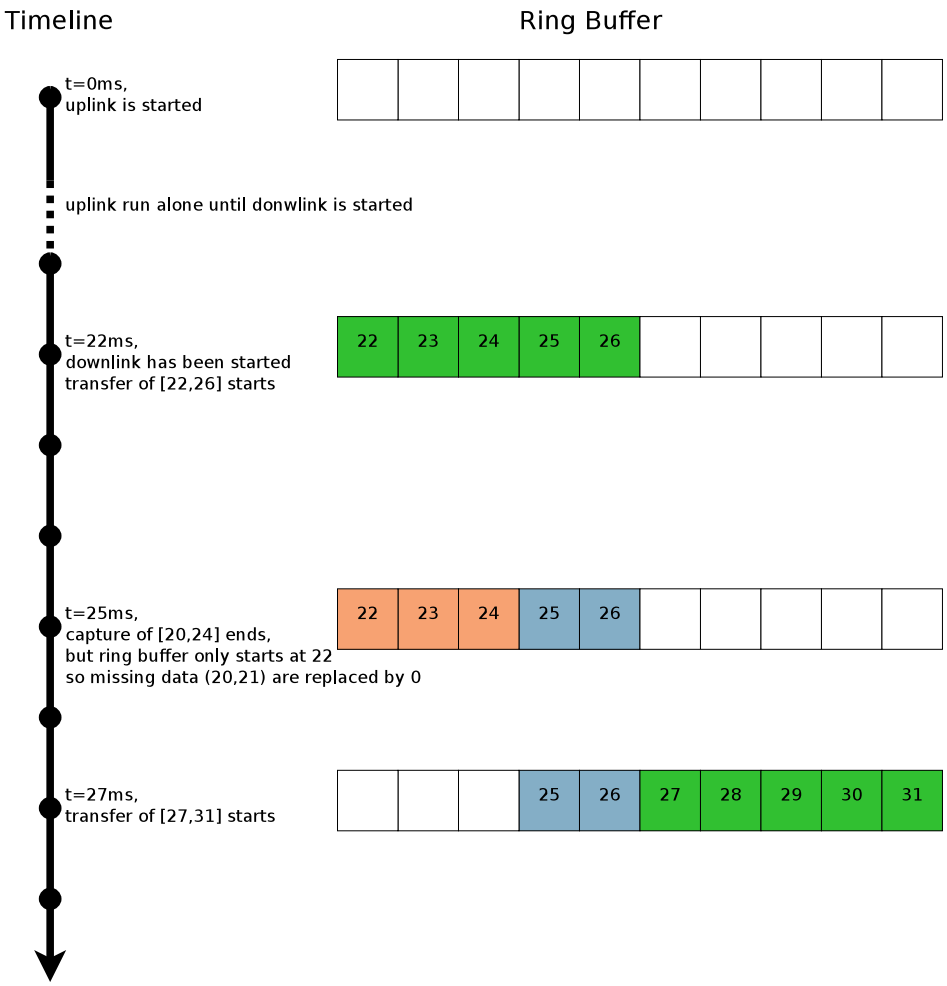

Figure 6: example of call start

Underrun on sink

If underun happens on sink, missing samples are replaced with 0.

We can distinguish 2 cases :

- Underrun is small, in this case when time align see the discontinous timestamp on reference port, it replaces missing data with 0 inside its ring buffer.

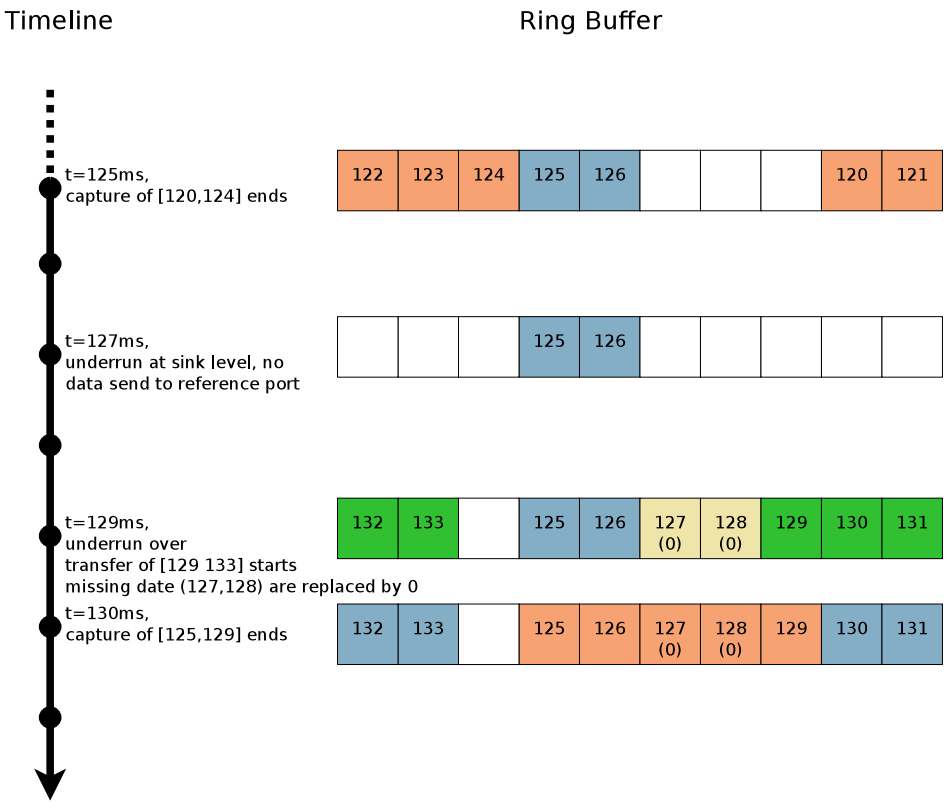

Figure 7: example of "small" underrun

- Underrun is big enough, in this case when the uplink buffer is received we don't have the corresponding samples in the ring buffer. Time align will replace missing data with 0

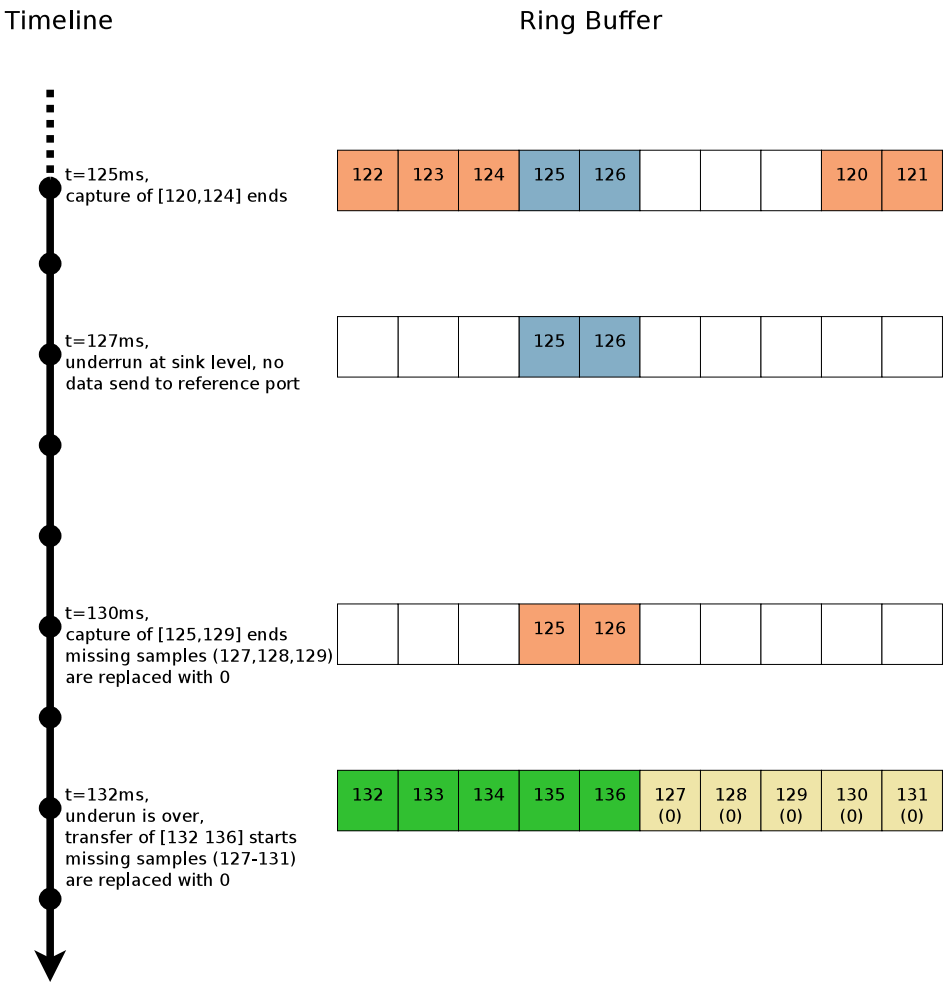

Figure 8: example of "big" underrun

Some implementation details

- We saw that the ring buffer contains continous data. Indeed if discontinuity is detected in reference timestamp, missing samples are replaced with 0 inside the ring buffer. Thus knowning the timestamp of the last sample, stored in mTimeStamp, and the size used in the ring buffer, stored in mRingUsedSize is enough to calculate the date of every samples in the ring buffer.

- mRealTimeReference is only test to false for test in which reference port is connected to a file.

- input out output buffer don't have the same size. Output buffer size is PROCESSING_BUFFER_DURATION, and input size is UL_INPUT_BUFFER_DURATION.

  The funtion packOutputBuffer, is in charge to concatenate NB_OUTPUT_PART (=PROCESSING_BUFFER_DURATION/UL_INPUT_BUFFER_DURATION) input buffers in 1 output buffer. The index of current subpart is stored in mOutputPart.

- output buffer contains input and reference data, in a multichannel packed (as opposed to interleaved) buffer.

5 Speech Proc traces

Time Align traces

DEBUG traces

to know if timestamp are present (or not) in buffers, and then if time align feature is activated :

- time align activated :

```
"Alignment starts with reference timestamp %d %d, buffer timestamp %d %d"
"Activate Alignment after some buffer with no alignment, with reference tim
estamp %d %d, buffer timestamp %d %d"
```

- time align is not activated :

```
"No time alignment because : reference status %d, buffer timestamp %d %d -
flags %d"
```

WARNING traces

- Overflow in ring buffer, may happen with or without time align feature

```
"Time align : Overflow in reference buffer "
```

- Discontinuity detected in timestamp on reference port (cf underrun)

```
"Time align : %d samples lost in reference data flow (buf %d, ring %d) . rep
 laced with 0"
```

- Missing data on the reference port, it is replaced with 0. This is the "big" underrun use case (underrun)

```
"Time align : ask data at %d us, whereas we already have samples up to %d us
  (- %d us)"
"Time align : add %d post padding samples"
```

- Data already overwritten (or never received) in the ring buffer. This may happen at the start (cf start Start use case) or if the ring buffer is too small.

```
"Time align : ask data at %d us, whereas we already have samples up to %d us
  (- %d us)"
"Time align : add %d pre padding samples"
```

UL & DL Wrappers traces

FLOW traces

Calls to processing library can be tracked with:

```
"start dl process"
"end dl process"
```

and

```
"start ul process"
"end ul process"
```

DATA traces

It is possible to dump to send/received to/from processing library by activating the correct trace level.

You can then use the SpeechProcActivity script (in multimedia/audio/dbg_tools) to extract data from your trace file.

- Activate trace level TRACE_SP_PORT_UL_IN (aka USER1) to dump uplink input port data
- Activate trace level TRACE_SP_PORT_DL_IN (aka USER2) to dump downlink input port data
- Activate trace level TRACE_SP_PORT_UL_REF (aka USER3) to dump uplink reference port data
- Activate trace level TRACE_SP_PORT_UL_OUT (aka USER4) to dump uplink output port data
- Activate trace level TRACE_SP_PORT_DL_OUT (aka USER5) to dump downlink output port data

Remarks

- even if trace is done on the ARM, data traces can only be captured via fido box.
- If hybrid buffers are located in TCM (cf DL_HYBRID_BUFFER_MEMORY, UL_HYBRID_-BUFFER_MEMORY), data will NOT be trace even if the corresponding trace level is activated (TRACE_SP_PORT_DL_OUT for downlink) and (TRACE_SP_PORT_UL_IN and/or TRACE_SP_PORT_UL_REF for uplink). Indeed trace routine make 8bits access that isn't supported on mmdsp TCM memory

6 Differences Explanation

Here is a brief description of the differences between the two previous description

Proxy Side:

- SpeechProcComponent : The only difference is that for hybrid version, port 0,2,4 are MPC based and then sample bitsize is 24.
- SpeechProcNmf_ProcessingComp : 3 mains differences :
  - For hybrid version we inherit from AFMNmfHostMpc_ProcessingComp class and for full host version we inherit from AFMNmfHost_ProcessingComp. Hopefully the methods to implement are very similar in both case (just one more in AFMNmfHostMpc_ProcessingComp)
  - Nmf network is not the same, then some bindings and interfaces (for configuration) are not exactly the same.
    - Most of the memory used by DSP component is allocated at proxy level (in order to be able to change the memory type without recompiling dsp part).

NMF side:

- For hybrid version, we need arm2mpc/mpc2arm components for address translation between ARM and DSP. They play no role at all on the processing chain.
- dsp_port component for hybrid version, is here to ease the proprietary communication with other NMF MPC components on port 4. It plays no role in the processing chain.

  hopefully it will be removed in next version.

- dl_wrapper and ul_wrapper are **exactly** the same for both version.
- speechcontroller is almost the same, expect that in the hybrid version it has one more component (dsp_port) to control.
- time_align has to be implemented twice. Once as a MPC component and once as a HOST component. They have the same behaviour and the same NMF interfaces.

speechProc library

- Any library implementing the speech_proc_interface API, can be used indifferently in hybrid or full host version.

7 Speech Proc Configuration

Description of speech_proc variables that can be modified to adapt speech_proc behaviour.

All the macros described in this page are defined in speech_proc_config.h

Unless specified, modification of this varaibles doesn't imply recompilation of dsp code.

Processing buffer size

The size of the buffer processed by wrappers are defined by :

PROCESSING_BUFFER_DURATION, that is the size in ms of buffer received/send by ul/dl wrappers

PROCESSING_COUNT, that is the number of call to processing library for one buffer

==> it means that processing lib will process (PROCESSING_BUFFER_DURATION/PROCESSING_COUNT) ms buffer For now (and will probably remain like this) it MUST be 10

so the 2 options are 20/2 or 10/1

Note

modification on these values **DOES** imply DSP recompiltaion (for hybrid version)

Nmf buffer size on Downlink output port

The size of buffer delivered by nmf component (dl_wrapper) is defined, in ms, by DL_OUTPUT_BUFFER_DURATION. It must be a divider of PROCESSING_BUFFER_DURATION and for now it MUST be equal to PROCESSING_BUFFER_DURATION

Nmf buffer size on reference port

The size of nmf buffer, exchanged between reference port and sink's feedback port is defined, in ms, by REF_INPUT_BUFFER_DURATION

Note

modification on these values **DOES** imply DSP recompiltaion (for hybrid version)

Nmf buffer size on uplink port

The size of nmf buffer, exchanged between uplink port and source, is defined, in ms, by UL_INPUT_-BUFFER_DURATION

Note

modification on these values **DOES** imply DSP recompiltaion (for hybrid version)

Buffer Size at OMX level

The size of buffer required at OMX level (i.e. when a port is not tunneled, or tunneled to a non AFM component using OMX standard communication) is defined for each port, in ms, by :

- OMX_DL_INPUT_DURATION
- OMX_DL_OUTPUT_DURATION
- OMX_UL_INPUT_DURATION
- OMX_UL_REFERENCE_DURATION
- OMX_UL_OUTPUT_DURATION

Reference and Uplink port channel infomation

Reference port

The maximum number of channel supported by OMX reference port is defined by MAX_NB_OMX_-REF_CHANNEL. If the port is configured with a higher number of channel, speech_proc component will return an error.

Now in some cases, if the speech library does not support (or take advantage) of stereo input we can force to have a mono signal at nmf level (independently of OMX configuration) by setting the macro FORCE_MONO_REFERENCE to 1. In this case, the binding component (pcmadapter or shmpcmin), will downsample the signal before sending it to reference port.

This allow to save mips by running samplerate conversion on only one channel.

Uplink port

The maximum number of channel supported by OMX uplink input port is defined by MAX_NB_INPUT_-CHANNEL. If the port is configured with a higher number of channel, speech_proc component will return an error.

Adjusting this value to the one needed may save some memory.

Samplerate converter

The size of the memory allocated for samplerate converter is defined, in mmdsp words, by SRC_HEAP_-SIZE.

It **MUST** be inline with samplerate used.

Ring buffer on reference port

The size allocated for reference's ring buffer is defined by its duration in ms RING_BUFFER_DURATION, and the maximum sample freq of uplink signal in samples/ms MAX_SAMPLE_FREQ.

MMDSP Memory used

All mmdsp memory zones are allocated dynamically by proxy, removing the need to recompile DSP code to change buffer location.

The memory used for downlink shared (between arm and dsp) buffer is defined by DL_HYBRID_BUFFER_MEMORY, among sp_memory_type_t. The size allocated for this buffer is (DL_OUTPUT_BUFFER_DURATION ∗ MAX_SAMPLE_FREQ) samples plus one buffer_t structure.

The memory used for uplink shared (between arm and dsp) buffer is defined by UL_HYBRID_BUFFER_MEMORY, inside sp_memory_type_t. The size allocated for this buffer is (PROCESSING_BUFFER_DURATION ∗ MAX_SAMPLE_FREQ ∗ (MAX_NB_REF_CHANNEL + MAX_NB_INPUT_CHANNEL)) samples plus one buffer_t structure.

The memory used for samplerate converter is defined by SRC_MEMORY, among (CM_MM_MPC_TCM24 | CM_MM_MPC_ESRAM24 | CM_MM_MPC_SDRAM24). The size allocated is defined by SRC_HEAP_SIZE.

The memory used for reference's ring buffer is defined by RING_BUFFER_MEMORY, among (CM_MM_MPC_TCM24 | CM_MM_MPC_ESRAM24 | CM_MM_MPC_SDRAM24). The size allocated is defined by RING_BUFFER_SIZE.

8 File Documentation

8.1 speech_proc.txt File Reference

8.2 speech_proc_config.h File Reference

Contains various buffer size used in speech_proc component.

Defines

- #define SPEECH_PROC_MAJOR 2
- #define SPEECH_PROC_MINOR 7
- #define SPEECH_PROC_REVISION 3
- #define PROCESSING_BUFFER_DURATION 20
- #define PROCESSING_COUNT 2
- #define DL_OUTPUT_BUFFER_DURATION PROCESSING_BUFFER_DURATION
- #define REF_INPUT_BUFFER_DURATION 5
- #define UL_INPUT_BUFFER_DURATION 5
- #define OMX_DL_INPUT_DURATION 20
- #define OMX_DL_OUTPUT_DURATION 20
- #define OMX_UL_INPUT_DURATION 20
- #define OMX_UL_REFERENCE_DURATION 20
- #define OMX_UL_OUTPUT_DURATION 20
- #define MAX_NB_OMX_REF_CHANNEL 2

- #define FORCE_MONO_REFERENCE 1
- #define MAX_NB_REF_CHANNEL 1
- #define MAX_NB_INPUT_CHANNEL 2
- #define SRC_HEAP_SIZE 135
- #define MAX_SAMPLE_FREQ 16
- #define RING_BUFFER_DURATION 20
- #define RING_BUFFER_SIZE (MAX_SAMPLE_FREQ ∗ MAX_NB_REF_CHANNEL ∗ RING_-BUFFER_DURATION)
- #define DL_HYBRID_BUFFER_MEMORY SP_MEM_TCM
- #define UL_HYBRID_BUFFER_MEMORY SP_MEM_TCM
- #define SRC_MEMORY CM_MM_MPC_TCM24
- #define RING_BUFFER_MEMORY CM_MM_MPC_TCM24

Enumerations

- enum sp_memory_type_t { SP_MEM_TCM, SP_MEM_ESRAM, SP_MEM_DDR }

8.2.1 Detailed Description

Contains various buffer size used in speech_proc component.

Author

ST-Ericsson

8.2.2 Define Documentation

8.2.2.1 #define DL_HYBRID_BUFFER_MEMORY SP_MEM_TCM

memory to use for downlink hybrid buffer

8.2.2.2 #define DL_OUTPUT_BUFFER_DURATION PROCESSING_BUFFER_DURATION

NMF buffer size of dl_wrapper output port, in ms

8.2.2.3 #define FORCE_MONO_REFERENCE 1

force mono data on nmf reference port

8.2.2.4 #define MAX_NB_INPUT_CHANNEL 2

max number of channel on uplink inp[ut port

8.2.2.5 #define MAX_NB_OMX_REF_CHANNEL 2

max number of channel on OMX ref port

8.2.2.6 #define MAX_NB_REF_CHANNEL 1

8.2.2.7 #define MAX_SAMPLE_FREQ 16

max sample freq, in sample/ms

8.2.2.8 #define OMX_DL_INPUT_DURATION 20

OMX buffer size of DL input port, in ms

8.2.2.9 #define OMX_DL_OUTPUT_DURATION 20

OMX buffer size of DL output port, in ms

8.2.2.10 #define OMX_UL_INPUT_DURATION 20

OMX buffer size of UL input port, in ms

8.2.2.11 #define OMX_UL_OUTPUT_DURATION 20

OMX buffer size of UL outputput port, in ms

8.2.2.12 #define OMX_UL_REFERENCE_DURATION 20

OMX buffer size of UL reference port, in ms

8.2.2.13 #define PROCESSING_BUFFER_DURATION 20

NMF buffer size of ul_wrapper output/input ports and dl_wrapper input port, in ms

8.2.2.14 #define PROCESSING_COUNT 2

number of call to processing lib per buffer

8.2.2.15 #define REF_INPUT_BUFFER_DURATION 5

NMF buffer size of time_align reference port, in ms

8.2.2.16 #define RING_BUFFER_DURATION 20

ring buffer duration, in ms

8.2.2.17 #define RING_BUFFER_MEMORY CM_MM_MPC_TCM24

memory to use for ring buffer

8.2.2.18 #define RING_BUFFER_SIZE (MAX_SAMPLE_FREQ ∗ MAX_NB_REF_CHANNEL ∗ RING_BUFFER_DURATION)

8.2.2.19 #define SPEECH_PROC_MAJOR 2

8.2.2.20 #define SPEECH_PROC_MINOR 7

8.2.2.21 #define SPEECH_PROC_REVISION 3

8.2.2.22 #define SRC_HEAP_SIZE 135

number of words allocated for SRC

8.2.2.23 #define SRC_MEMORY CM_MM_MPC_TCM24

memory to use for SRC

8.2.2.24 #define UL_HYBRID_BUFFER_MEMORY SP_MEM_TCM

memory to use for uplink hybrid buffer

8.2.2.25 #define UL_INPUT_BUFFER_DURATION 5

NMF buffer size of time_align input port, in ms

8.2.3 Enumeration Type Documentation

8.2.3.1 enum sp_memory_type_t

Enumerator:

SP_MEM_TCM
SP_MEM_ESRAM
SP_MEM_DDR

Index