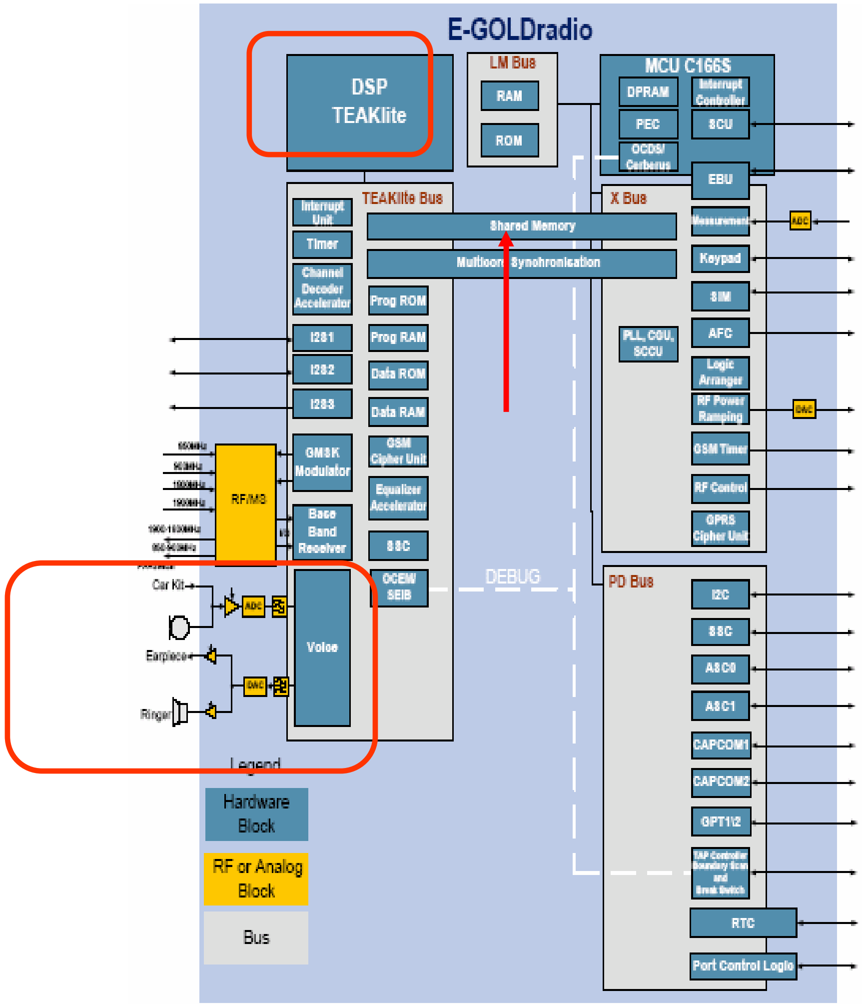
BP30 Audio drivers training: agenda

- Audio DSP Commands and parameters overview
- Audio Driver overview
- Interface description

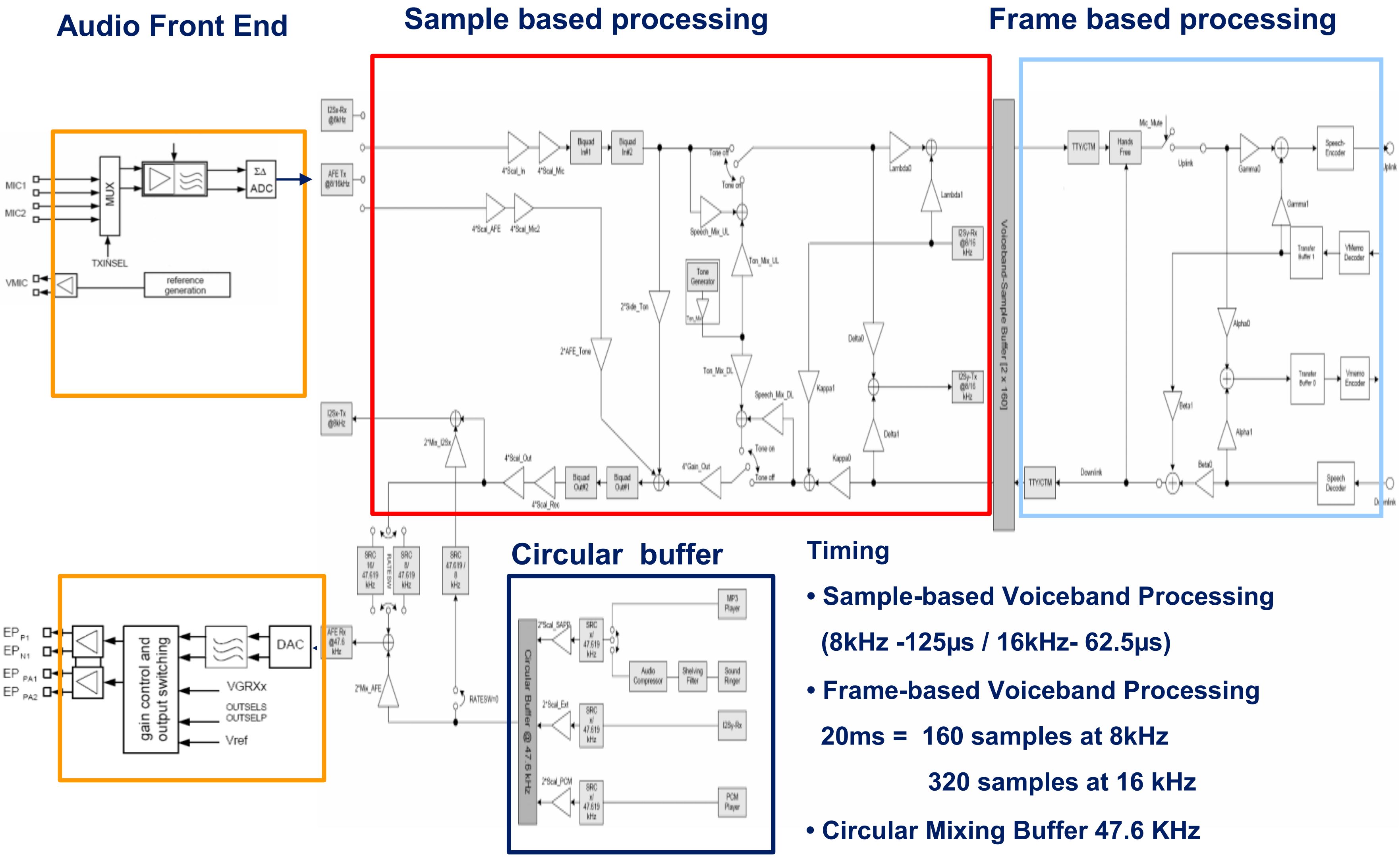
Andrea Guerra
L1& Drivers SW Engineer

NEONSEVEN S.r.l.
Via Stazione di Prosecco 15
34010 Sgonico (TS)
Italy
tel: + 39 040 25 29 400
fax: + 39 040 25 29 394
Mob: + 39 329 028 66 29
email: andrea.guerra@neonseven.com
www: www.neonseven.com

Audio control in E-GOLDradio



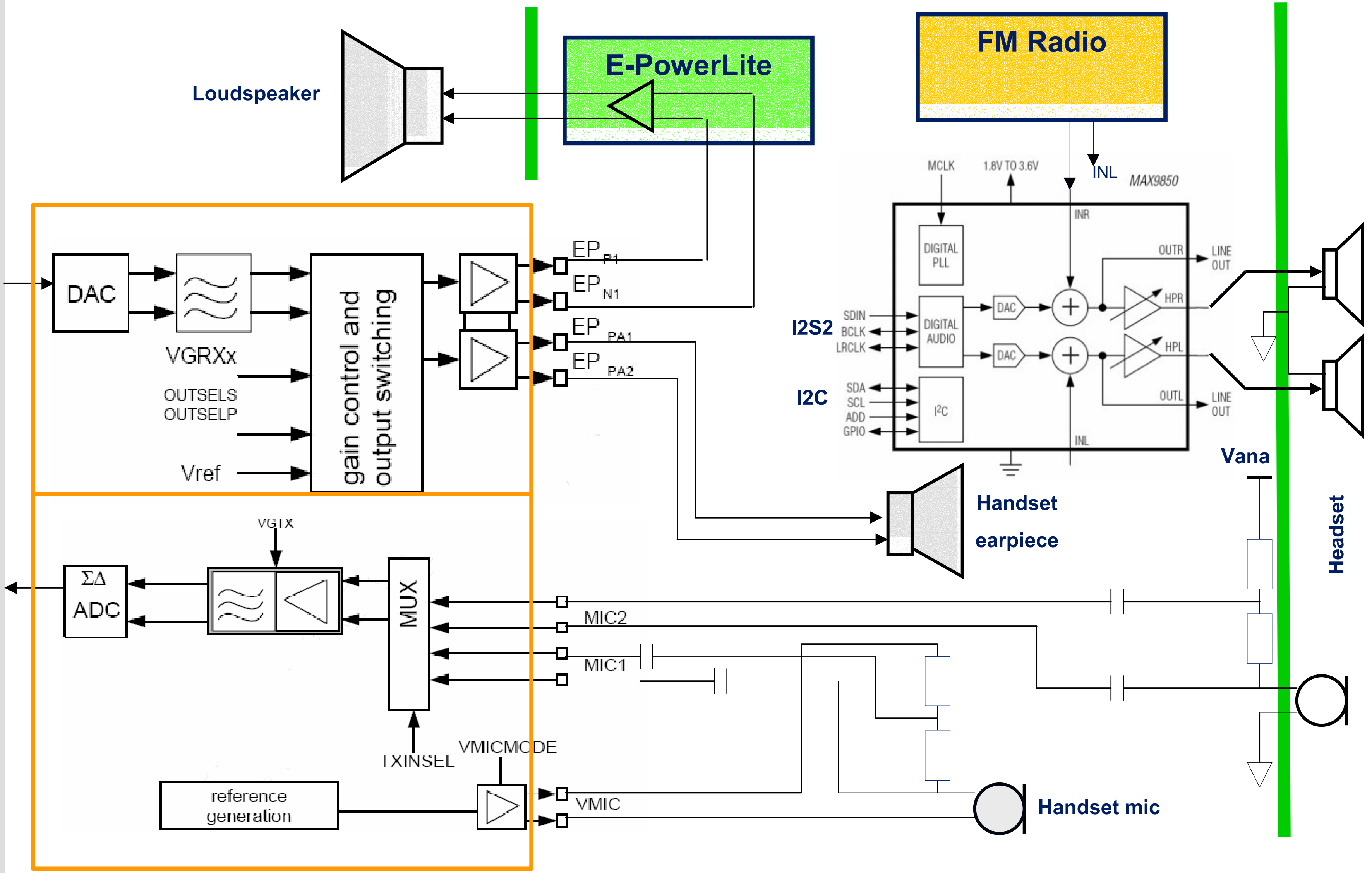
Voiceband processing



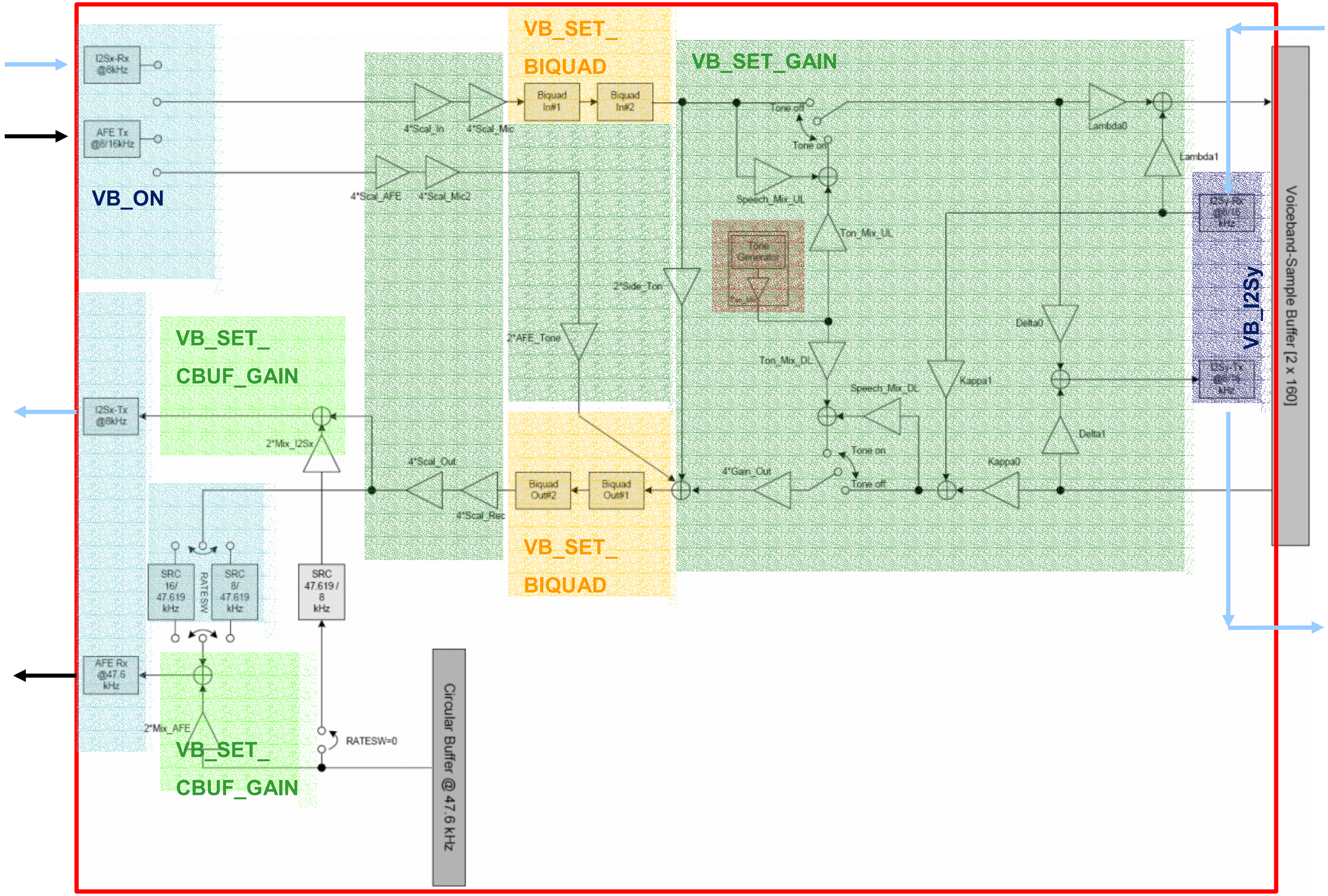
Timing

- Sample-based Voiceband Processing
  (8kHz -125µs / 16kHz- 62.5µs)
- Frame-based Voiceband Processing
  20ms = 160 samples at 8kHz
  320 samples at 16 kHz
- Circular Mixing Buffer 47.6 KHz

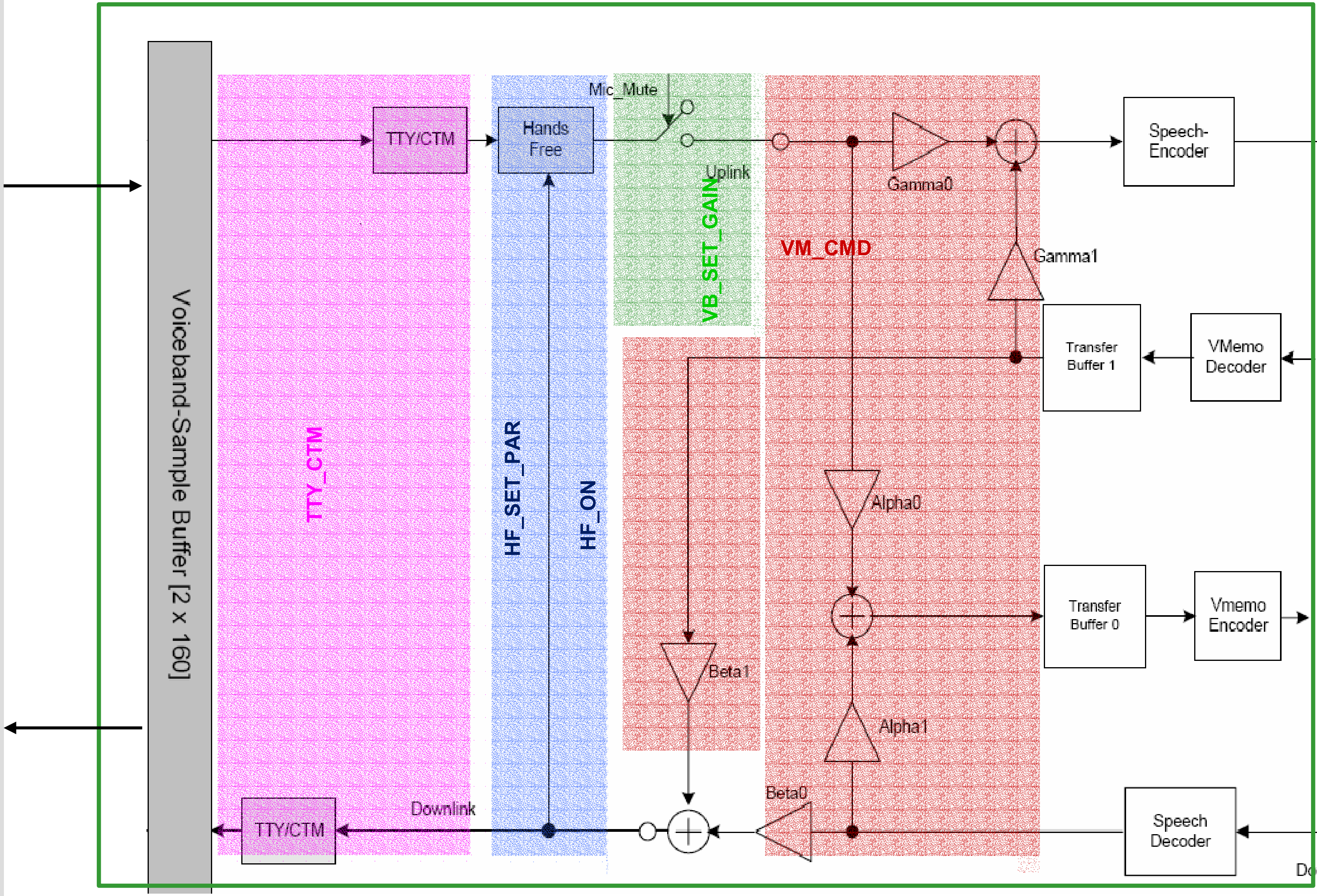
Analog part- BP30 platform-GLOBE 6 board



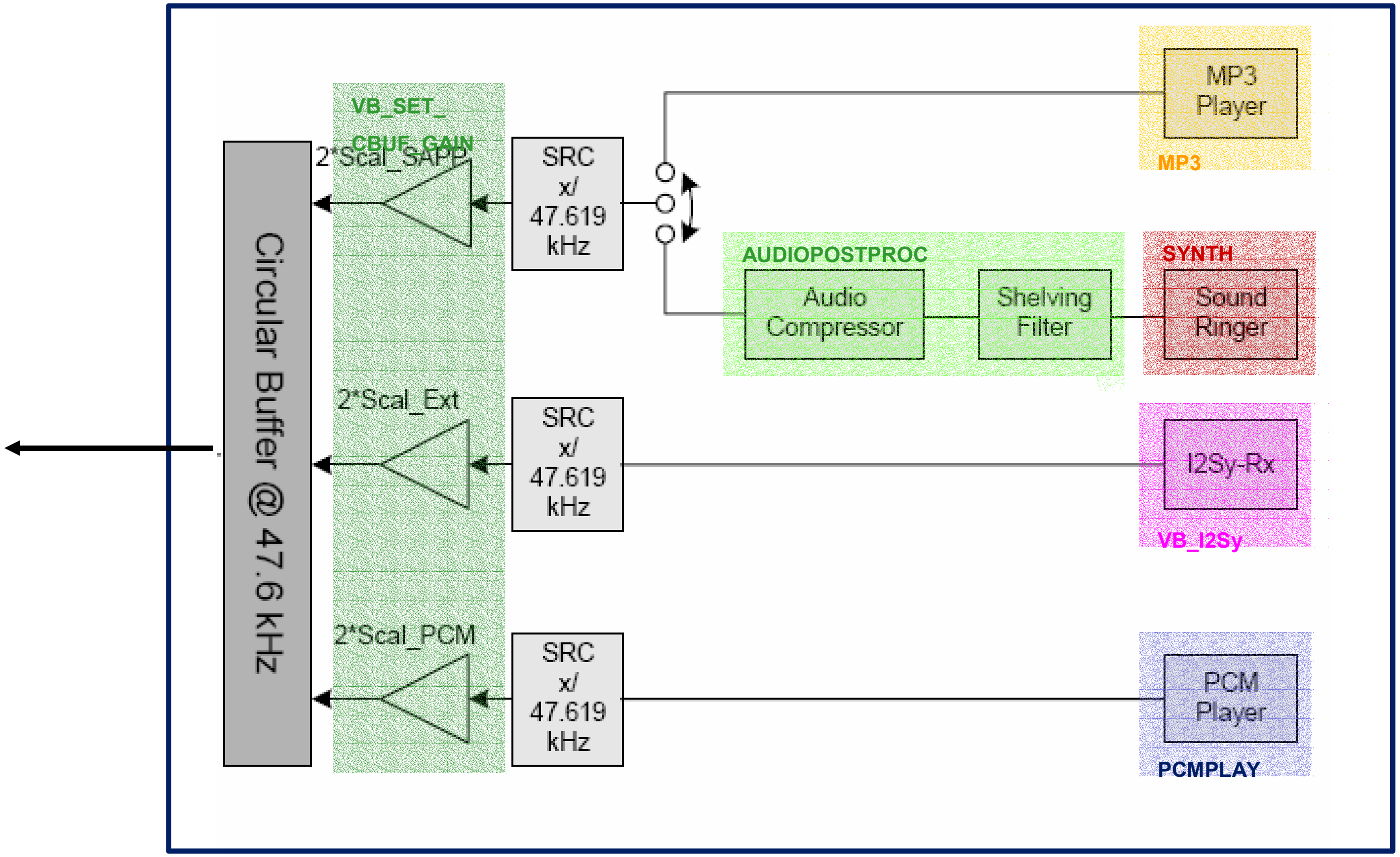
Sample-Based Voiceband Processing: DSP commands



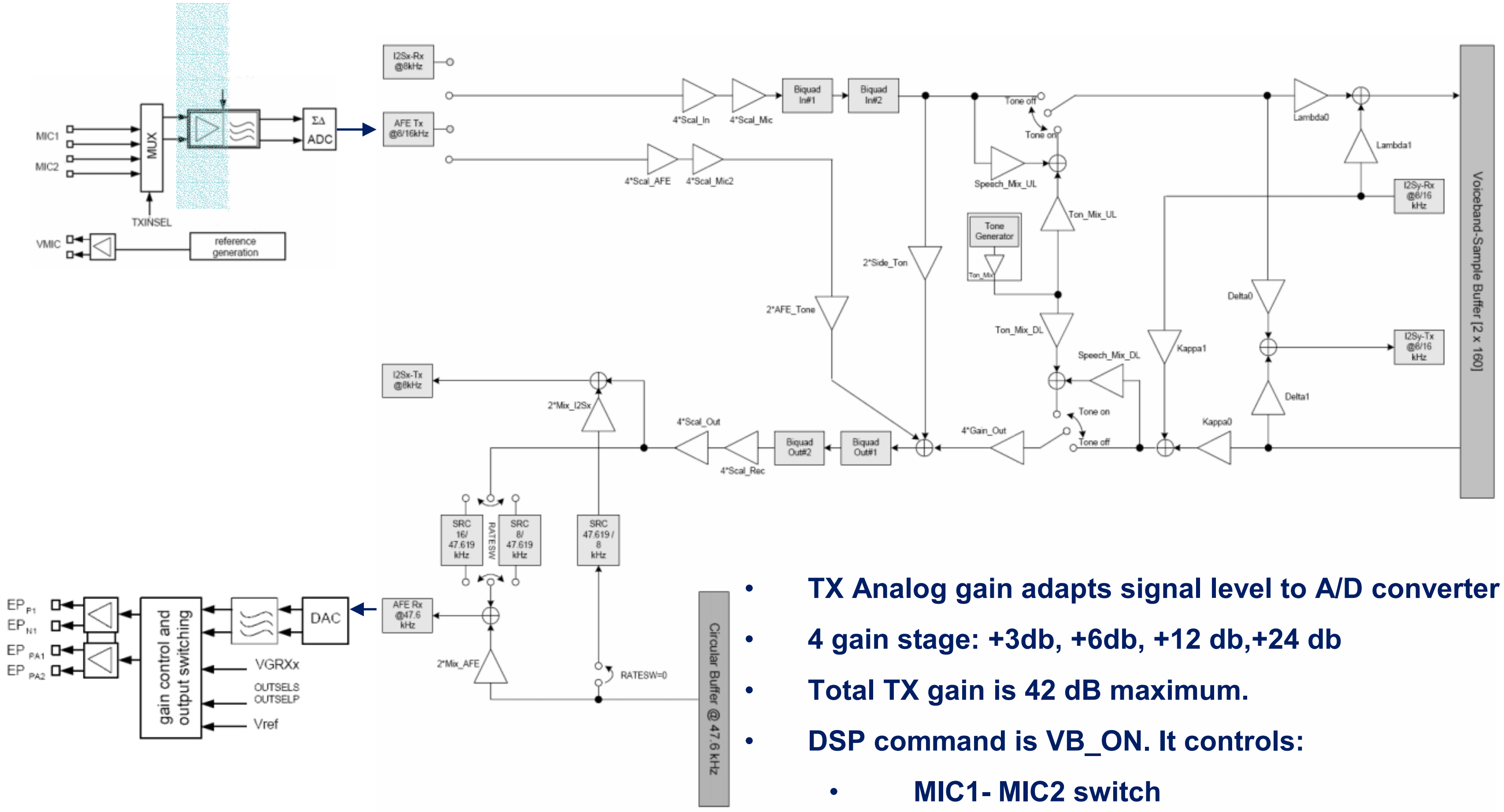
Frame-Based Voiceband Processing: DSP commands



Circular Buffer : DSP commands

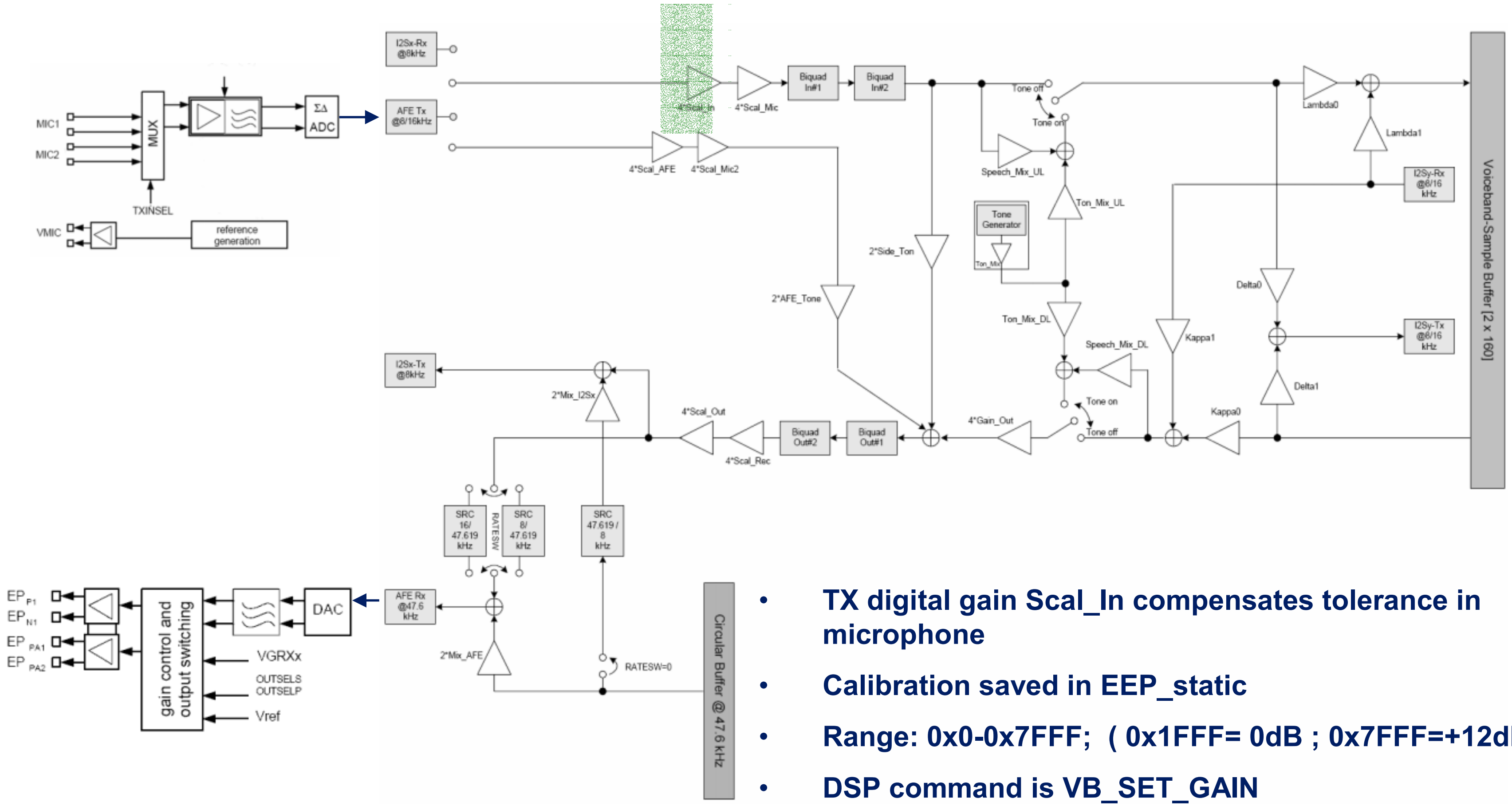


Audio path: TX analog gain



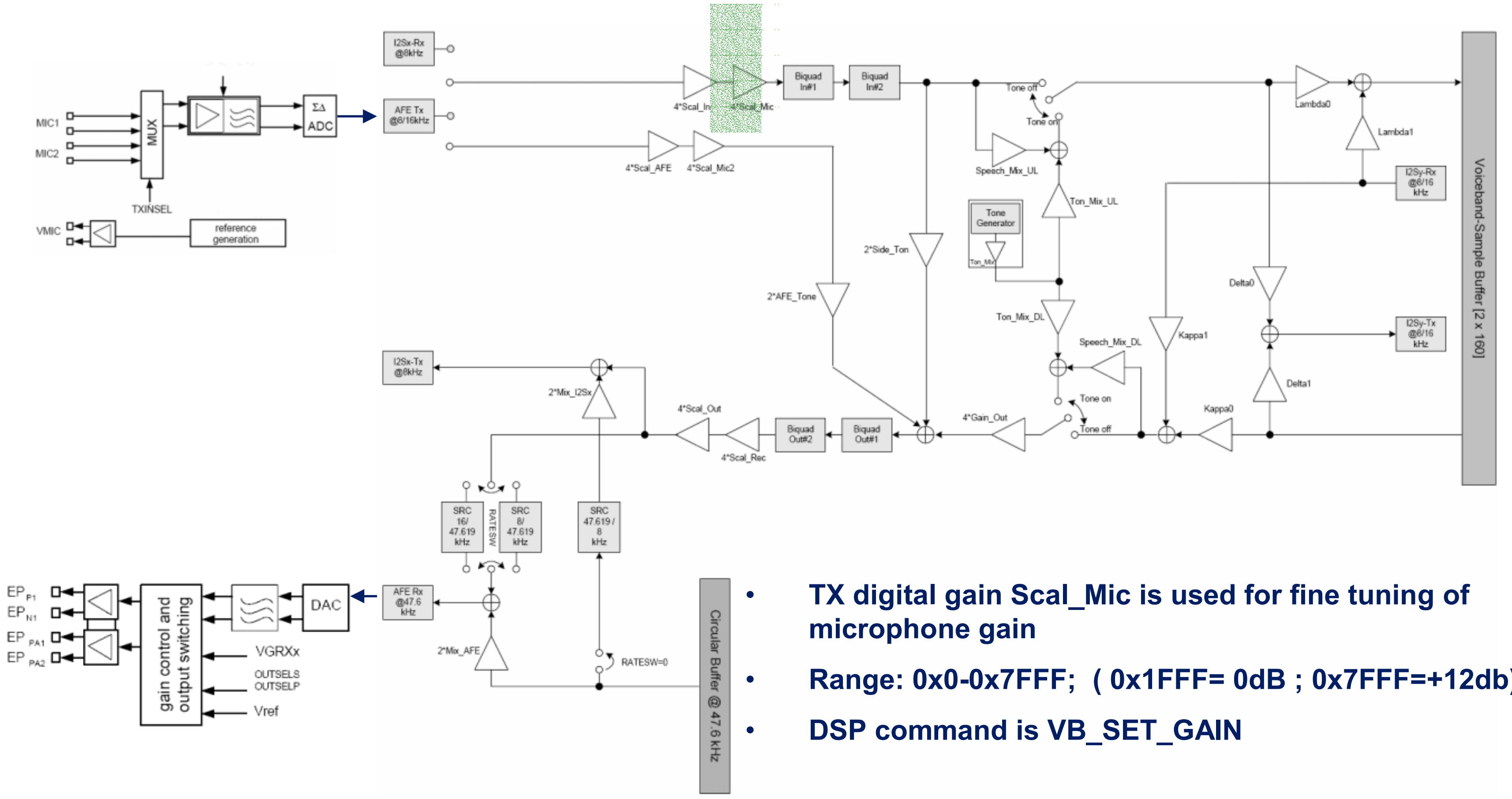
- TX Analog gain adapts signal level to A/D converter
- 4 gain stage: +3db, +6db, +12 db,+24 db
- Total TX gain is 42 dB maximum.
- DSP command is VB_ON. It controls:
  - MIC1- MIC2 switch
  - Mic Supply voltage : 1.8V, 2.0V, 2.2V
  - the sample rate of the AFE input (8kHz/16kHz)

Audio path: TX digital gain



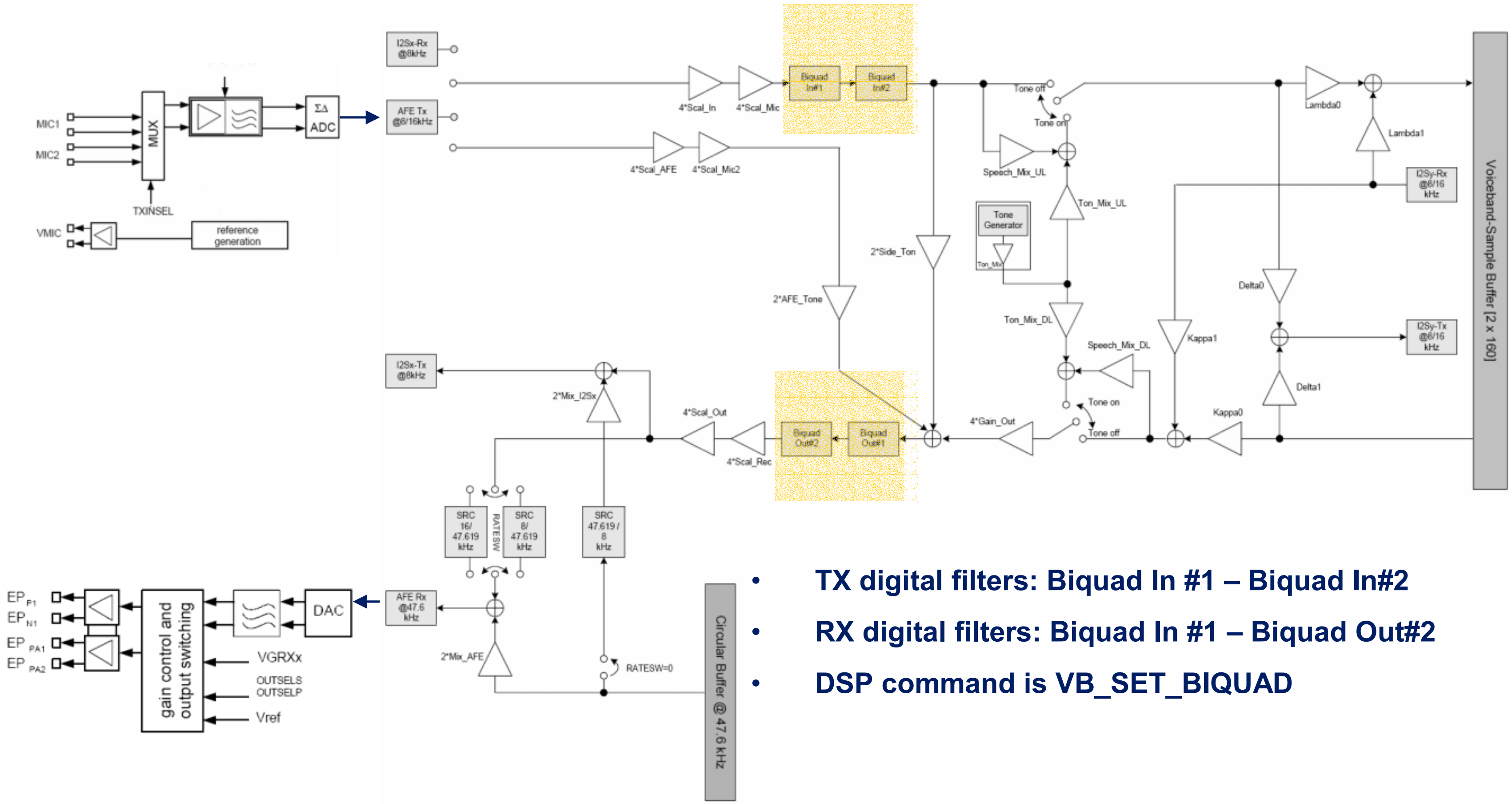
- **TX digital gain Scal_In compensates tolerance in microphone**
- **Calibration saved in EEP_static**
- **Range: 0x0-0x7FFF; (0x1FFF= 0dB ; 0x7FFF=+12db)**
- **DSP command is VB_SET_GAIN**

Audio path : TX digital gain



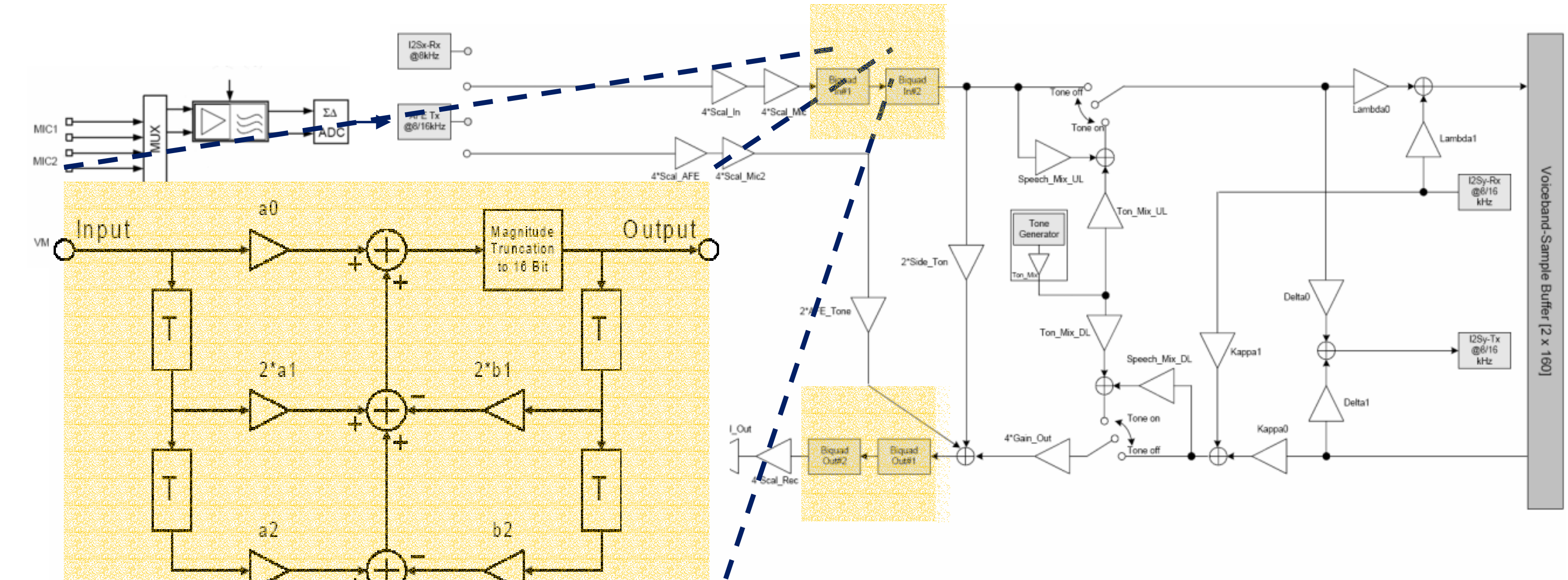
- **TX digital gain Scal_Mic is used for fine tuning of microphone gain**
- **Range: 0x0-0x7FFF; (0x1FFF= 0dB ; 0x7FFF=+12db)**
- **DSP command is VB_SET_GAIN**

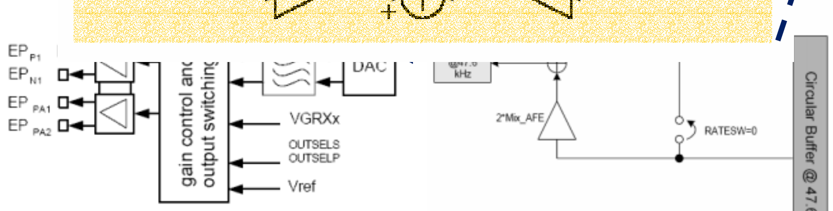
Audio path : Biquad filters



- **TX digital filters: Biquad In #1 – Biquad In#2**
- **RX digital filters: Biquad In #1 – Biquad Out#2**
- **DSP command is VB_SET_BIQUAD**

Audio path : Biquad filters

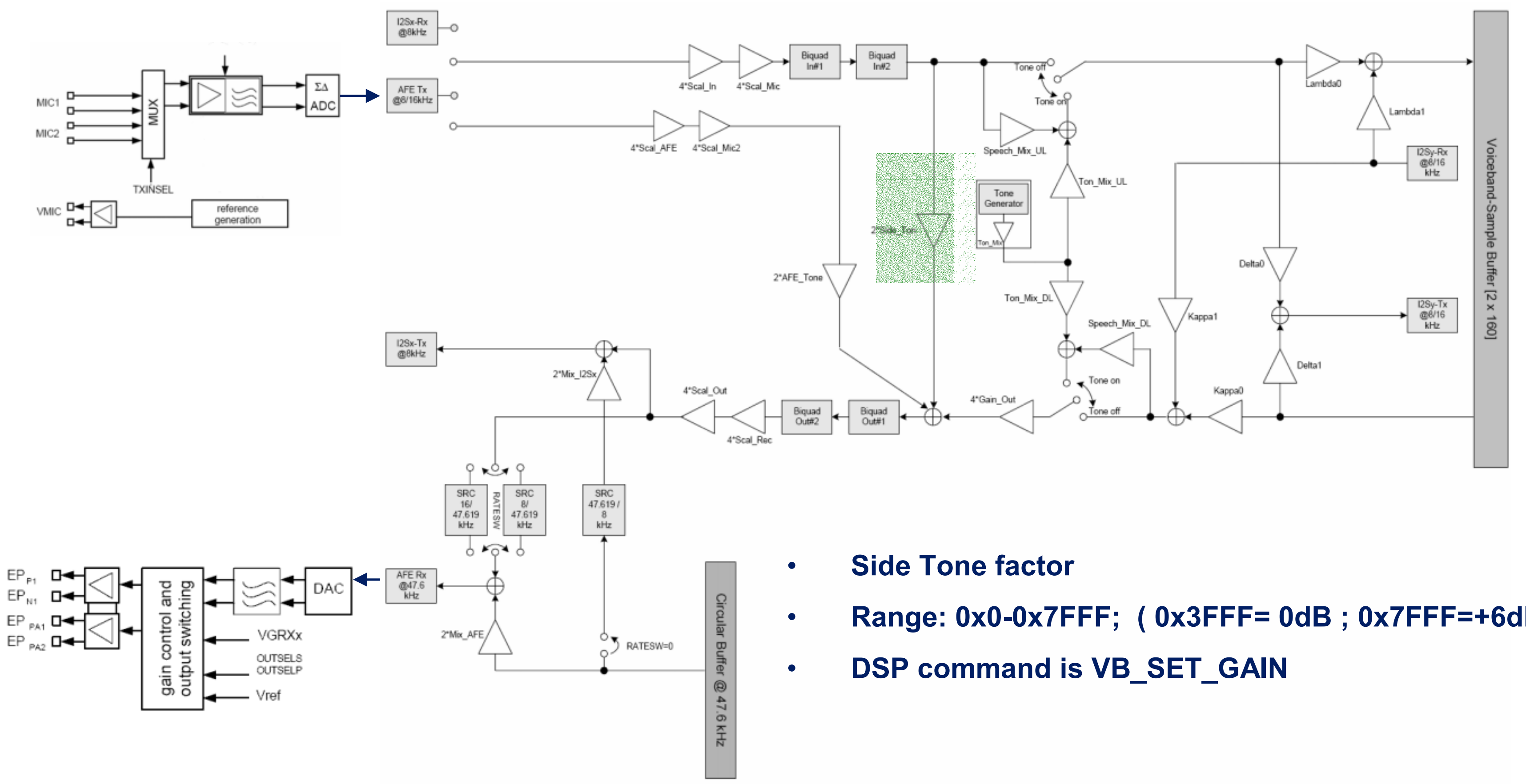




$$\frac{Y(z)}{X(z)} = \frac{a_0 + 2a_1 z^{-1} + a_2 z^{-2}}{1 + 2b_1 z^{-1} + b_2 z^{-2}}$$

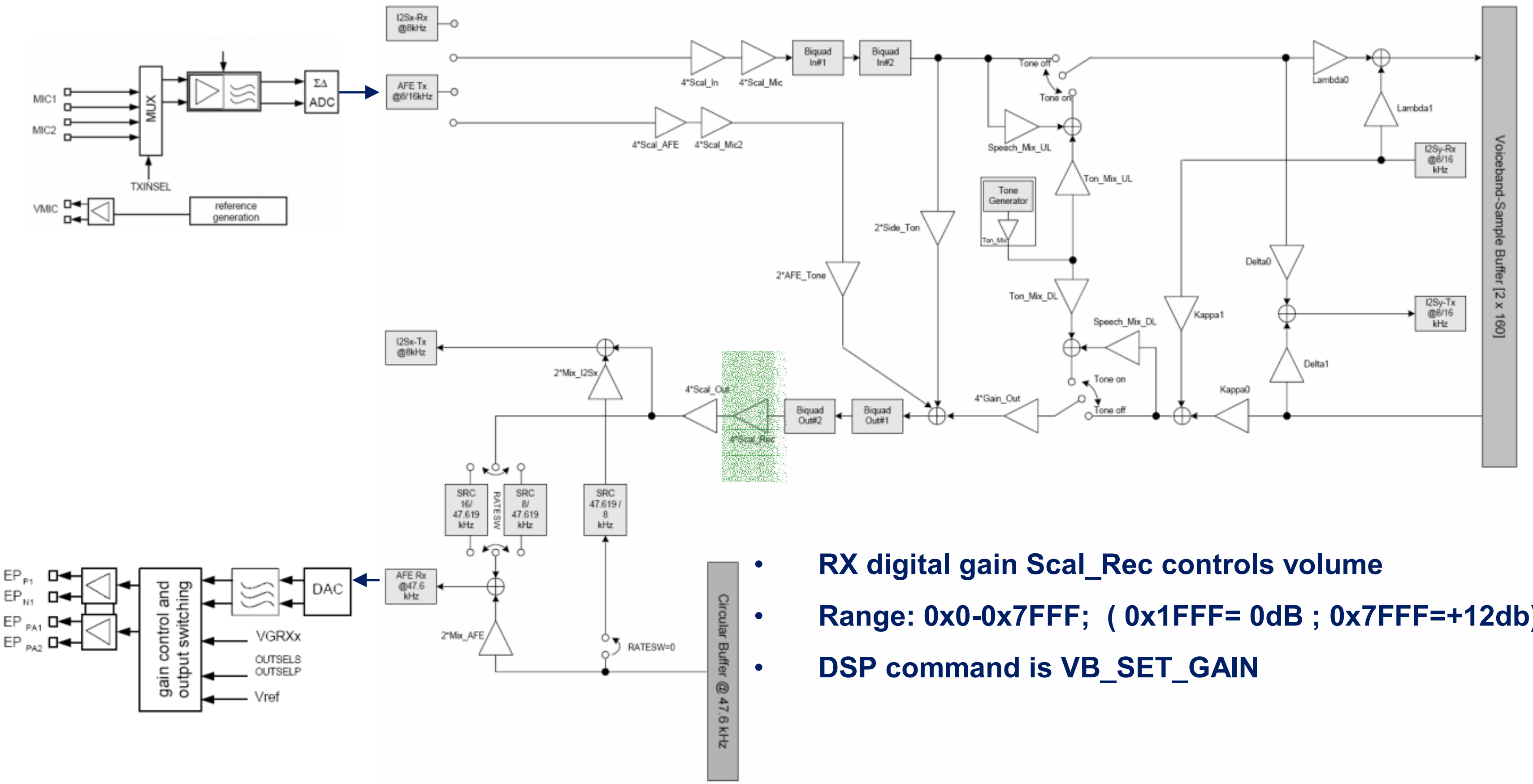
- **TX digital filters: Biquad In #1 – Biquad In#2**
- **RX digital filters: Biquad In #1 – Biquad Out#2**
- **DSP command is VB_SET_BIQUAD**
- **4 x 5 parameters; Range: 0x8000= -1 to 0x7FFF=1**
- **They adjust frequency response and can be configured as LPF, HPF, BPF**

Audio path : Side Tone



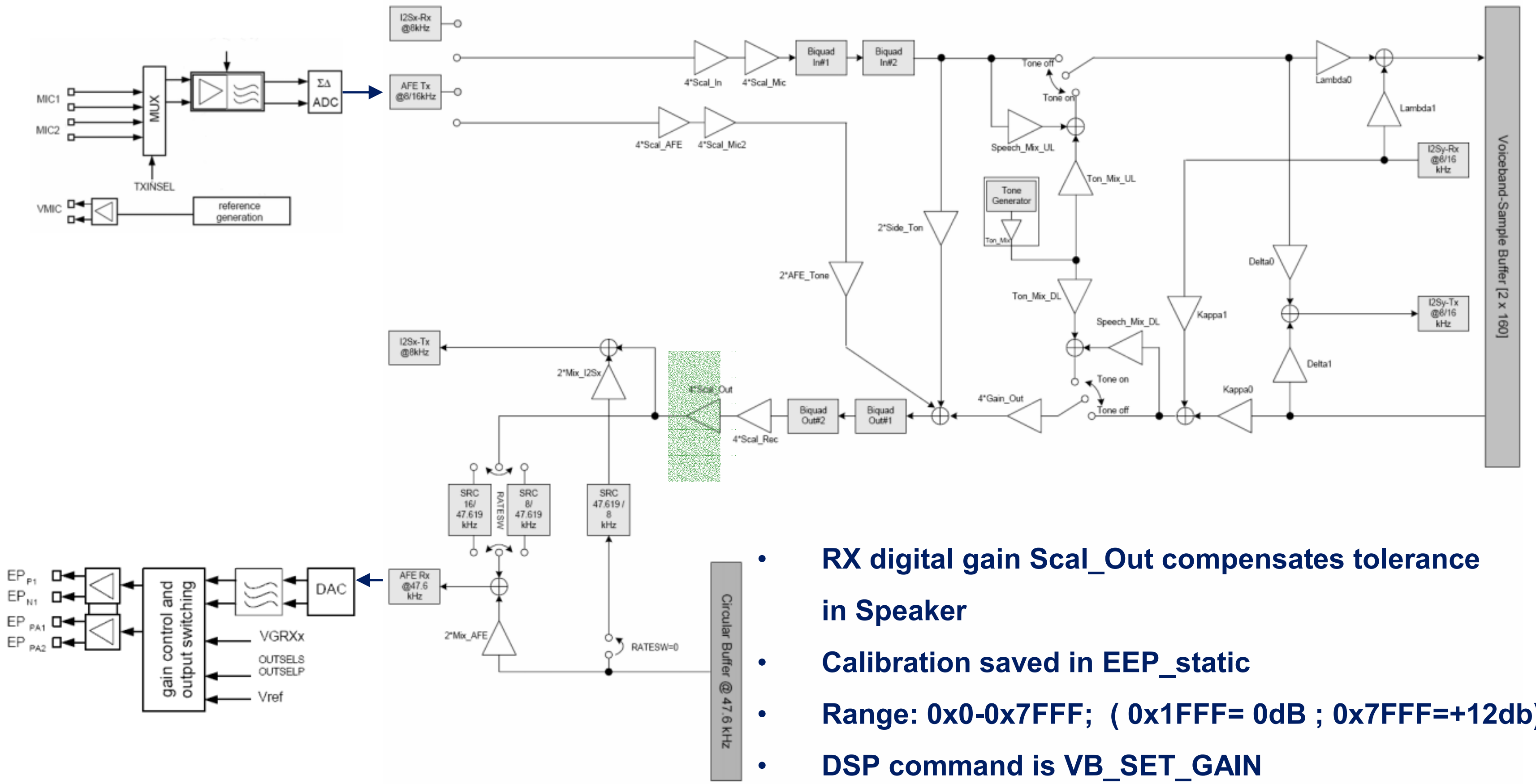
- Side Tone factor
- Range: 0x0-0x7FFF; (0x3FFF= 0dB ; 0x7FFF=+6db)
- DSP command is VB_SET_GAIN

Audio path : RX digital gain



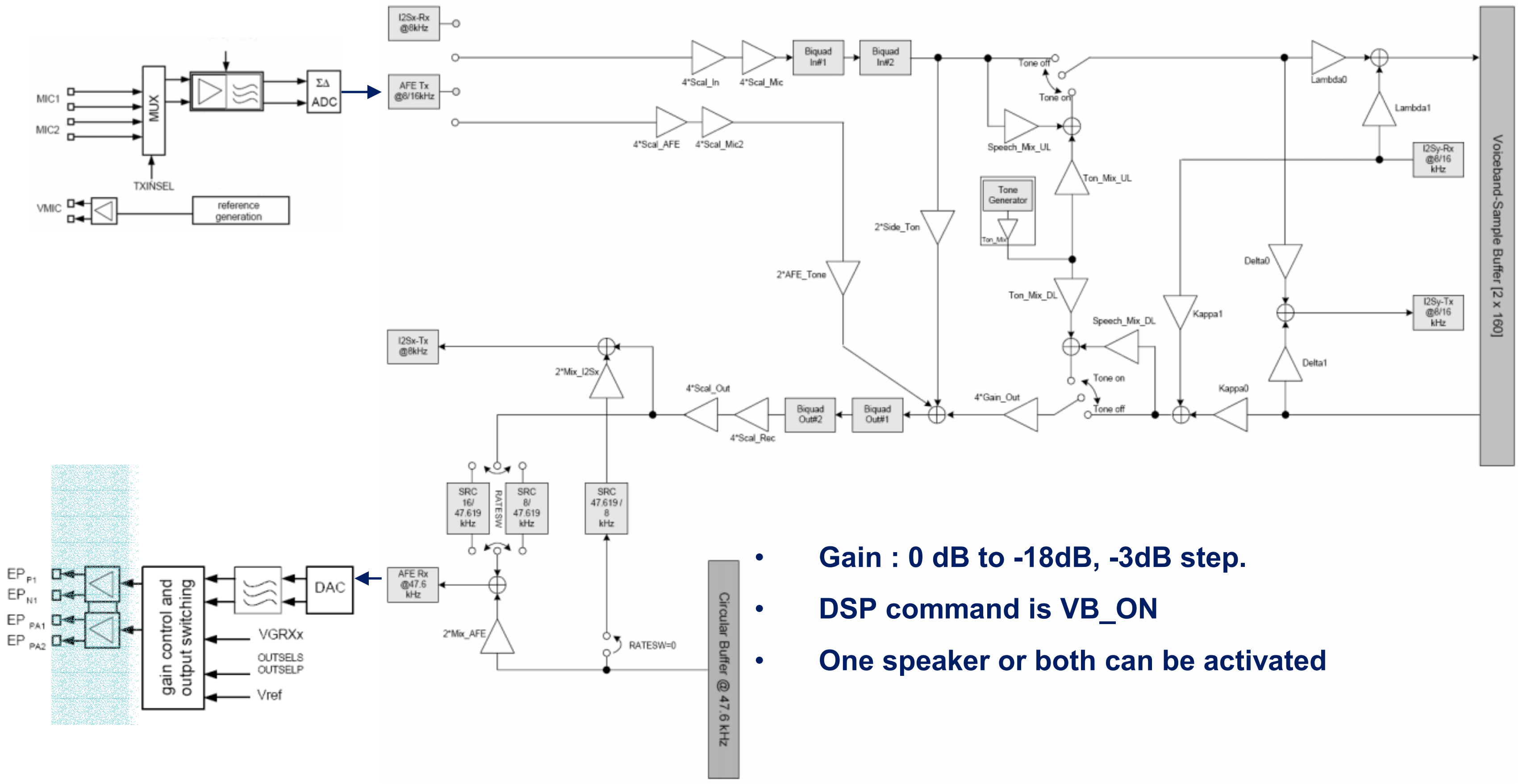
- **RX digital gain Scal_Rec controls volume**
- **Range: 0x0-0x7FFF; (0x1FFF= 0dB ; 0x7FFF=+12db)**
- **DSP command is VB_SET_GAIN**

Audio path: Rx digital gain



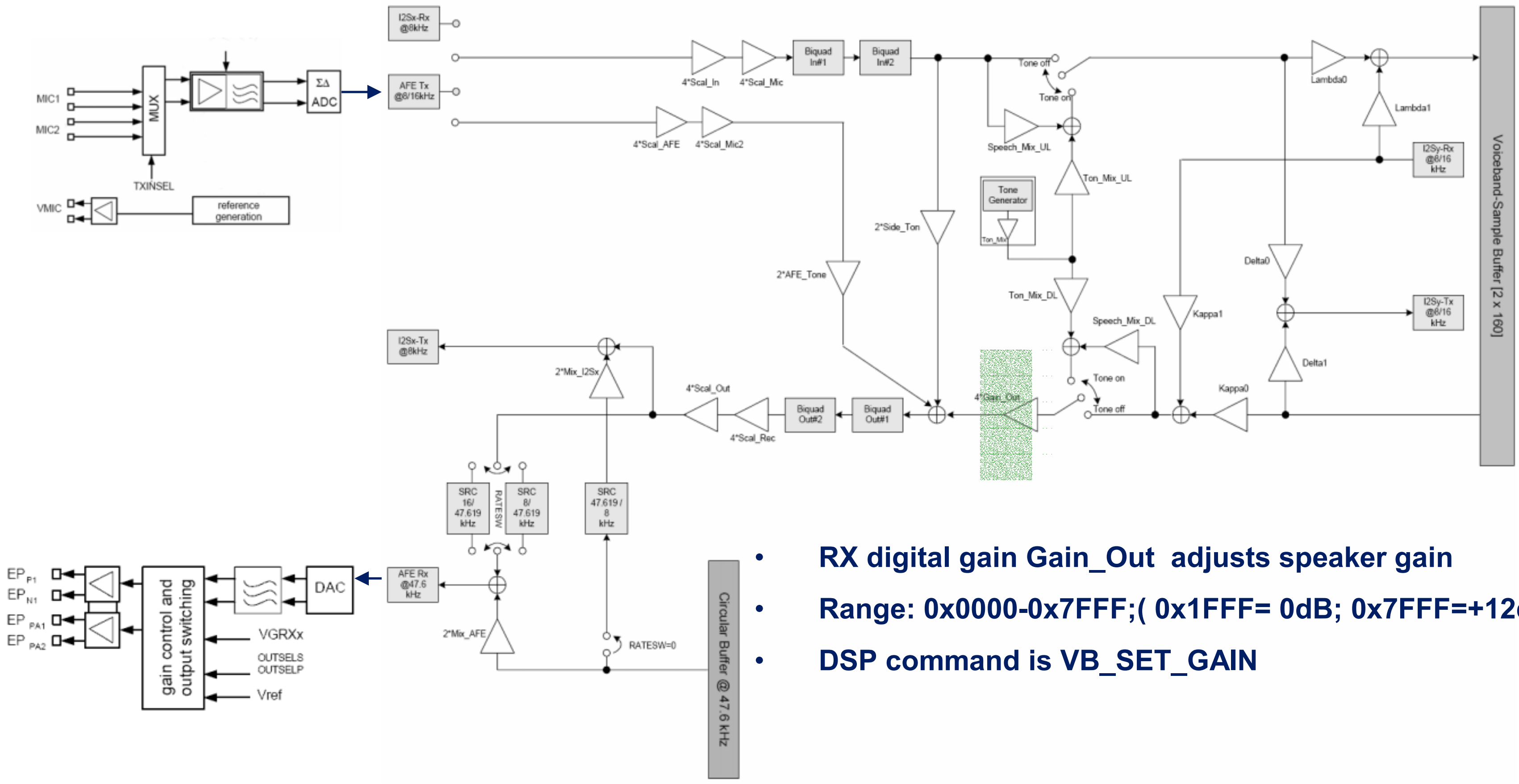
- **RX digital gain Scal_Out compensates tolerance in Speaker**
- **Calibration saved in EEP_static**
- **Range: 0x0-0x7FFF; (0x1FFF= 0dB ; 0x7FFF=+12db)**
- **DSP command is VB_SET_GAIN**

Audio path: RX analog gain



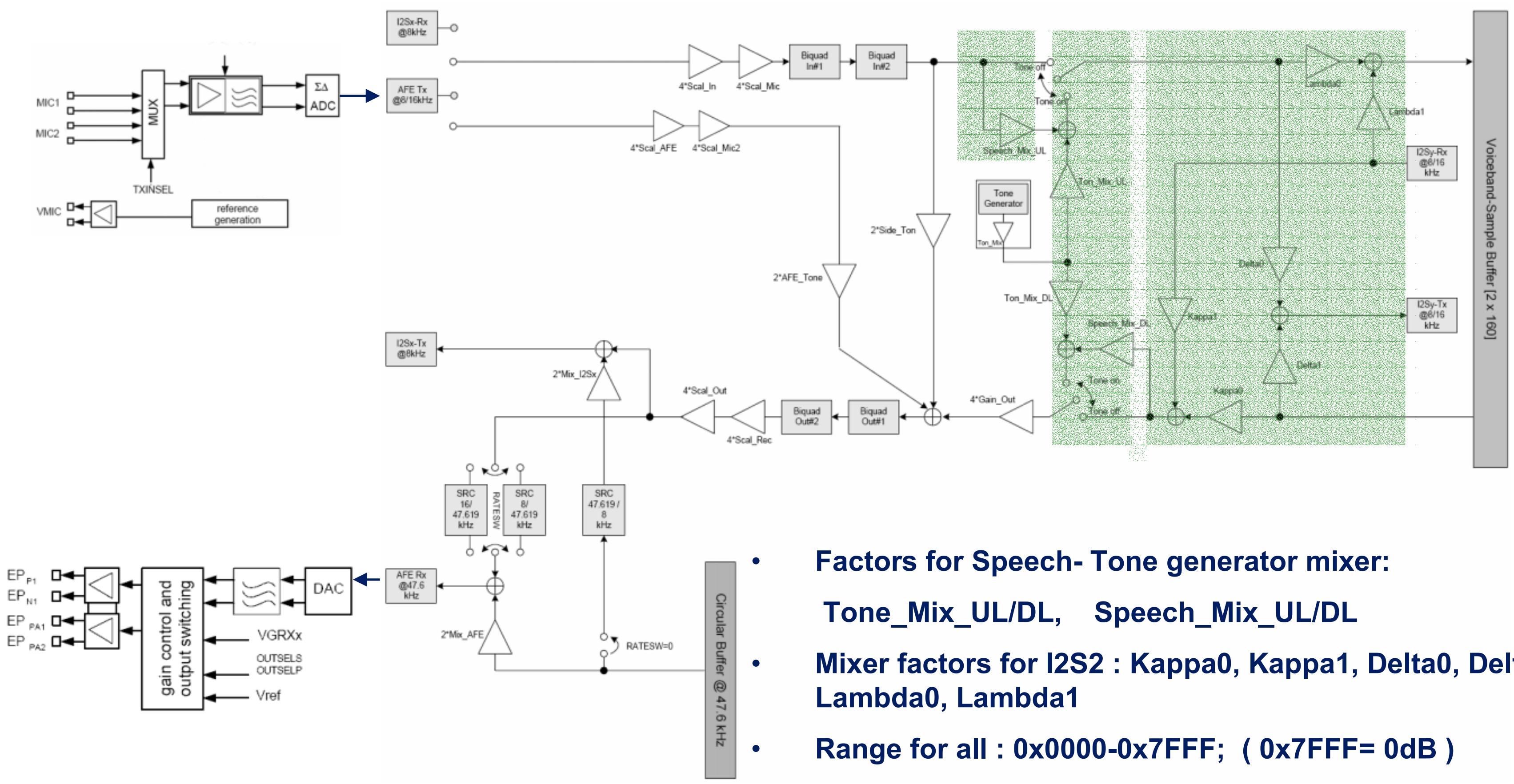
- **Gain : 0 dB to -18dB, -3dB step.**
- **DSP command is VB_ON**
- **One speaker or both can be activated**

Audio path: Gain out



- **RX digital gain Gain_Out adjusts speaker gain**
- **Range: 0x0000-0x7FFF;(0x1FFF= 0dB; 0x7FFF=+12db)**
- **DSP command is VB_SET_GAIN**

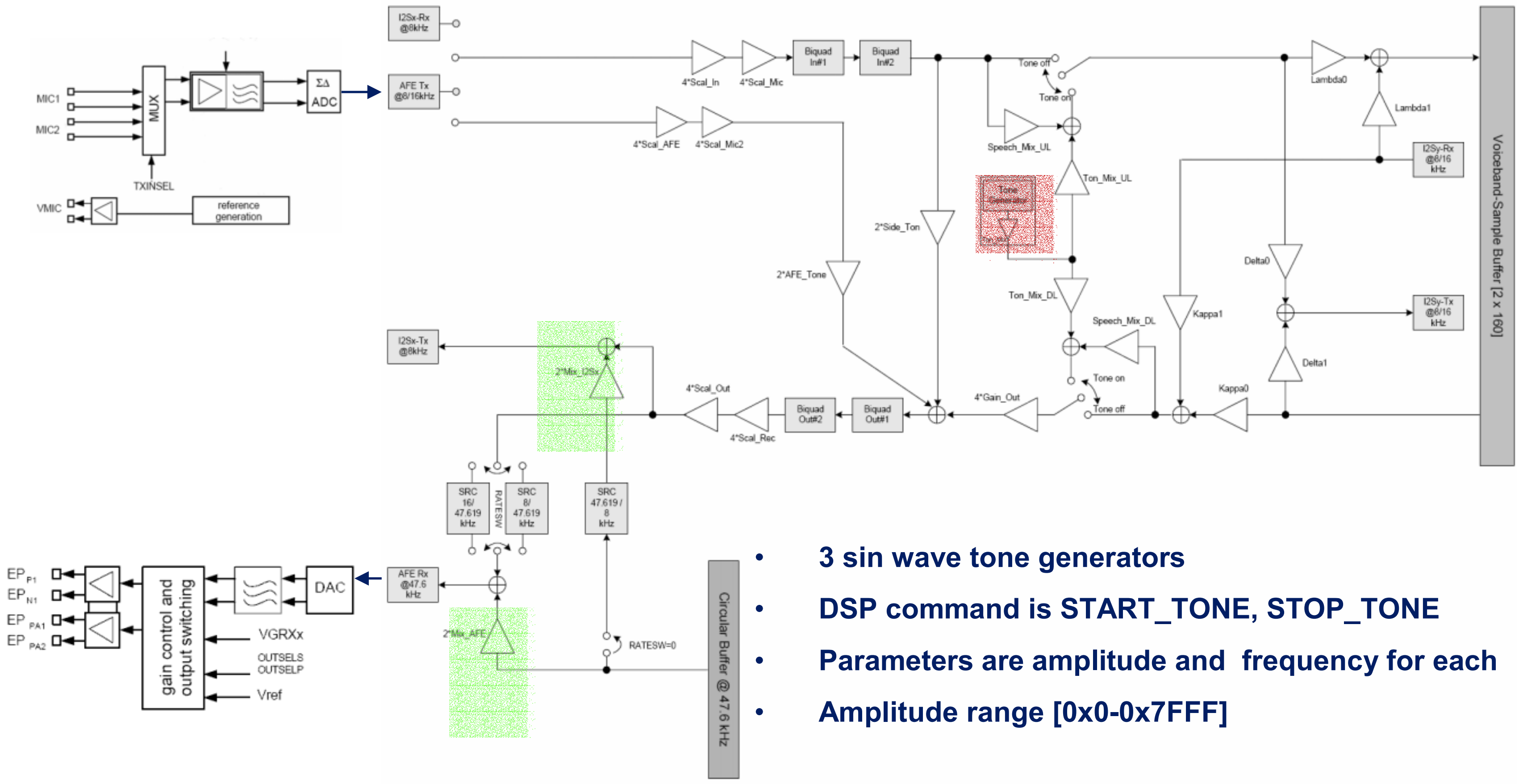
Audio path: Tone generator mixer



- **Factors for Speech- Tone generator mixer:**

  Tone_Mix_UL/DL, Speech_Mix_UL/DL
- **Mixer factors for I2S2 : Kappa0, Kappa1, Delta0, Delta1, Lambda0, Lambda1**
- **Range for all : 0x0000-0x7FFF; (0x7FFF= 0dB)**
- **DSP command is VB_SET_GAIN**

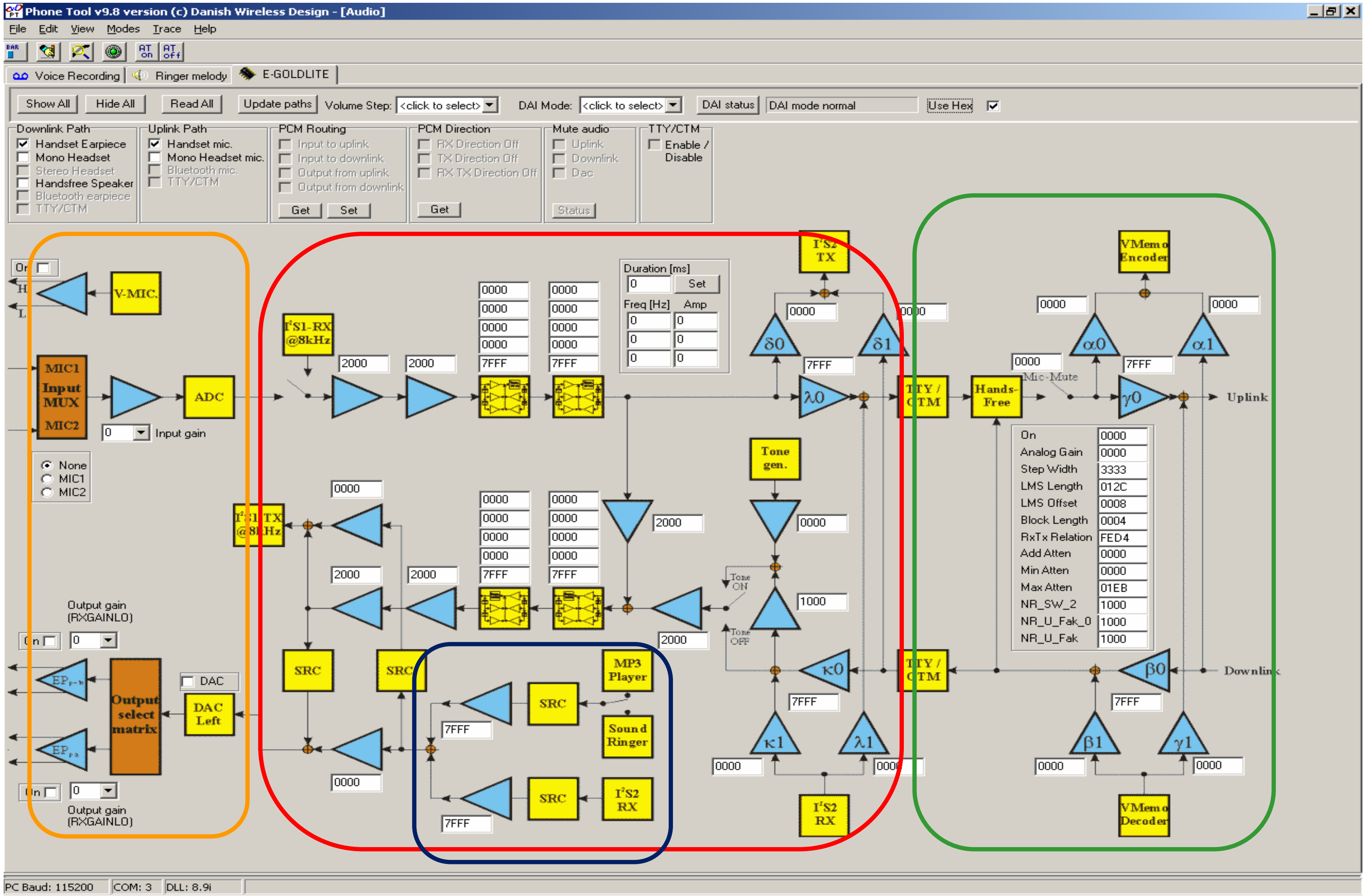
Audio path: Tone generator



- 3 sin wave tone generators
- DSP command is START_TONE, STOP_TONE
- Parameters are amplitude and frequency for each
- Amplitude range [0x0-0x7FFF]

- Mixer factor for circular buffer: Mix_AFE, Mix_I2S1;
- Range [0x0-0x7FFF]; (0x3FFF=0dB; 0x7FFF=+6db)

Phone tool audio interface



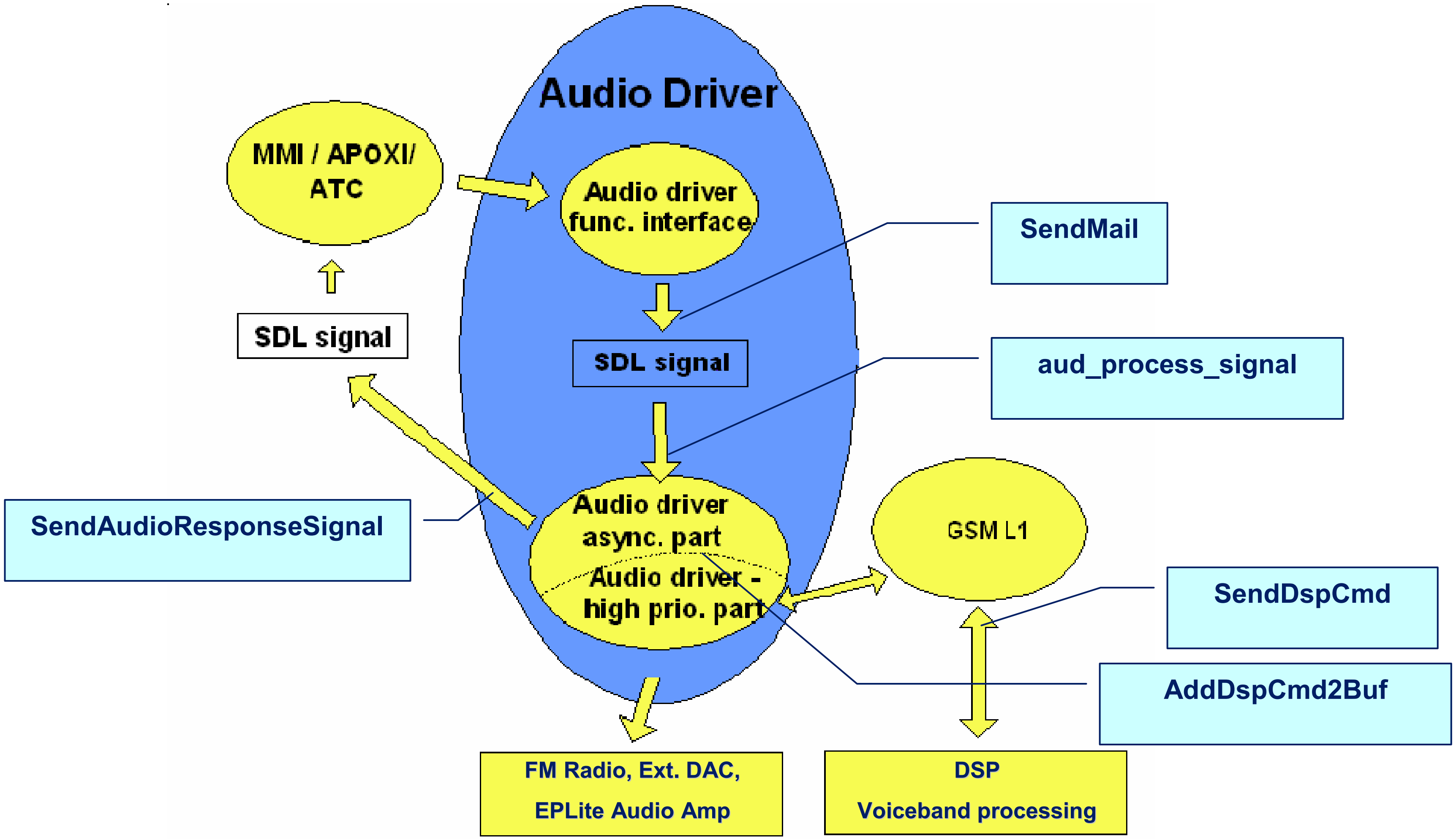
Audio Driver overview

- Driver Sw Version for BP30 platform : N7_BP30_DRV_07.30 (15.02.06)
- Audio driver reference version : DWD MP1G_23.30

(Developed by DWD for SGOLD/ SGOLDLite

modified by N7 for EgoldLite/EgoldRadio)

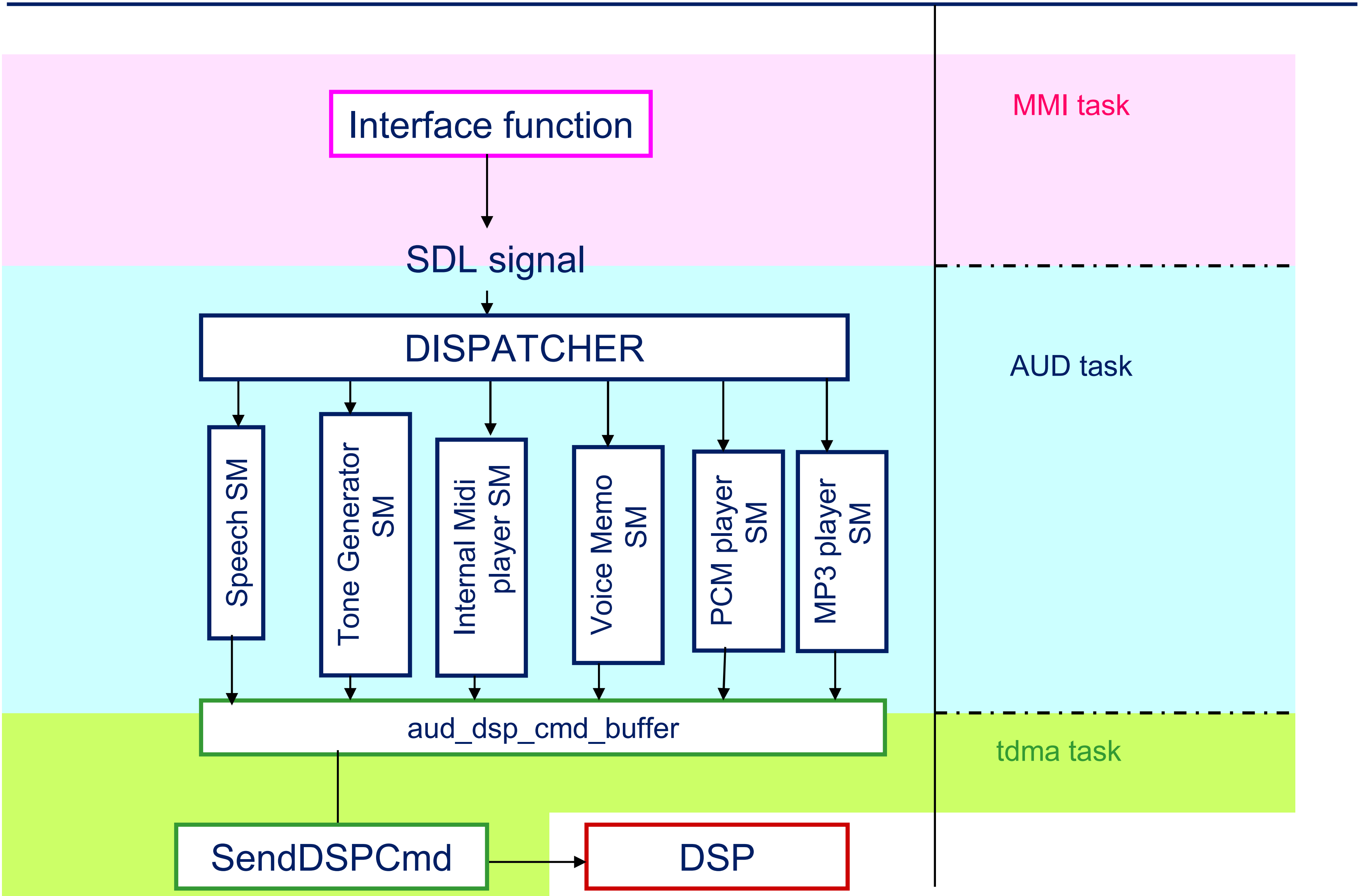
Audio Driver overview



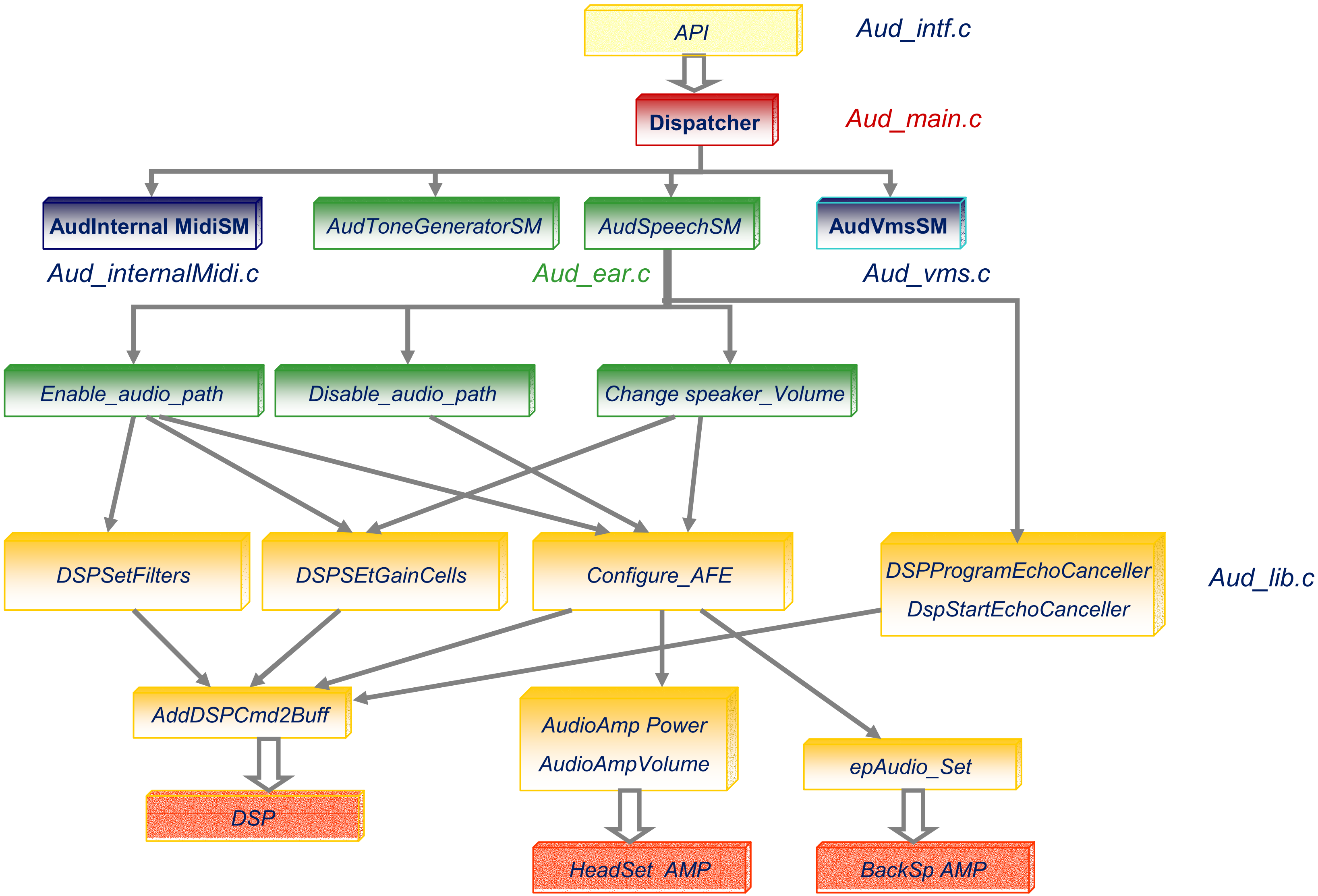
Audio driver files

- **Aud_intf.c** - Interface functions to the clients (MMI, APOXI.)
- **Aud_drv.h** - Interface header file. Contains prototypes for interface functions, enums etc.
- **Aud_main.c** - Main dispatcher
- **Aud_ear.c** - State machines for control of basic audio
- **Aud_internal_midilib.c** - State machine for internal polyringer
- **Aud_vms.c** - State machine for Voice Memo Service
- **Aud_lib.c** - Function library
- **Aud_tone.c** - Definition of tones for the internal tone generator (DTMF- tones, supervisory tones etc.)
- **Aud_pcm_player.c** - State machine for pcm player.
- **Aud_pcm_rec.c** - State machine for pcm recorder .
- **Aud_fmr.c** - State machine for FM radio
- **Aud_tst.c** - Test functions (AT command interface).
- **Aud_data.c** - Audio data structures
- **Aud_com.h** - Header file, only used internally in the audio driver

Audio Driver overview



Audio driver functions- structure example



Audio Driver resources

Each HW block is called a resource. The following resources exist:

- Speech
- Tone generator
- MP3 player
- Internal MIDI polyringer
- Voice Memo playback
- Voice Memo recording
- PCM player
- FM-radio

Interface description

General functions

SINT8 **AUD_allocate_resource**(UINT16 id,
aud_resource_enum resource,
aud_priority_enum priority)

priority : aud_priority_normal
aud_priority_high

SINT8 **AUD_release_resource**(UINT8 handle)

void **AUD_init**(void)

Interface description

Audio path management

SINT8 **AUD_add_uplinkpath**(aud_uplink_source_enum path);

SINT8 **AUD_remove_uplinkpath**(aud_uplink_source_enum path)

SINT8 **AUD_add_downlinkpath**(aud_downlink_source_enum path)

SINT8 **AUD_remove_downlinkpath**(aud_downlink_source_enum path)

aud_uplink_source_enum

- aud_handset_mic
- aud_headset_mic
- aud_I2Sx_rx

aud_downlink_source_enum

- aud_normal_earpiece
- aud_mono_headset
- aud_stereo_headset (dummy)
- aud_backspeaker
- aud_I2Sx_tx

Interface description

- Example

handle = **AUD_allocate_resource** (0, aud_resource_speech, aud_priority_normal);

Return_code= **AUD_speech_enable** (handle);

:

AUD_speech_disable (handle);

AUD_release_resource (handle);

Interface description

DSP Tone generator

SINT8 **AUD_tone_start** (UINT8 handle,
aud_tone_id_enum tone, UINT16 nof_repats,
SINT16 mix_factor)

SINT8 **AUD_tone_start_user_tone** (UINT8 handle,
void *tone_data, aud_tone_type_enum type,
UINT32 nof_tones, UINT16 nof_repeats,
SINT16 mix_factor);

SINT8 **AUD_tone_stop** (UINT8 handle)

SINT8 **AUD_key_tone** (aud_tone_id_enum key_tone, SINT16
mix_factor)

Interface description

Example

```
handle = AUD_allocate_resource (0, aud_resource_tone_generator,
                                   aud_priority_normal);

AUD_tone_start (handle, aud_tone_info_low_bat_in_call, 1, 0x4000);   /*
    Remenber to check the return code */

:

AUD_release_resource (handle);
```

Interface description

Midi player

SINT8 **AUD_midi_start** (UINT8 handle, aud_ringer_tone_id_enum midi_id, UINT16 nof_repeats);

SINT8 **AUD_midi_start_user_melody** (UINT8 handle, UINT8 huge *melody_data, UINT32 size, aud_format_enum format, UINT16 nof_repeats);

SINT8 **AUD_midi_stop** (UINT8 handle);

SINT8 **AUD_midi_suspend** (UINT8 handle, UINT8 SlotID);

SINT8 **AUD_midi_resume** (UINT8 handle, UINT8 SlotID)

SINT8 **AUD_midi_stop_suspend** (UINT8 handle, UINT8 SlotID)

Audio parameters location in source code

File: aud_data.c:

const aud_setting_type **AUD_setting**={….. }

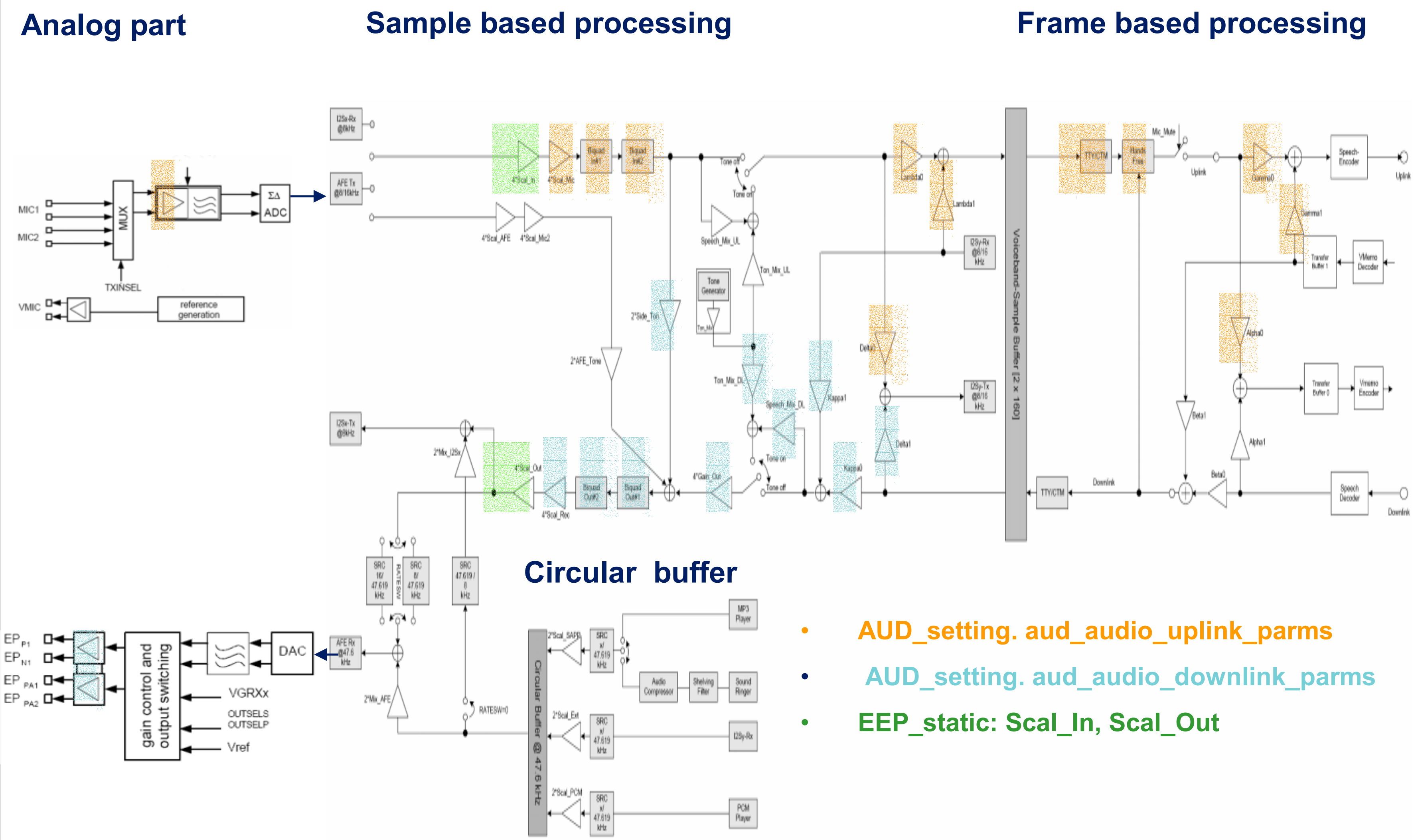
File: eep.c:

const eep_static_type **EEP_static _at**(EE_STATIC_BLOCK_ADDRESS) ={ ….. }

File: eep.h

```
typedef struct {
  eepchr_default_charger_parms_type         chr_charger;
  eepchr_default_capacity_parms_type        chr_capacity;
  eepaud_default_audio_uplink_parms_type    aud_audio_uplink_parms[3];
  eepaud_default_audio_downlink_parms_type  aud_audio_downlink_parms[5];
  eeplcd_default_display_parms_type  lcd_parms;
} eep_default_type;
```

Audio parameters location in source code



- AUD_setting. aud_audio_uplink_parms
- AUD_setting. aud_audio_downlink_parms
- EEP_static: Scal_In, Scal_Out

Voice memo 1

- The voice memo is used for **recording** and **playback** of voice
- The voice memo can be used in both **idle** and **Tch26** mode
- **Full rate** and **AMR** speech codec are available.
- In AMR format **8** different sample rates are available.

Rate parameter	Sampling rate [kBit/sec]	Bytes pr. Speech frame	Bytes pr. Sec.	Bytes pr. Minute
0	4.75	13	650	38.1 KB
1	5.15	14	700	41.0 KB
2	5.90	16	800	46.9 KB
3	6.70	18	900	52.7 KB
4	7.40	20	1000	58.6 KB
5	7.95	21	1050	61.5 KB
6	10.20	27	1350	79.1 KB
7	12.20	32	1600	93.8 KB
Full rate	Full rate	34	1700	99.6 KB

Voice memo 2

- Recordings can be made either to **RAM** or directly to flash memory (**FFS**).
- When using AMR, 1 min. voice recorded takes up between 38-99.6 kbytes memory (configurable)
- File: **Aud_vms.c** – State machine and control functions for voice memo.

Audio Driver – voice memo
Interface function

- SINT8 **AUD_vm_start_recording**(UINT8 handle, aud_vm_mode_enum vm_mode, aud_media_enum media_type, aud_dsp_format_enum format, UINT8 rate, UINT16 *file_handle, UINT16 buffer_size);

```
typedef enum {
    aud_vm_mode_standby,
    aud_vm_mode_tch }
    aud_vm_mode_enum;
```

```
typedef enum {
    aud_media_ffs,
    aud_media_ram,
    aud_media_mmf }
    aud_media_enum;
```

```
typedef enum {
    aud_dsp_format_fr,
    aud_dsp_format_amr }
    aud_dsp_format_enum;
```

- SINT8 **AUD_vm_stop_recording**(UINT8 handle)

- SINT8 **AUD_vm_start_playback**(UINT8 handle, aud_vm_mode_enum vm_mode, aud_media_enum media_type, aud_dsp_format_enum format, UINT16 *file_handle, UINT16 buffer_size);

- SINT8 **AUD_vm_stop_playback**(UINT8 handle)

Audio Driver – voice memo
Interface function - example

Record a voice message:

```
/* First create and open a file in FFS for streaming */

handle = AUD_allocate_resource(<mmi_id>,
    aud_resource_recording,     aud_priority_normal);

AUD_vm_start_recording(handle, aud_vm_mode_standby,
    aud_media_ffs, aud_dsp_format_amr, rate, ffs_file_ptr,
    Buffer_Size);   /* Remember to check the return code */

....

....

AUD_vm_stop_recording(handle); /* HW is powered down */

AUD_release_resource(handle);
```

Handsfree functionality

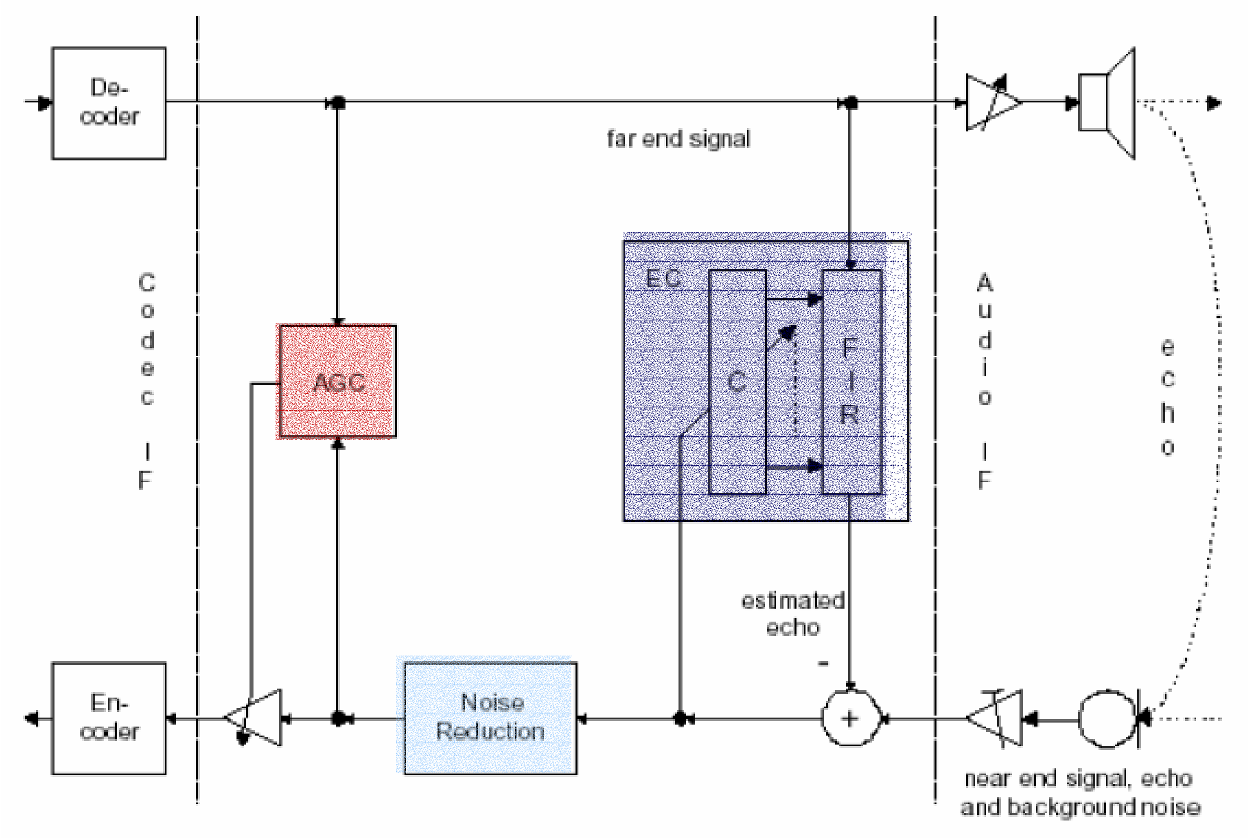
- Handsfree functionality is a FW application to reduce acoustic echoes inserted at the microphone by an acoustic coupling beetween loudspeaker and microphone of the mobile.
- It consists of three independently operating parts:

  Echo Canceller (EC), time domain based Block-NLMS (normalized least mean square) algorithm

  Automatic Gain Control (AGC)

  Noise Reduction (NR)
- These handsfree's functional blocks are indipendent each other. They all contribute to echo suppression.They only affect signals within Audio Scheduler TX path.
- Handfree works on a 20ms frame base processing 160 samples per routine call.
- Handsfree is controlled by the **HF_ON** command and its behaviour is controlled by parameters given via command **HF_SET_PAR.**

Handfree block diagram



Echo Canceller parameters

- LMS_OFFSET range: 0,400. default:8

  Number of samples in FIR filter delay line.

- LMS_LENGTH range: 2,400. default:300

  LMS FIR filter length, in samples. (400*125uS=50mS max)

  limitation: 2<= LMS_LENGTH+ LMS_OFFSET<=400

- BLOCK_LENGHT range:2,4,5,8. default :4

  LMS coefficent adaptation block lenght. The higher this number, theslower but more accurate the adaptation converge.

- STEP_WIDTH range 0,32767 ; default : 13107 (0x3333)

  The higher this value, the faster the echo characteristic gets adapted. Limit: STEP_WIDTH *BLOCK_LENGTH<=2*32767

AGC parameters

- RXTX_RELATION range :-960,960 ; default : -300

  This parameters rules the power relation between loudspeaker and microphone. In the worst echo case, must be:

  MicPower(dB) + RxTx(dB) < LoudspPower(dB) (AGC runs otherwise)

  with RxTx[dB]=RXTX_RELATION*3/32

  This is the most critical parameter in handfree. Values typical for handset are in range 50 to 150. For backspeaker: -100 to -400

- HF_MIN_ATTEN range: 0,960. default:0

  Minimal attenuation of the mic signal by the AGC.

  Level(dB)=3/32* HF_MIN_ATTEN

- HF_MAX_ATTEN range: 0,960. default:491

  Maximal attenuation of the mic signal by the AGC.

  Level(dB)=3/32* HF_MAX_ATTEN

- ADD_ATTEN range: 0,960. default:0

  When AGC decides to attenuate, ADD_ATTEN is added to the calculated attenuation.

Noise Reduction parameters

- NR_SW_2 range: 0,0x7FFF. default: 4096

  Max attenuation. Linear with 0x7FFF means 1 (0dB)

  Ex. 0x4000= 0.5 = -6dB

- NR_U_FAK_0 range: 0,0x4000. default: 4096

  Factor of NR in the band 0 (0Hz-250 Hz). Linear with 0x4000 for 1

  An increased factor causes a better NR but also speech distortion

- HF_NR_U_FAK range: 0,0x4000. default: 4096

  Factor of NR in the bands 1 to 7 (250Hz -3750Hz). Linear with 0x4000 for 1

End of Audio Driver Presentation