MCU-L1 audio features

L1M_AS001-1

Ver 3.1

Department: European Wireless Terminal Chipset Business Unit.

	Originator
Name	Francois Mazard
Date	01/29/01

HISTORY

Version	Date	Author	Approval manager	Approval date	Notes
Ver: 1.0	01/29/2001	Francois Mazard	Alain Vallauri	01/30/2001	1
Ver: 1.1	02/21/2001	Francois Mazard	Katrin Matthes	02/22/2001	2
Ver: 2.0	03/15/2001	Francois Mazard	Alain Vallauri	03/21/2001	3
Ver: 2.1	04/11/2001	Francois Mazard	Alain Vallauri	04/11/2001	4
Ver: 2.2	05/02/2001	Francois Mazard	Alain Vallauri	05/04/2001	5
Ver: 2.3	06/14/2001	Francois Mazard	Alain Vallauri	06/14/2001	6
Ver: 2.4	08/23/2001	Francois Mazard	Alain Vallauri	08/23/2001	7
Ver : 2.5	10/31/2001	Francois Mazard	Alain Vallauri	10/31/2001	8
Ver : 2.6	11/17/2001	Francois Mazard	Frederic Turgis	06/21/2002	9
Ver : 2.7	10/02/2002	Frederic Turgis	K. Matthes/G. Dassot	10/09/2002	10
Ver : 2.8	10/30/02	Frederic Turgis	F. Mazard	11/07/02	11
Ver : 2.9	12/09/02	Frederic Turgis	K. Matthes	01/17/03	12
Ver : 2.10	02/17/03	Fabrice Olivero	F. Turgis	02/18/03	13
Ver : 2.11	05/27/03	Frederic Turgis	K. Matthes	06/23/03	14
Ver : 2.12	08/21/03	Frederic Turgis	K. Matthes	09/03/2003	15
Ver: 3.0	01/20/03	Frederic Turgis	Fabrice Olivero	02/19/2004	16
Ver: 3.1	27/09/2005	Daniel Ezekiel	Francois Mazaed	27/09/2005	17

NOTES :

1. Creation.
2. Change the Melody task: add the initialization phase (the bit B_MELO is added: the bit is a switch for the DSP to know if it's a tone or a melody).
   Change the tone and tone uplink task in order to clear the B_MELO.
   Minor changes of the speech reognition reco task after the implementation: the result of the recognition task is available and good when the last processing is done.
   Minor changes of the speech recognition update task after implementation: the DSP alignment task could be done in only 1 frame therefor it's impossible to detect when this task starts. The L1S state WAIT_DSP_ALIGNMENT_TASK_START is deleted.

Changes in the speech recognition processing task: the processing task could be done in 1 frame so the MCU can't see when it starts. It just see when it stops. Therefor, the code to check if the processing task is started (i.e. L1A state: WAIT_PROCESSING_START, L1S state WAIT_DSP_PROCESSING_START and the message L1_SR_PROCESSING_START_CON) are deleted.
Improve the understanding of the *Cust_srback_save_speech* speech recognition background function.
FIR task: delete the DSP code dependency.
Delete the chapter *Audio gain functions*.
Vocoder Mute: Add some information about this driver.
Add the MMI rules concerning the GPRS.
Add the chapter *compilation flags.*
Add an excel sheet in annexes chapter to sum up the audio task compatibility for the DSP code 32 and 33.
Delete *l1audio_func.c*, *l1audio_ver.h* file.
Add *l1audio_signa.h* file.

3. Correction of some speech recognition L1S state machines that don't be aligned with the principle: state->transition->action.
   Speech recoding during a speech recogntion task: correction due to the problem BUG01200. Two new margins are added (i.e. SC_SR_SPEECH_WORD_BEGIN_MARGIN and SC_SR_SPEECH_WORD_END_MARGIN) in the speech recognition database in order to enhance the pronouncation of the first and the last letters of the recorded word.
   Add a new features for the data communication between the L1 and the Flash manager for the voice memorization and melody(c.f. REQ01191):
   - v Add the chapter L1-Flash manager interface
   - v Change the Melody message and L1A-L1S states machines.
   - v Change the Voice memorization recording/playing messages and state machines.
   - v Add the file l1audio_func.c.
4. Move all L1A and L1S flowcharts and the chapter "structure of L1" to the S917 specification. Add an hyper link in order to reach the L1A and L1S flowcharts and the chapter "structure of L1 written in the specification S917. Correct the compatibility table (c.f. Annex).
5. Add the melody E1 format in the annex. Add some new information concerning the Cust_get_pointer concept. Add the tables about the frequency and the amplitude of the tones and keybeep (c.f. annex keybeep/tones genration). Update messages that contains the key beep and tones definition.
6. Add the message type numbers of the audio messages in the annex section "audio message identifier".
7. Add the audio mode configuration feature.
8. Correction of the FIR example (BUG1703). Add the Analogue Base Band audio drivers and align these drivers with the L1 sofware version: 1362_107_404 (CHG1756).
9. Improvement of the AEC (REQ01998): allow the AEC in all modem mode. Fix the problem in the update-check proces flow: message MMI_SR_RECO_STOP_CON by MMI_SR_UPDATE_CHECK_STOP_CON. Update audio drivers due to Iota (c.f. BUG2027). Integration of the melody format E2. Update the tone, melody E1 and voice memorization record feature due to the change REQ02204.
10. Integration of new AEC
11. Integration of Voice Memo AMR
12. Changed Layer1 RAM buffer format for Voice Memo AMR feature. It intends to be closer to IF1 format defined in [1], in order to ease MMS development (REQ02748). Updated MiditoE1 tools description (CHG02356)
13. Integration of Global Text Telephony

14. Integration of SYREN API + bug correction in new AEC + update Melody E2 + interface change in Speech Reco + BUG3296 (bit 9 of aec_ctrl can be removed)
15. Reference DSP specifications for audio task compatibilities + update “Audio Message identifier” paragraph + deleted chapter 17.
16. Reference DSP specifications for customization of parameters of new AEC + first release into ClearCase.
17. aligned the document with the feature supported in TCS3.2. Add the reference to the ROM38 description.

References:

[1] 3GPP TS 26.101, "AMR Speech Codec Frame Structure", version 4.1.0 (2001-26), 3rd Generation Partnership Project (3GPP)

[2] MMS005 – Midi to E1 tools V.4.2

[3] L1D_GS802, L1D_GS803, L1D_GS804, L1D_GS805, L1D_GS813 – ROM code description

[4] L1_SUP_TM002 (Tuning of new AEC)

Table of content

List of figures

1 INTRODUCTION

The aim of this document is to describe the L1 audio tasks listed below:

- Key beep.
- Tri-tones.
- Voice recording.
- Voice playing.
- Speech recognition (enrollment, update, reco and update-check).
- Melody (format E1).
- Echo canceller.
- FIR configuration.
- Vocoder mute.

For each task, the following item are described:

- Some rules that the MMI must respect to use correctly the task.
- The messages interface between the MMI and the Layer 1.
- The messages from the L1S to the L1A.
- The process flow of the task from the MMI to the DSP.

Note that the MMI rules, described for each audio task, can be used to build easily the MMI state machine corresponding to the audio tasks

2 Key beep

2.1 MMI rules

The MMI must respect the following rules:

- When the MMI starts a keybeep (MMI_KEYBEEP_START_REQ message), it must wait the start confirmation (MMI_KEYBEEP_START_CON) before to stop the keybeep.
- The MMI isn't allowed to start a new keybeep before to receive a stop confirmation (MMI_KEYBEEP_STOP_CON).
- The MMI can receive a stop confirmation (MMI_KEYBEEP_STOP_CON) in two cases:
  - The MMI requested to stop the keybeep (MMI_KEYBEEP_STOP_REQ) and the L1 confirms the stop confirmation message.
  - The keybeep generation is finished and the L1 informs the MMI with the stop confirmation message.
- The MMI is allowed to run the keybeep task during all GSM and GPRS modes.
- The MMI is allowed to run the keybeep task with all another audio tasks.
- For the DSP code 32, during a transition between the GSM and the GPRS, the keybeep task is stopped.

2.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_KEYBEEP_START_REQ	T_MMI_KEYBEEP_REQ
MMI->L1	MMI_KEYBEEP_STOP_REQ	Trigger
MMI<-L1	MMI_KEYBEEP_START_CON	Trigger
MMI<-L1	MMI_KEYBEEP_STOP_CON	Trigger

T_MMI_KEYBEEP_REQ message type

d_k_x1_kt0(UWORD16)
Specifies the frequency (8 most significant bit) and the amplitude (8 less significant bit) of the beep 0. For the value of this parameter see the table 1&2 of the annex (key beep/tones generation).

d_k_x1_kt1(UWORD16)
Specifies the frequency (8 most significant bit) and the amplitude (8 less significant bit) of the beep 1. For the value of this parameter see the table 1&2 of the annex (key beep/tones generation).

d_dur_kb(UWORD16)
Specifies the duration of the key beep (1…32767). This duration corresponds to a number of audio frames (i.e. 20 ms). Note d_dur_kb=0 isn't allowed.

2.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	L1_KEYBEEP_START_CON	Trigger
L1A<-L1S	L1_KEYBEEP_STOP_CON	Trigger

2.4 Process flow

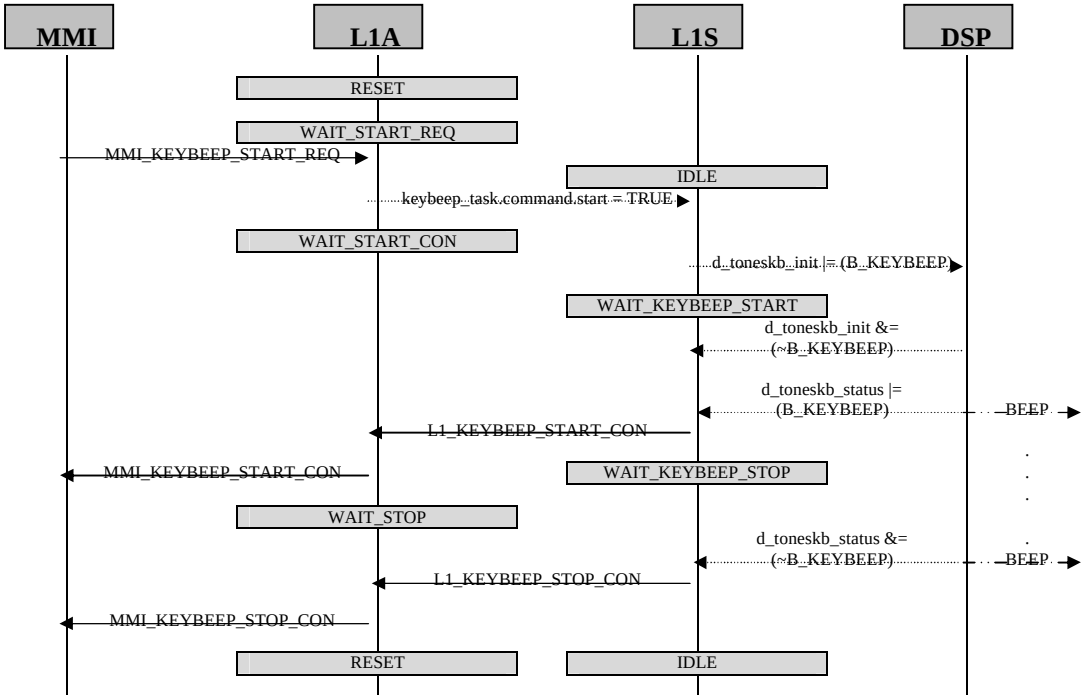


Figure 1: Key beep (Automatic stop)

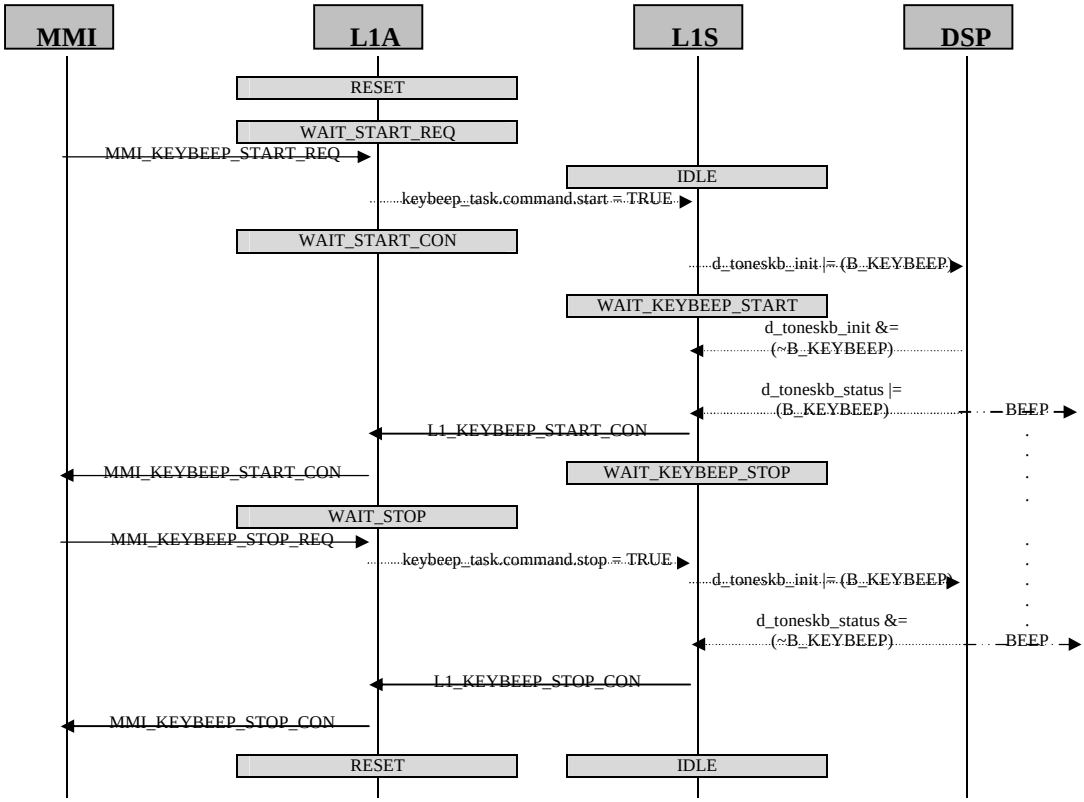


Figure 2: Key beep new solution (Requested stop)

2.5 L1A state machine

Figure 3: L1A keybeep state machine

2.6 L1S state machine

Figure 4: L1S keybeep state machine

3 Tones

3.1 MMI rules

The MMI must respect the following rules:

- When the MMI starts a tone (MMI_TONE_START_REQ message), it must wait the start confirmation (MMI_TONE_START_CON) before to stop the tone.
- The MMI isn't allowed to start a new tone before to receive a stop confirmation (MMI_TONE_STOP_CON).
- The MMI can receive a stop confirmation (MMI_TONE_STOP_CON) in two cases:
  - The MMI requested to stop the tone (MMI_TONE_STOP_REQ) and the L1 confirms with the stop confirmation message.
  - The tone generation is finished and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to run a tone task with:
  - A melody 0.
  - A melody 1.
  - A voice memorization (recording).
- The MMI is allowed to run the tone task during all GSM and GPRS modes.
- For the DSP code 32, during a transition between the GSM and the GPRS, the tone task is stopped.

3.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	**MMI_TONE_START_REQ**	**T_MMI_TONE_REQ**
MMI->L1	**MMI_TONE_STOP_REQ**	**Trigger**
MMI<-L1	**MMI_TONE_START_CON**	**Trigger**
MMI<-L1	**MMI_TONE_STOP_CON**	**Trigger**

T_MMI_TONE_REQ message type
d_k_x1_t0(UWORD16) Specifies the frequency (8 most significant bit) and the amplitude (8 less significant bit) of the tone 0. For the value of this parameter see the table 1&2 of the annex (key beep/tones generation).
d_k_x1_t1(UWORD16) Specifies the frequency (8 most significant bit) and the amplitude (8 less significant bit) of the tone 1. For the value of this parameter see the table 1&2 of the annex (key beep/tones generation).

d_k_x1_t2(UWORD16)
Specifies the frequency (8 most significant bit) and the amplitude (8 less significant bit) of the tone 2. For the value of this parameter see the table 1&2 of the annex (key beep/tones generation).

d_pe_rep(UWORD16)
Number of period to process the tri-tone melody played in dedicated mode (1…0xffff).
Note: d_pe_rep=0 isn't allowed, d_pe_rep=0xffff means the tone is played infinitely.

d_pe_off(UWORD16)
Specifies the duration of the period of the tri-tone melody played in dedicated mode (0…32767). A period consists of a sequence and, if required, a silent phase. This duration corresponds to a number of audio frames (i.e. 20 ms).

d_se_off(UWORD16)
Specifies the duration of the sequence of the tri-tone (0…32767). This duration corresponds to a number of audio frames (i.e. 20 ms). Note that 0<=d_se_off<=d_pe_off.

d_bu_off(UWORD16)
Specifies the duration of a burst of the tri-tone (0…32767). This duration corresponds to a number of audio frames (i.e. 20 ms). Note that 0<=d_bu_off<=d_se_off<=d_pe_off.

d_t0_on(UWORD16)
Specifies the stating time of the tone 0 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 0 must be started. Note that d_t0_on<=d_bu_off.

d_t0_off(UWORD16)
Specifies the ending time of the tone 0 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 0 must be stopped. Note that d_t0_on<=d_t0_off.

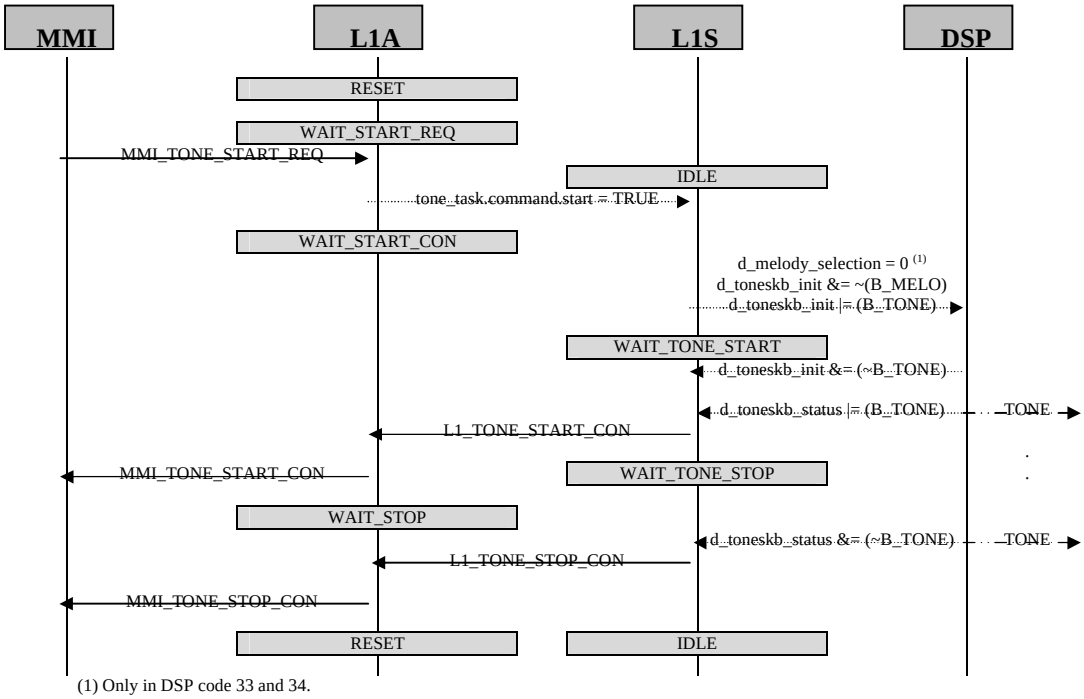
d_t1_on(UWORD16)
Specifies the stating time of the tone 1 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 1 must be started. Note that d_t2_on<=d_bu_off.

d_t1_off(UWORD16) Specifies the ending time of the tone 1 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 1 must be stopped. Note that d_t1_on<=d_t1_off.
d_t2_on(UWORD16) Specifies the stating time of the tone 2 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 2 must be started. Note that d_t2_on<=d_bu_off.
d_t2_off(UWORD16) Specifies the ending time of the tone 2 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 2 must be stopped. Note that d_t2_on<=d_t2_off.

3.3 L1A-L1S message interface

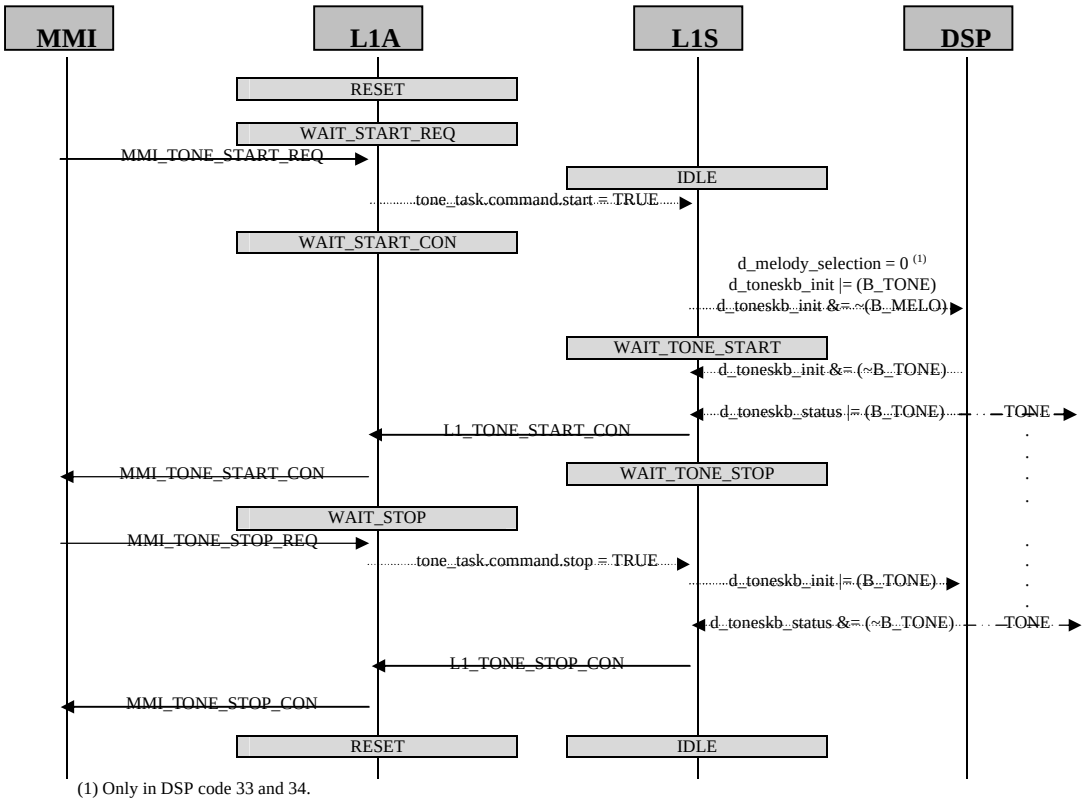
Direction	Message name	Type
L1A<-L1S	**L1_TONE_START_CON**	**Trigger**
L1A<-L1S	**L1_TONE_STOP_CON**	**Trigger**

3.4 Process flow



(1) Only in DSP code 33 and 34.

Figure 5: Tone (Automatic stop)



(1) Only in DSP code 33 and 34.

Figure 6: Tone (Requested stop)

3.5 L1A state machine

Figure 7: L1A tone state machine

3.6 L1S state machine

Figure 8: L1S tone state machine

4 L1-Flash manager interface

The aim of this chapter is to define an interface between the Layer 1 and the Customer Flash manager in order to improve the data exchange between the Flash and RAM memories.
This solution described below is created to correct the following problems:

- The size of the voice memorization was limited due to the RAM memory. Indeed, when the MMI wants to record a voice memo, it allocates a RAM buffer corresponding to the total size of the voice memo. But the size of the RAM is limited compare to the Flash so the Voice memorization is limited compare to the possibility given by the Flash size.
- The L1 can't read directly the data from flash. Indeed, some Flash manager move its data at anytime (i.e. defragmentation of the Flash…), so if the MMI sends an Flash address to L1 to indicate where is the data to read, this address can become wrong.

Moreover, this interface is only developed for the Melody and for the voice memorization recording/playing. For the speech recognition task, the background task is used to communicate between the Flash manager and the L1.

4.1 Process flow

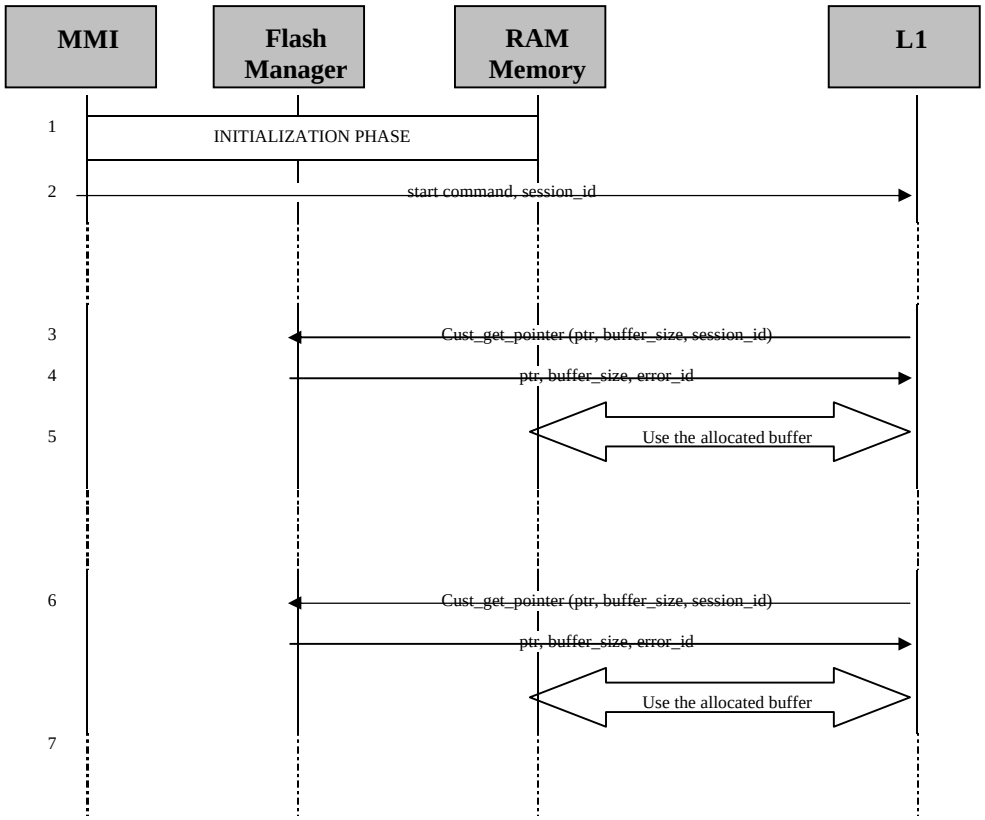


Figure 9: L1-Flash manager communication.

NOTES :

1. Before to start a melody or a voice memorization recording/playing, the MMI runs an initialization phase in order to have some data/space available in a RAM buffer.
2. The MMI task starts the L1 audio task and transmits the **session_id** information. The session_id is used to recognize the type of data exchanged between this new L1 task and the Flash manager (i.e. a melody 0, a melody 1, a voice memorization…). This variable is customer dependent.
3. The L1 task starts. Sometimes, when it needs to have new data or new space in RAM (depends on the task), its calls the Flash manager function **Cust_get_pointer** (c.f. next section for the prototype) with the arguments *ptr*, *buffer_size* and *session_id*. The *buffer_size* corresponds to the size that the L1 needs. The *ptr* is the address corresponding to the last RAM data used (if it exist).
4. The L1-Flash Manager returns to the L1 the information corresponding to the data/space currently available in the RAM memory for this *session_id*. So, it returns the start address of the buffer (pointer **ptr**) and the size (**buffer_size**).
5. Now, the L1 can used the buffer allocated by the Flash Manager.
6. The L1 can call the *Cust_get_pointer* function as mush as it needs data or space for the current *session_id*. Each time the L1 calls this function, the previous buffer (allocated with the previous *Cust_get_pointer* call*)* is no more used by the L1. Therefor, the MMI can used it.
7. When the L1 task is stopped (by MMI or L1), the L1 stops to call this function and sends a stop indication message corresponding to the task to the MMI.

4.2 Cust_get_pointer function

4.2.1 Role

This function is a part of the Flash Manager, it is called by the L1 to request a RAM buffer to download/ save the data corresponding to the session_id and to deallocated the previous RAM buffer.
With this function, the L1 transmits the size of the buffer requested and the address of the last data saved or downloaded.
This functions returns the size of the allocated buffer (more or less than the requested size) and the address of the beginning of the buffer.
Moreover, the first time this function is called by the Layer 1, the pointer argument is NULL since has not been allocated yet.
In a loop back mode (special case), the Layer 1 calls this function with the pointer argument is NULL and the buffer size is 0 when it wants to access to the beginning of the data transmitted by the Flash Manager.

4.2.2 Prototype

UWORD8 **Cust_get_pointer** (*UWORD16* ****ptr**, *UWORD16* ***buffer_size**, *UWORD8* **session_id**)

Note :

1. The address of the next data after the last available data and the buffer size are transmitted by reference because these two values are updated in this function and reuse in the L1 code.
2. If this function contains the arguments ***ptr = NULL, *buffer_size = 0**, it means that it's the first time that the L1 calls this function for this session_id. OR it means that the L1 request the beginning of the data transmitted by the Flash Manager.
3. This function returns an error id (UWORD8):

Error name	Error number	Role
NO_ERROR	0	No error is occurred
SESSION_ERROR	1	Wrong session id
POINTER_ERROR	2	Wrong pointer argument
DATA_AVAIL_ERRO	3	No more data available

4. **This function is called by the L1S therefore the real-time constraint is important, so this function must be small and optimized.**
5. **The current L1 doesn't take into consideration the error above.**

5 Melody format E1

The following melody ringer interface is for the melody format E1. This format is described in the annex. For information, due to the new audio interface described in this specification, the melody format E0 is removed.

5.1 MMI rule

The MMI must respect the following rules (these rules are applied for the melody 0 and 1):

- When the MMI starts a melody (MMI_MELODY0_START_REQ message), it must wait the start confirmation (MMI_MELODY0_START_CON) before to stop the melody 0.
- The MMI isn't allowed to start a new melody before to receive a stop confirmation (MMI_MELODY0_STOP_CON).
- The MMI can receive a stop confirmation (MMI_MELODY0_STOP_CON) in two cases:
  - The MMI requested to stop the melody (MMI_MELO0_STOP_REQ and the L1 confirms the stop confirmation message
  - The melody generation is finished (no loop-back) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed run a melody with:
  - A tone.
  - A voice memorization (recording and playing).
  - A speech recognition (enrollment, update, update-check and recognition).
- The MMI isn't allowed play the format E0 of the melody.
- The MMI is allowed to play the melody 1 and the melody 0 in parallel. Note: these two melodies are played independently.
- The MMI can run the melody in all GSM and GPRS modes.
- For the DSP code 32, during a transition between the GSM and the GPRS, the melody task is stopped.

5.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	**MMI_MELODY0_START_REQ**	**T_MMI_MELODY_REQ**
MMI->L1	**MMI_MELODY1_START_REQ**	**T_MMI_MELODY_REQ**
MMI->L1	**MMI_MELODY0_STOP_REQ**	**Trigger**
MMI->L1	**MMI_MELODY1_STOP_REQ**	**Trigger**
MMI<-L1	**MMI_MELODY0_START_CON**	**Trigger**
MMI<-L1	**MMI_MELODY1_START_CON**	**Trigger**
MMI<-L1	**MMI_MELODY0_STOP_CON**	**Trigger**
MMI<-L1	**MMI_MELODY1_STOP_CON**	**Trigger**

T_MMI_MELODY_REQ message type

session_id (UWORD8)
Specifies the customer data identification corresponding to this melody (for L1-Flash manager communication).

loopback (BOOL)
Specifies if the melody is played in loopback mode (loopback = TRUE) or not (loopback = FALSE).
Note: if the loop back mode is enable, the melody is restarted when all dedicated oscillators are stopped.

oscillator_used_bitmap (UWORD16)

This bit field register indicates which oscillators must be used to play the melody:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Oscillator	xx	xx	xx	xx	xx	xx	xx	xx	7	6	5	4	3	2	1	0

Note : “xx” means that this bit isn’t used, IT MUST BE PUT TO 0.

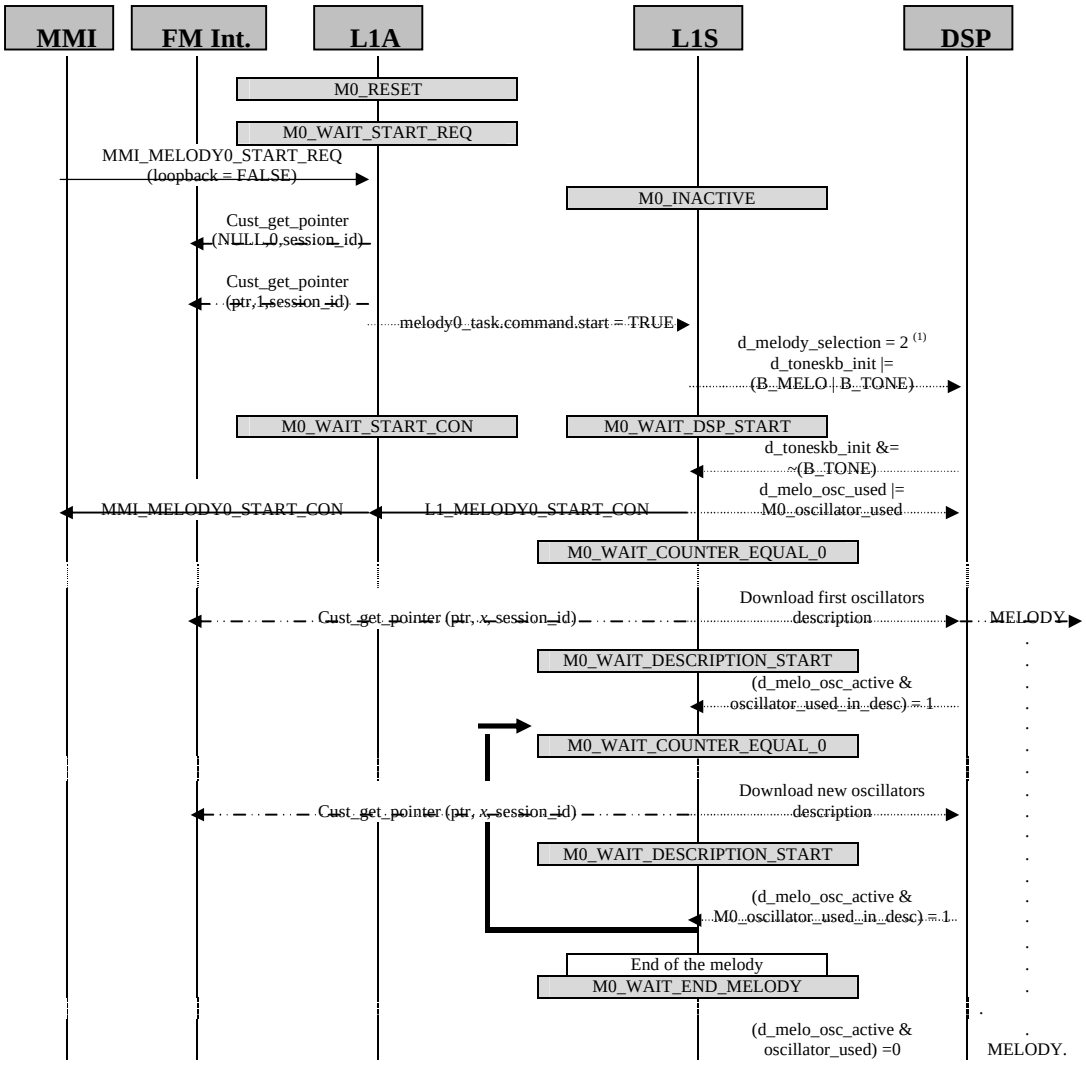
An oscillator is used if the corresponding bit is set to 1 otherwise it must be set to 0.

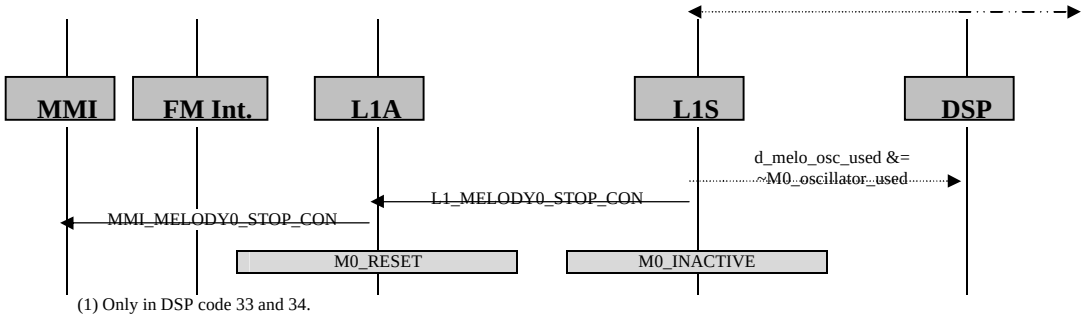
5.3 L1A-L1S interface

Direction	Message name	Type
L1A<-L1S	**L1_MELODY0_START_CON**	**Trigger**
L1A<-L1S	**L1_MELODY1_START_CON**	**Trigger**
L1A<-L1S	**L1_MELODY0_STOP_CON**	**Trigger**
L1A<-L1S	**L1_MELODY1_STOP_CON**	**Trigger**

5.4 Process flow

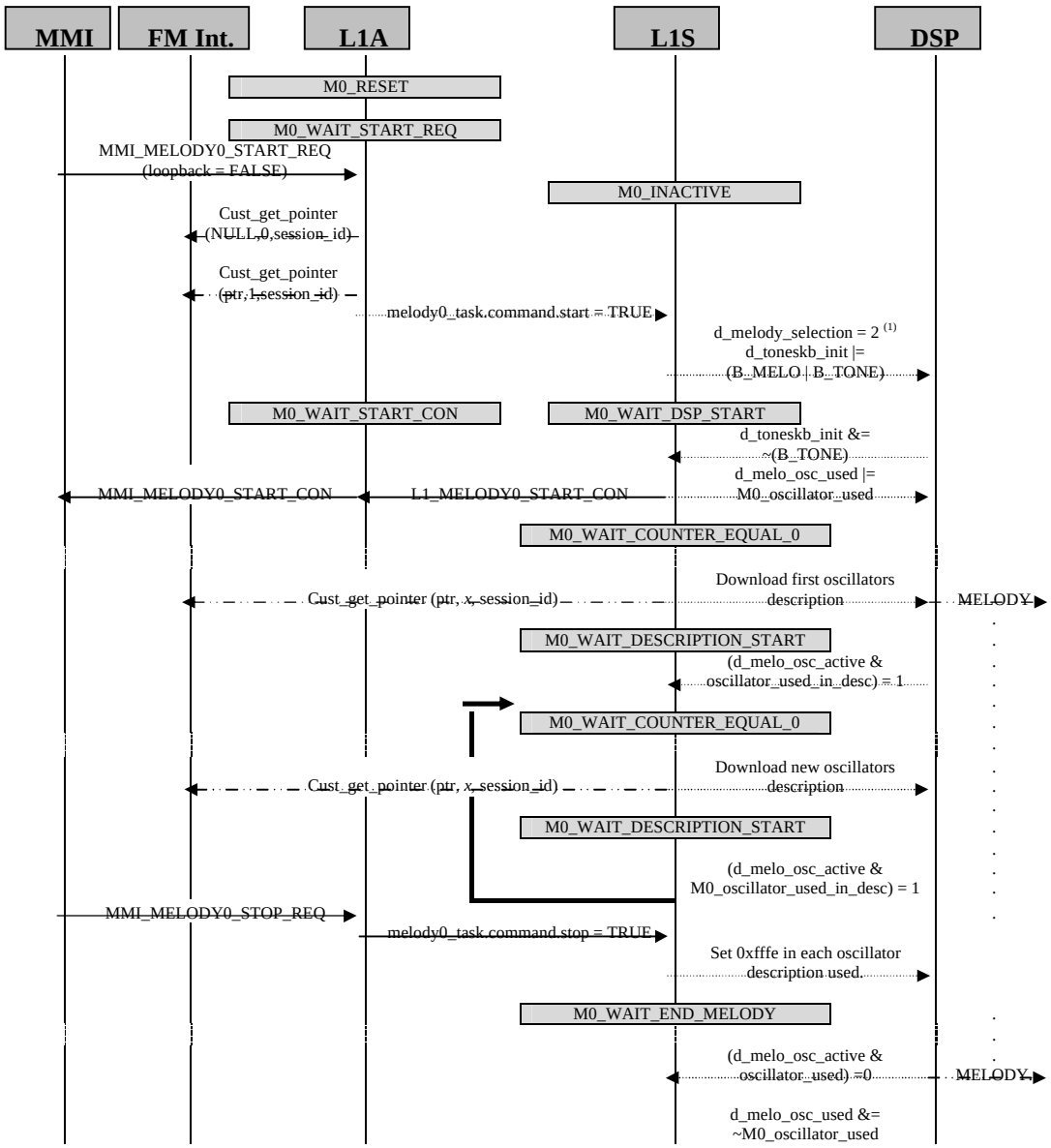
The following descriptions are for the melody 0. The same behavior could be described for the melody 1.
Note: FM Int. means the Flash Manager Interface.

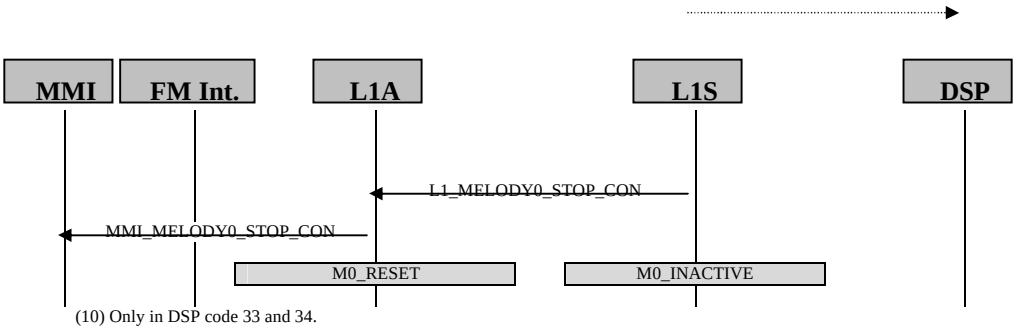




(1) Only in DSP code 33 and 34.

Figure 10: Melody E1 (automatic stop)





(10) Only in DSP code 33 and 34.

Figure 11: Melody E1 (requested stop)

5.5 L1A state machine

5.6 L1S state machine

6 Melody format E2

The following melody ringer interface is for the melody format E2.

6.1 MMI rule

The MMI must respect the following rules to play the melody format E2 (these rules can be applied for melody 1 too):

- When the MMI starts a melody 0 (MMI_MELODY0_E2_START_REQ message), it must wait for the start confirmation (MMI_MELODY0_E2_START_CON) before stopping this melody.
- The MMI isn't allowed to start a new melody 0 before to receive a stop confirmation (MMI_MELODY0_E2_STOP_CON).
- The MMI can receive a stop confirmation (MMI_MELODY0_E2_STOP_CON) in two cases:
  - The MMI requested to stop the melody (MMI_MELODY0_E2_STOP_REQ) and the L1 confirms the stop confirmation message.
  - The melody generation is finished (no loop-back) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to play a melody format E1 in the same time as the E2.
- The MMI is allowed to play the melody 1 and the melody 0 in parallel. Note: these two melodies are played independently. **But the MMI needs to take care that the 16 oscillators are shared between the 2 melodies. Therefore, the sum of the maximum number of note of each melody must be less or equal than 16.**
- **The sum of the size of the instruments used to play melodies must be less than SC_AUDIO_MELODY_E2_MAX_SIZE_OF_INSTRUMENT== 3800 16-bits word (c.f. l1audio_cust.h file).**

Locosto TCS3.2 doesn't support this feature.

6.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_MELODY0_E2_START_REQ	T_MMI_MELODY_E2_REQ
MMI<-L1	MMI_MELODY0_E2_START_CON	Trigger
MMI->L1	MMI_MELODY0_E2_STOP_REQ	Trigger
MMI<-L1	MMI_MELODY0_E2_STOP_CON	Trigger
MMI->L1	MMI_MELODY1_E2_START_REQ	T_MMI_MELODY_E2_REQ
MMI<-L1	MMI_MELODY1_E2_START_CON	Trigger
MMI->L1	MMI_MELODY1_E2_STOP_REQ	Trigger
MMI<-L1	MMI_MELODY1_E2_STOP_CON	Trigger

T_MMI_MELODY_E2_REQ message type

session_id (UWORD8)
Specifies the customer data identification corresponding to this melody (for L1-Flash manager communication (c.f. chapter L1-Flash manager interface)).

loopback (BOOL)
Specifies if the melody is played in loopback mode (loopback = TRUE) or not (loopback = FALSE).
Note: if the loop back mode is enabling, the melody is restarted when all the dedicated oscillators are stopped.

6.3 L1A-L1S message interface

Direction	Message name	Type
L1S->L1A	L1_MELODY0_E2_STOP_CON	Trigger
L1S->L1A	L1_MELODY1_E2_STOP_CON	Trigger

6.4 Audio Background-L1A message interface

Direction	Message name	Type
L1A->Audio Back	L1_BACK_MELODY_E2_LOAD_INSTRUMENT_REQ	T_L1_BACK_MELODY_E2_LOAD_INSTRUMENT_REQ
L1A->Audio Back	L1_BACK_MELODY_E2_UNLOAD_INSTRUMENT_REQ	T_L1_BACK_MELODY_E2_UNLOAD_INSTRUMENT_REQ
Audio Back->L1A	L1_BACK_MELODY_E2_LOAD_INSTRUMENT_CON	T_L1_BACK_MELODY_E2_LOAD_INSTRUMENT_CON
Audio Back->L1A	L1_BACK_MELODY_E2_UNLOAD_INSTRUMENT_CON	T_L1_BACK_MELODY_E2_UNLOAD_INSTRUMENT_CON

T_L1_BACK_MELODY_E2_LOAD_INSTRUMENT_REQ message type

melody_id (UWORD8)
Specifies which melody wants to load some instruments: melody0 (melody_id=0) or melody1 (melody_id=1).

number_of_instrument (UWORD8)
Specifies the number of instrument to download. Range : 0… SC_AUDIO_MELODY_E2_MAX_NUMBER_OF_INSTRUMENT=8 (c.f. l1audio_cust.h).

waves_table_id[SC_AUDIO_MELODY_E2_MAX_NUMBER_OF_INSTRUMENT] (UWORD8)
Table of the identifier of the instrument to download. Note this instrument ID comes from the header of the melody E2 and correspond to a *.mwa* file thanks to the ID to name file converter *defaut.lsi* (c.f. HER815).

T_L1_BACK_MELODY_E2_LOAD_INSTRUMENT_CON message type

melody_id (UWORD8)
Specifies which melody to confirm the load of some instruments: melody0 (melody_id=0) or melody1 (melody_id=1).

T_L1_BACK_MELODY_E2_UNLOAD_INSTRUMENT_REQ message type
melody_id (UWORD8) Specifies which melody wants to unload some instruments: melody0 (melody_id=0) or melody1 (melody_id=1).
number_of_instrument (UWORD8) Specifies the number of instrument to unload. Range : 0… SC_AUDIO_MELODY_E2_MAX_NUMBER_OF_INSTRUMENT=8 (c.f. l1audio_cust.h).

T_L1_BACK_MELODY_E2_UNLOAD_INSTRUMENT_CON message type
melody_id (UWORD8) Specifies which melody to confirm the unload of some instruments: melody0 (melody_id=0) or melody1 (melody_id=1).

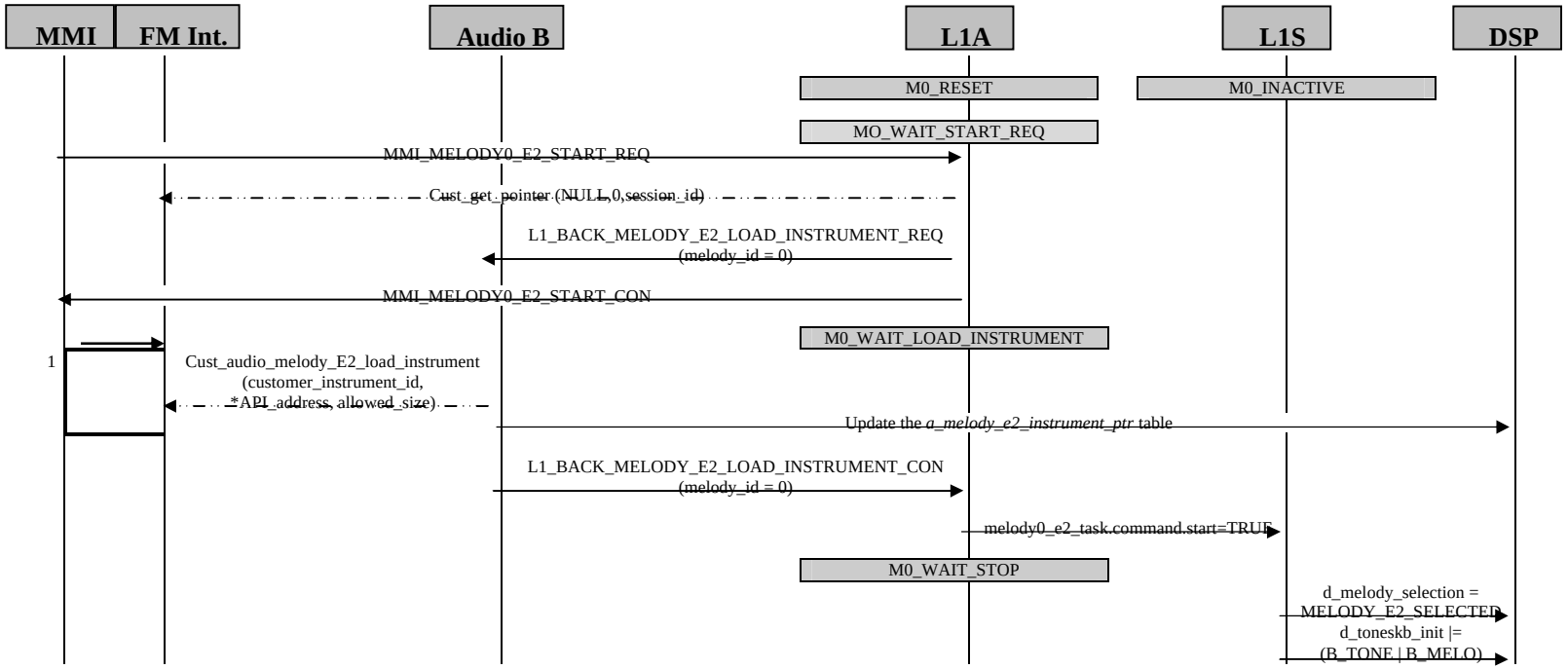
6.5 L1S-DSP interface

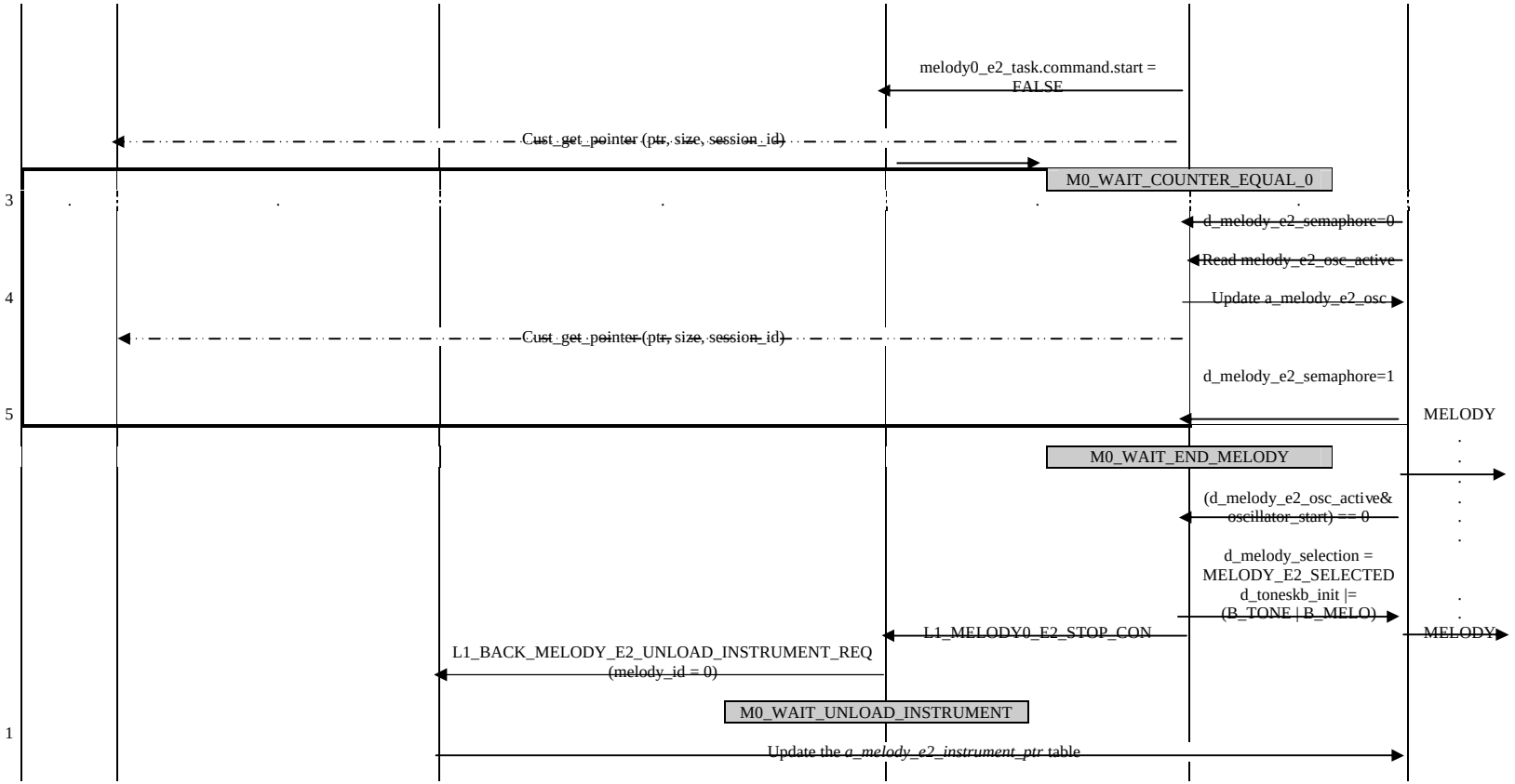
Name	Size (16 bits word)	DSP API address		Description
d_melody_e2_osc_stop	1	0x1500		This bits field indicates to the DSP which oscillator must be stopped immediately
d_melody_e2_osc_active	1	0x1501		This bits field indicates to the MCU which oscillator is active
d_melody_e2_semaphore	1	0x1502		This semaphore is used to validate the new MCU oscillators request.
a_melody_e2_osc0… a_melody_e2_osc15	3*16	a_melody_e2_osc0	0x1503	These 16 tables contain the configuration of the 16 oscillators.
		a_melody_e2_osc1	0x1506	
		a_melody_e2_osc2	0x1509	
		a_melody_e2_osc3	0x150C	
		a_melody_e2_osc4	0x150F	
		a_melody_e2_osc5	0x1512	
		a_melody_e2_osc6	0x1515	
		a_melody_e2_osc7	0x1518	
		a_melody_e2_osc8	0x151B	
		a_melody_ e2_osc9	0x151E	
		a_melody_e2_osc10	0x1521	
		a_melody_e2_osc11	0x1524	
		a_melody_e2_osc12	0x1527	
		a_melody_e2_osc13	0x152A	
		a_melody_e2_osc14	0x152D	
		a_melody_e2_osc15	0x1530	
d_melody_e2_globaltimefactor	1	0x1533		This word contains the GlobalTimeFactor value mentioned in the header area of the E2 file format (c.f. [1])
a_melody_e2_instrument_ptr	8	0x1534..0x153B		This table contains the 8 API DSP addresses that indicate the beginning of the *.mwa* files put in the *a_melody_e2_instrument_wave* area.
a_melody_e2_deltatime	1	0x153C		Delta time to next note in 20ms unit. DSP decrements it and stops playing when it reaches 0.
a_melody_e2_instrument_wave	3800	0x1807…0x26DE		This table contains the waves samples of the instruments used for the melody E2. In fact this memory area contains the *.mwa* file of the instruments (c.f. [1])

So, the E2 feature needs 3861 **16-bits words** in the API memory to work correctly.

6.5.1 Process flow

The following descriptions are for the melody 0. The same behavior could be described for the melody 1.
Note: FM Int. means the Flash Manager Interface and Audio B. means the audio background task.





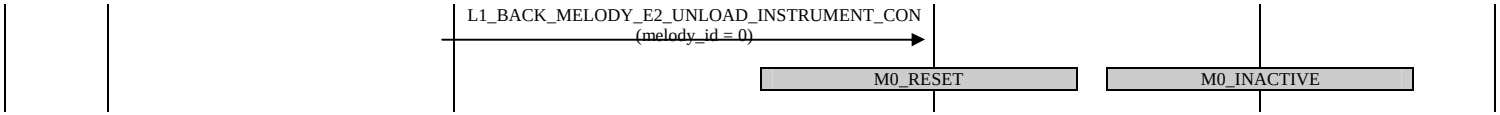
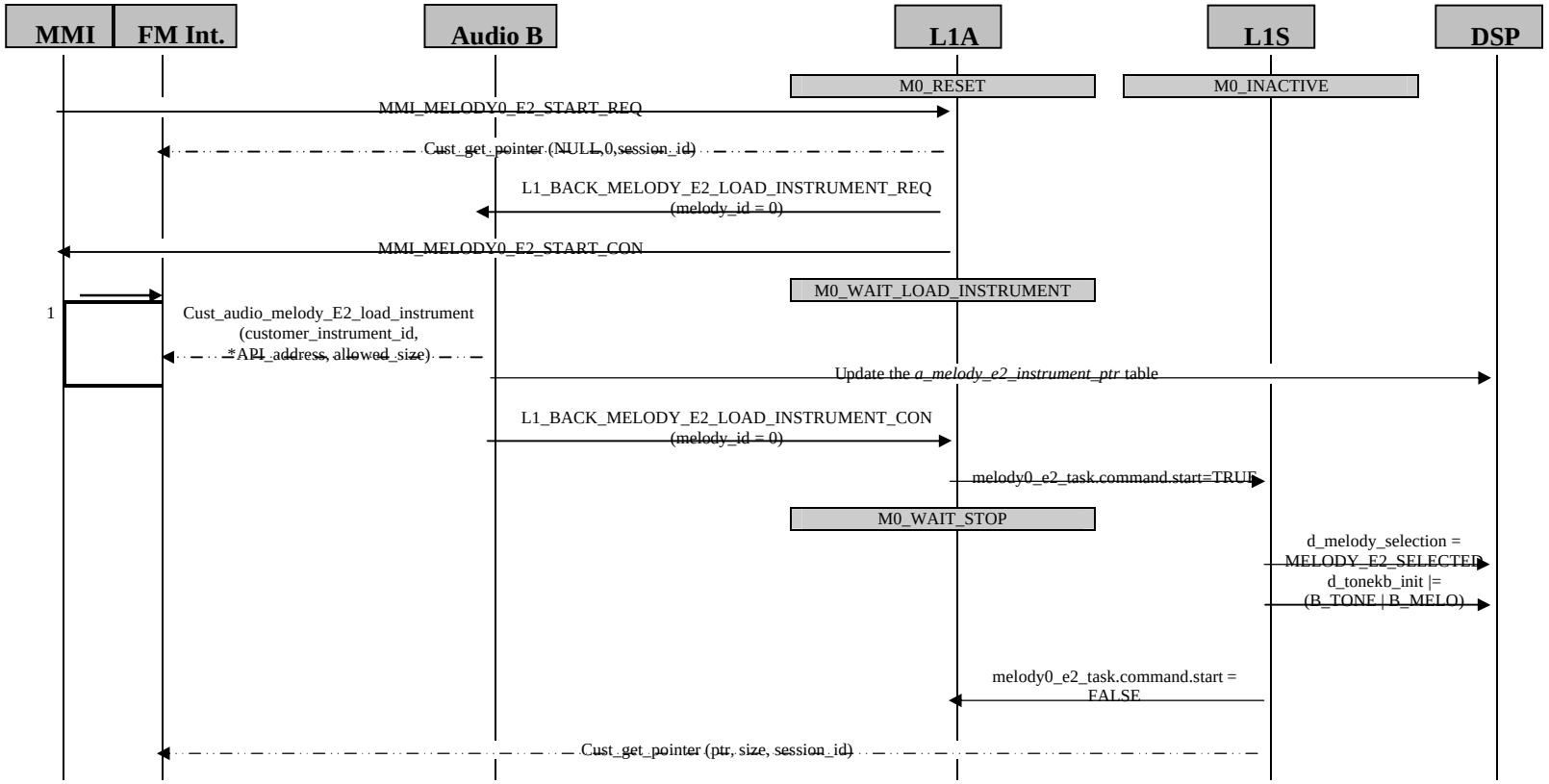
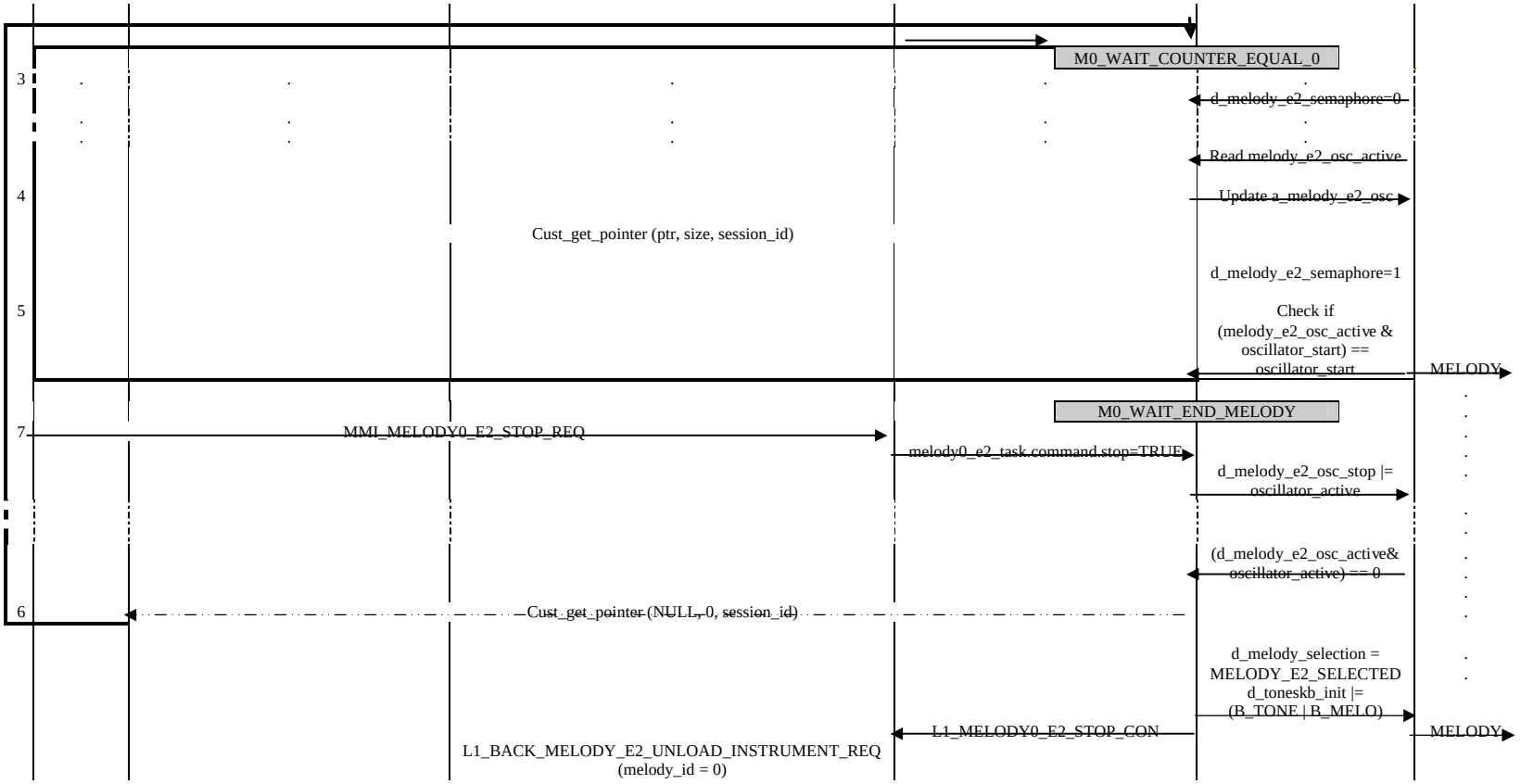


Figure 15: Melody E2 (automatic stop with no loop back)

Note :

1. The function *Cust_audio_melody_E2_load_instrument* is called for each instrument to download the corresponding instrument description to the API memory (c.f. below for the description of this function). If an instrument is already downloaded, this function isn't call again for this instrument.
2. The function *Cust_get_pointer* can be called every time during the L1S melody E2 task. It depends on the size of the data given by customer flash manager in each call.
3. The L1S waits until it's time to download the first note description and until the semaphore is set to 0 by the DSP.
4. L1S skips the description of the notes that can't be loaded (more than 16 oscillators already running). This behavior avoids the crash of the L1 in case of the number of oscillator available for the melody isn't enough.
5. If it is not the end of the melody the L1S returns to the state M0_WAIT_COUNTER_EQUAL_0, otherwise it goes to the state M0_WAIT_END_MELODY where it waits until the last note of the melody 0 was played.





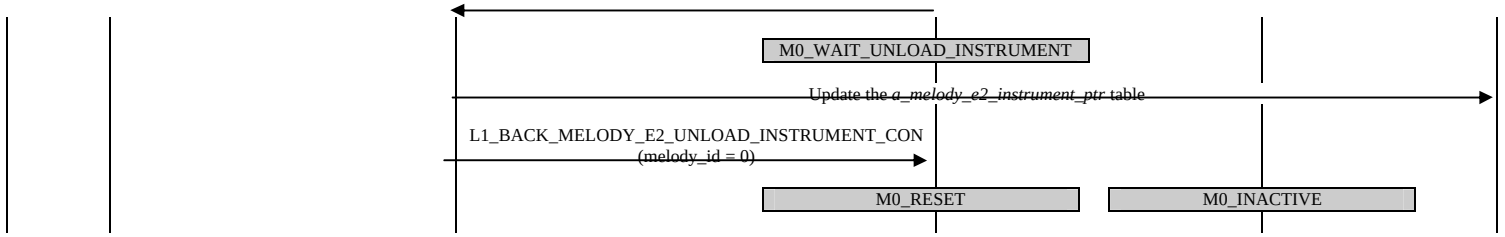


Figure 16: Melody E2 (requested stop with loop back)

Note :

1. The function *Cust_audio_melody_E2_load_instrument* is called for each instrument to download the corresponding instrument description to the API memory (c.f. below for the description of this function). If an instrument is already downloaded, this function isn't call again for this instrument.
2. The function *Cust_get_pointer* can be called every time during the L1S melody E2 task. It depends on the size of the data given by customer flash manager in each call.
3. The L1S waits until it's time to download the first note description and until the semaphore is set to 0 by the DSP.
4. L1S skips the description of the notes that can't be loaded (more than 16 oscillators already running). This behavior avoids the crash of the L1 in case of the number of oscillator available for the melody isn't enough.
5. If it is not the end of the melody the L1S returns to the state M0_WAIT_COUNTER_EQUAL_0, otherwise it goes to the state M0_WAIT_END_MELODY where it waits until the last note of the melody 0 was played.
6. In case of loop back mode, the function *Cust_get_pointer* is called with the parameter *NULL* and *0*. After, the L1S jumps to the state M0_WAIT_COUNTER_EQUAL_0.
7. If a stop message is received from the MMI, the L1S stops immediately the note belonging to the melody 0 and returns the stop confirmation.

6.6 L1A state machine

Figure 17: L1A melody E2 state machine

6.7 L1S state machine

Figure 18: L1S melody E2 state machine (1/2)

Figure 19: L1S melody E2 state machine (2/2)

6.8 Cust_audio_melody_E2_load_instrument

UWORD16 Cust_audio_melody_E2_load_instrument (customer_instrument_id,
***API_address,**
allowed_size)

The audio background task calls this function to request to the MMI to download a new instrument description (.mwa file) to the API memory pointed by API_address.

This function must return the size of the downloaded instrument in 16-bits word unit. If the instrument doesn't exist or can't be downloaded because the size allowed (allowed_size) is less than the size of the instrument, this function must return 0.

This function must immediately stop when an emergency stop is requested by L1 (i.e. the value l1a_l1s_com.melody1_e2_task.parameters.emergency_stop is set to TRUE)

Customer_instrument_id (UWORD8)
Specifies the identifier of the instrument to download. This identifier comes form the header of the melody E2.

API_address (*API)
Specifies the API memory address where the index of the model into the user database.

Allowed_size (UWORD16)
Specifies the size allowed to download the instrument. This size is in 16-bits words unit.

7 Voice memorisation

7.1 Principle

The voice memorization feature is a task to record and play some speech samples pronounced by the mobile user. So this task can be divided in two subtasks:

- Recording task: task to save the samples corresponding the to the user speech.
- Playing task: task to play the speech samples previously recorded.

The compression format used is based on the full rate vocoder. So, the rate is 20 word (16 bits) per 20 ms. So to record 1 min. of speech, the MMI must allowed 60s/0.02s * 20 = 60 000word = 58,6Kwords. Moreover if the DTX use, the rate changes if the user speak or not. If it's doesn't speak, only the background noise is recorded so the rate if 1 word per 20ms. And if the use pronounces something the rate is 20 word per 20 ms.

7.2 Playing task

7.2.1 MMI rules

The MMI must respect the following rules

- When the MMI starts a voice memorization playing task (MMI_VM_PLAY_START_REQ message), it must wait the start confirmation (MMI_VM_PLAY_START_CON) before to stop this task.
- The MMI isn't allowed to start a new voice memorization playing task before to receive a stop confirmation message (MMI_VM_PLAY_STOP_CON).
- The MMI can receive a stop confirmation message (MMI_VM_PLAY_STOP_CON) in two cases:
  - The MMI requested to stop the current voice memorization playing task (i.e. MMI_VM_PLAY_STOP_REQ) and the L1 confirms with the stop confirmation message.
  - The current task is stopped automatically (end of the voice buffer) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to run a voice memorization playing task with:
  - A melody 0.
  - A melody 1.
  - A voice memorization recording.
  - A speech recognition (enrollment, update, update-check, recognition).
- The MMI is allowed to run the voice memorization playing task in all GSM modes excepted with the IDS module at 9600bps in non-transparent mode.
- For the GPRS, the MMI is allowed to run in all modes.
- For the DSP code 32, during a transition between the GSM and the GPRS, the voice memorization playing task is stopped.

7.2.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_VM_PLAY_START_REQ	T_MMI_VM_PLAY_REQ
MMI->L1	MMI_VM_PLAY_STOP_REQ	Trigger
MMI<-L1	MMI_VM_PLAY_START_CON	Trigger
MMI<-L1	MMI_VM_PLAY_STOP_CON	Trigger

T_MMI_VM_PLAY_REQ message type

session_id (UWORD8)
Specifies the customer data identification corresponding to this melody (for L1-Flash manager communication).

7.2.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	L1_VM_PLAY_START_CON	Trigger
L1A<-L1S	L1_VM_PLAY_STOP_CON	Trigger

7.2.4 Process flow

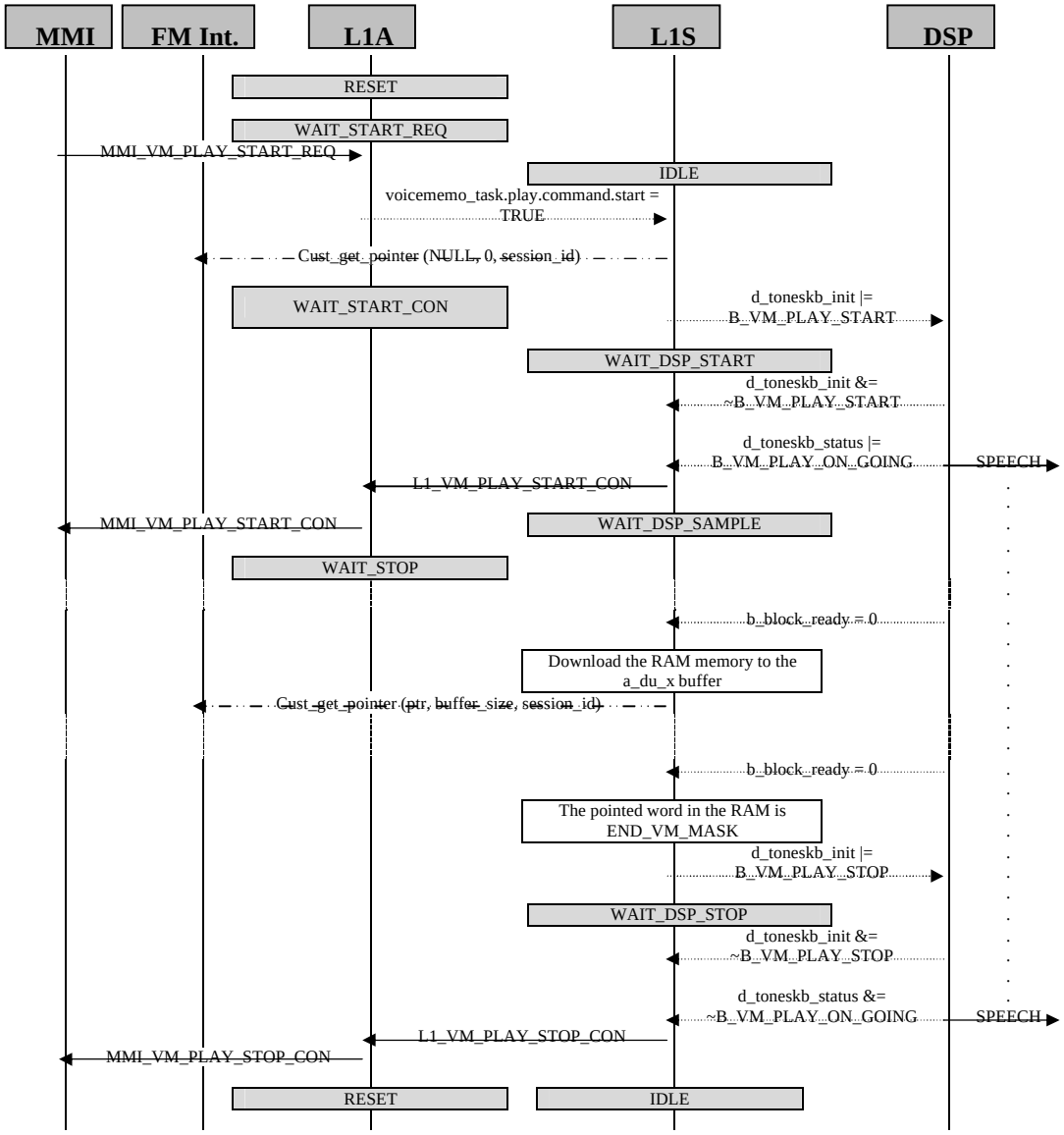


Figure 20: Voice memo (playing in all mode with an automatic stop)

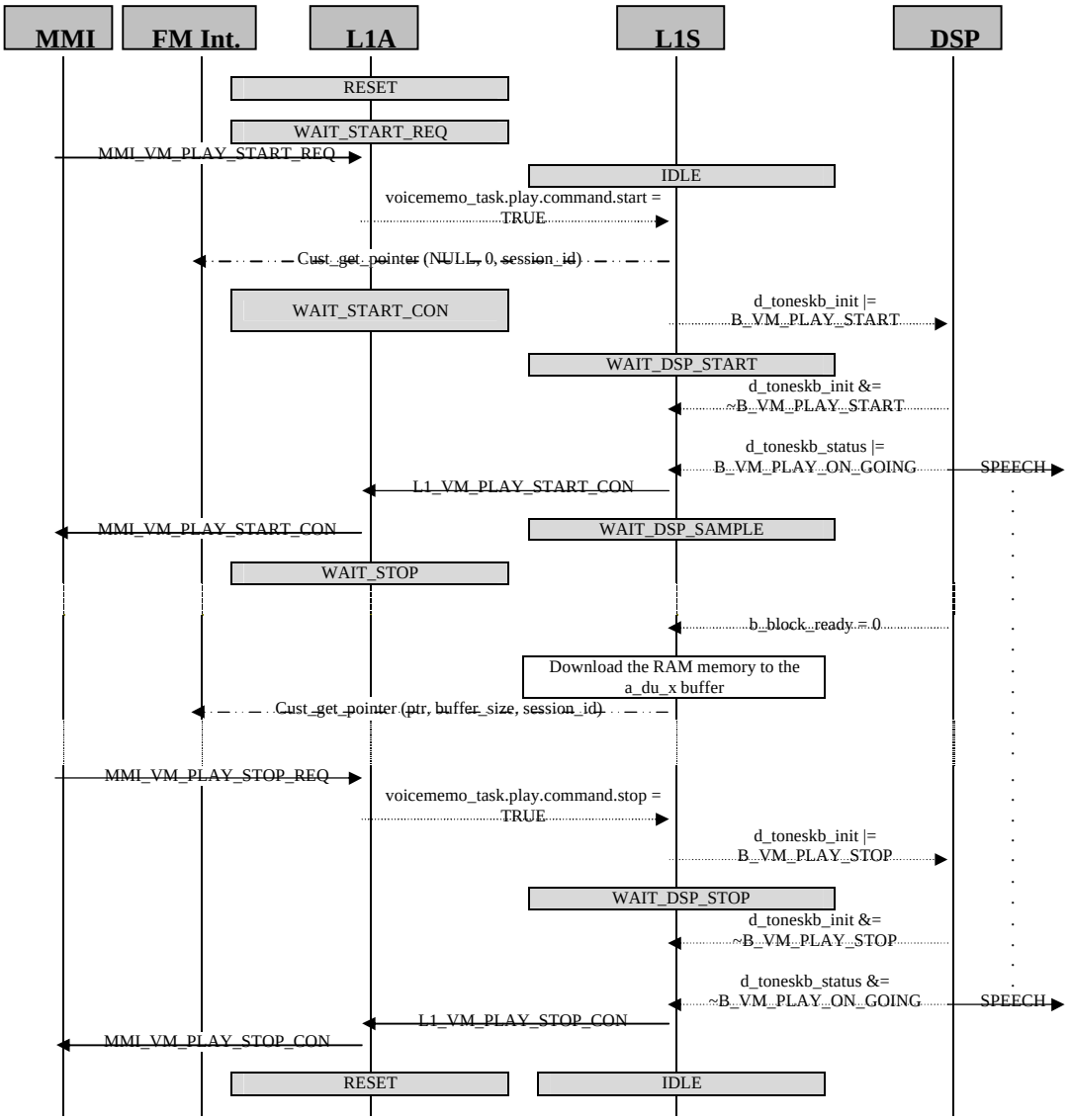


Figure 21: Voice memo (playing in all mode with a requested stop)

7.2.5 L1A state machine

Figure 22: Voice memo playing (L1A state machine)

7.2.6 L1S state machine

Figure 23: Voice memo (L1S voice memo playing state machine)

7.3 Recording task

7.3.1 MMI rules

The MMI must respect the following rules

- When the MMI starts a voice memorization recording task (MMI_VM_RECORD_REQ message), it must wait the start confirmation (MMI_VM_RECORD_START_CON) before to stop this task.
- The MMI isn't allowed to start a new voice memorization task before to receive a stop confirmation message (MMI_VM_RECORD_STOP_CON).
- The MMI can receive a stop confirmation message (MMI_VM_RECORD_STOP_CON) in two cases:
  - The MMI requested to stop the current voice memorization recording task (i.e. MMI_VM_RECORD_STOP_REQ) and the L1 confirms with the stop confirmation message.
  - The recording task is stopped automatically (end of the voice buffer) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to run a voice memorization recording task with:
  - A melody 0.
  - A melody 1.
  - A tone.
  - A speech recognition (enrollment, update, update-check, recognition).
  - A voice memorization playing.
- The MMI is allowed to run the voice memorization recording task in all GSM and GPRS modes.
- For the DSP code 32, during a transition between the GSM and the GPRS, the voice memorization recording task is stopped.

Moreover, a tone (c.f. parameters of the MMI_VM_RECORD_REQ message) is generated automatically in the UL and DL path when the voice memorization recording task run during the dedicated speech mode (c.f. the section below "Tone uplink").

7.3.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_VM_RECORD_START_REQ	T_MMI_VM_RECORD_REQ
MMI->L1	MMI_VM_RECORD_STOP_REQ	Trigger
MMI<-L1	MMI_VM_RECORD_START_CON	Trigger
MMI<-L1	MMI_VM_RECORD_STOP_CON	T_MMI_VM_RECORD_CON

T_MMI_VM_RECORD_CON message type

recorded_size(UWORD32)
Specifies the size in 16-bit word unit of the recorded data..
Note: This size takes into account the end mask put at the end of the recorded data (i.e. SC_VM_END_MASK).

T_MMI_VM_RECORD_REQ message type

session_id (UWORD8)
Specifies the customer data identification corresponding to this melody (for L1-Flash manager communication).

maximum_size(UWORD32)
Specifies the maximum size (in 16-bit word unit) of the recorded data. This value takes into account the end mask put at the end of the recorded data (i.e. SC_VM_END_MASK).
Therefor, if the recorded duration is 1 min (without DTX, rate 1 16-bit word/1ms), the size is 60s * 1000 +1 = 117,2 Kbytes

dtx_used(BOOL)
Specifies the DTX mode for the voice memo record (0 = No DTX, 1 = DTX)

record_coeff_ul (UWORD16)
Coefficient added during the voice memo recording to the uplink TCH path in format Q8.8.
Example: record_coef_ul = 0x0100 = 1, record_coef_dl = 0x0080 = 0,5.

record_coeff_dl (UWORD16)
Coefficient added during the voice memo recording to the downlink TCH path in format Q8.8.
Example: record_coef_dl = 0x0100 = 1, record_coef_ul = 0x0080 = 0,5.

d_k_x1_t0(UWORD16)
Specifies the frequency (8 most significant bit) and the amplitude (8 less significant bit) of the tone 0. For the value of this parameter see the table 1&2 of the annex (key beep/tones generation).

d_k_x1_t1(UWORD16)
Specifies the frequency (8 most significant bit) and the amplitude (8 less significant bit) of the tone 1. For the value of this parameter see the table 1&2 of the annex (key beep/tones generation).

d_k_x1_t2(UWORD16)
Specifies the frequency (8 most significant bit) and the amplitude (8 less significant bit) of the tone 2. For the value of this parameter see the table 1&2 of the annex (key beep/tones generation).

d_pe_rep(UWORD16)
Number of period to process the tri-tone melody played in dedicated mode (0…0xffff).
If d_pe_rep=0 no tri-tone is played.

d_pe_off(UWORD16)
Specifies the duration of the period of the tri-tone melody played in dedicated mode (0…32767). A period consists of a sequence and, if required, a silent phase. This duration corresponds to a number of audio frames (i.e. 20 ms).

d_se_off(UWORD16)
Specifies the duration of the sequence of the tri-tone (0…32767). This duration corresponds to a number of audio frames (i.e. 20 ms). Note that 0<=d_se_off<=d_pe_off.

d_bu_off(UWORD16)
Specifies the duration of a burst of the tri-tone (0…32767). This duration corresponds to a number of audio frames (i.e. 20 ms). Note that 0<=d_bu_off<=d_se_off<=d_pe_off.

d_t0_on(UWORD16)
Specifies the stating time of the tone 0 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 0 must be started. Note that d_t0_on<=d_bu_off.

d_t0_off(UWORD16)
Specifies the ending time of the tone 0 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 0 must be stopped. Note that d_t0_on<=d_t0_off.

d_t1_on(UWORD16)
Specifies the stating time of the tone 1 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 1 must be started. Note that d_t2_on<=d_bu_off.

d_t1_off(UWORD16)
Specifies the ending time of the tone 1 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 1 must be stopped. Note that d_t1_on<=d_t1_off.

d_t2_on(UWORD16)
Specifies the stating time of the tone 2 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 2 must be started. Note that d_t2_on<=d_bu_off.

d_t2_off(UWORD16)
Specifies the ending time of the tone 2 (0…32767). Considering the length of a burst period, in term of number of audio frame (i.e. 20 ms), this values describes when the tone 2 must be stopped. Note that d_t2_on<=d_t2_off.

7.3.3 L1A-L1S message interface

Direction	Message name	Type
L1A->L1S	**L1_VM_RECORD_START_CON**	**Trigger**
L1A->L1S	**L1_VM_RECORD_STOP_CON**	**T_L1_VM_RECORD_CON**

T_L1_VM_RECORD_CON message type

recorded_size(UWORD32)
Specifies the size in 16-bit word unit of the recorded data..
Note: This size takes into account the end mask put at the end of the recorded data (i.e. SC_VM_END_MASK).

7.3.4 Process flow

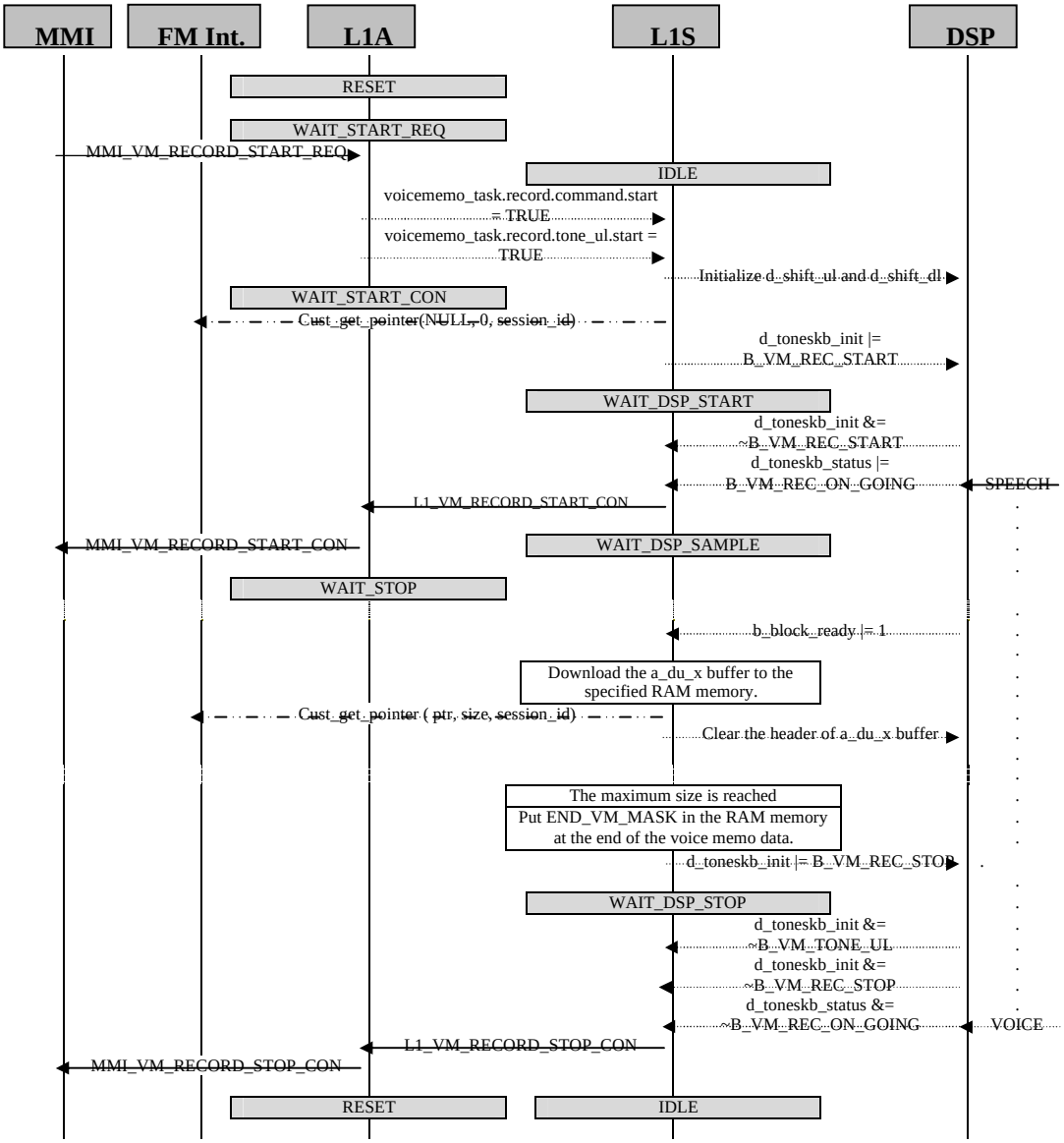
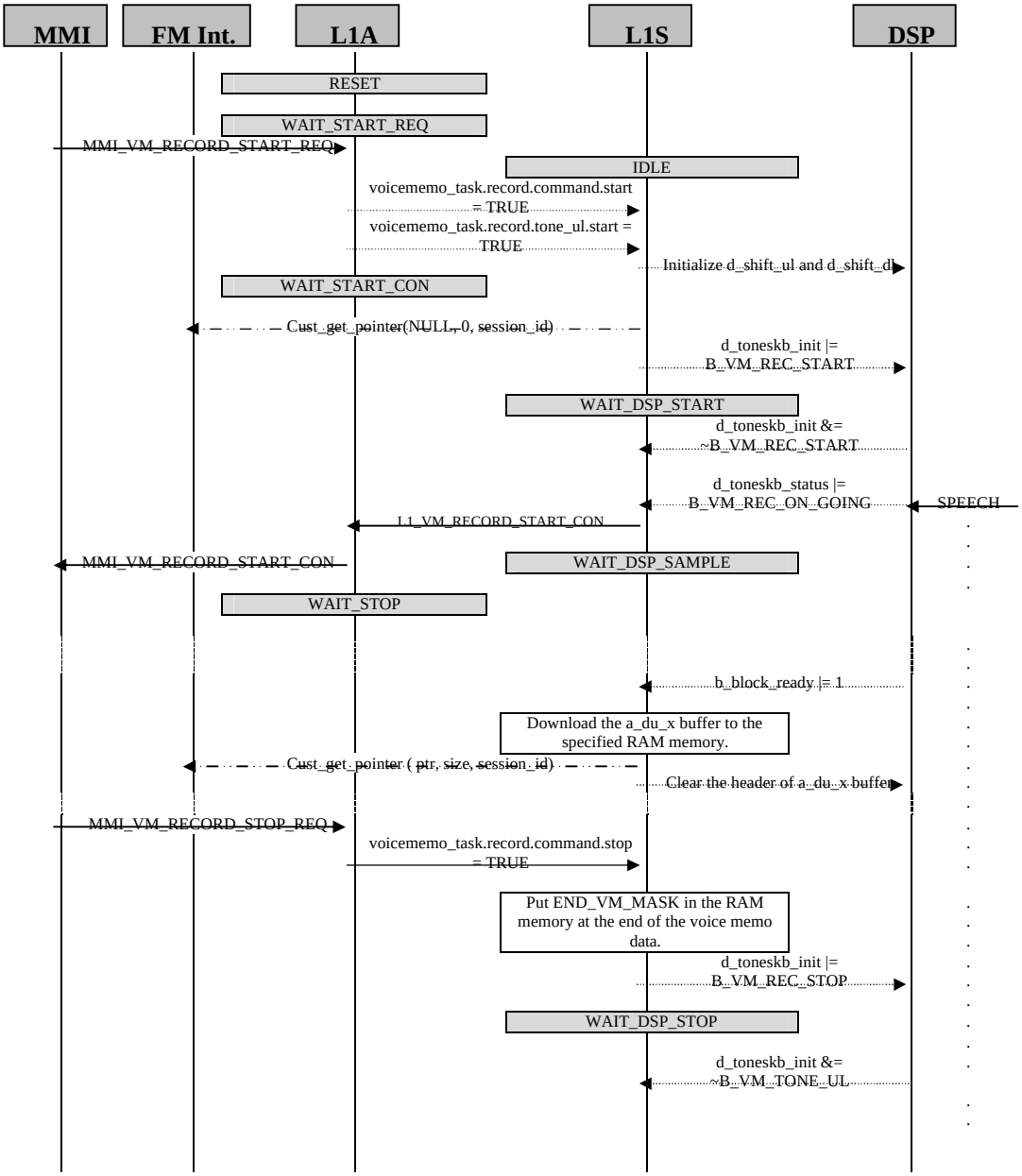


Figure 24: Voice memo (recording with an automatic stop)



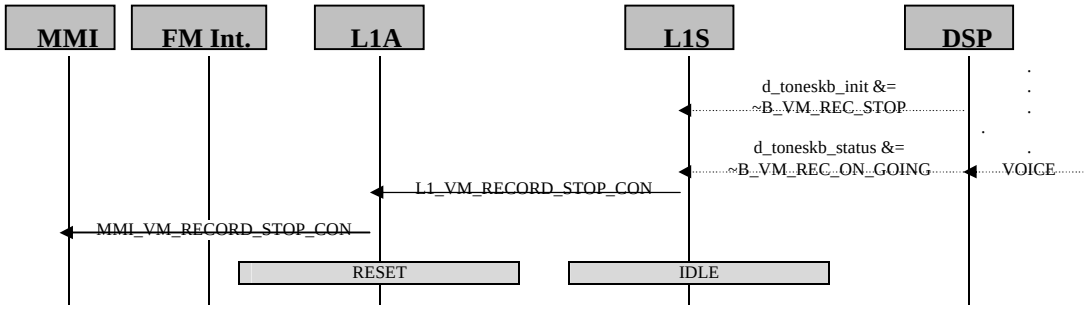


Figure 25: Voice memo (recording with a requested stop)

7.3.5 L1A state machine

Figure 26: Voice memo recording (L1A state machine)

7.3.6 L1S state machine

Figure 27: Voice memo (L1S voice memo recording state machine)

7.4 Tone uplink

7.4.1 Definition

The tone uplink task is a tone generated in the UL and DL audio path during the voice memorization recording task and only in dedicated speech mode. The parameters of these tones are included in the voice memorization recording start message (i.e. MMI_VM_RECORD_START_REQ).
Finally, the tone uplink task is started when the voice recording task starts (i.e. the L1A set voicememo_task.record.tone_ul.start to TRUE) and the tone is generated only if the L1S is in dedicated mode (i.e. l1a_l1s_com.dedic_set.aset <> NULL and l1a_l1s_com.dedic_set.aset->achan_ptr->mode = TCH_FS_MODE or TCH_HS_MODE or TCH_EFR_MODE).

7.4.2 Process Flow

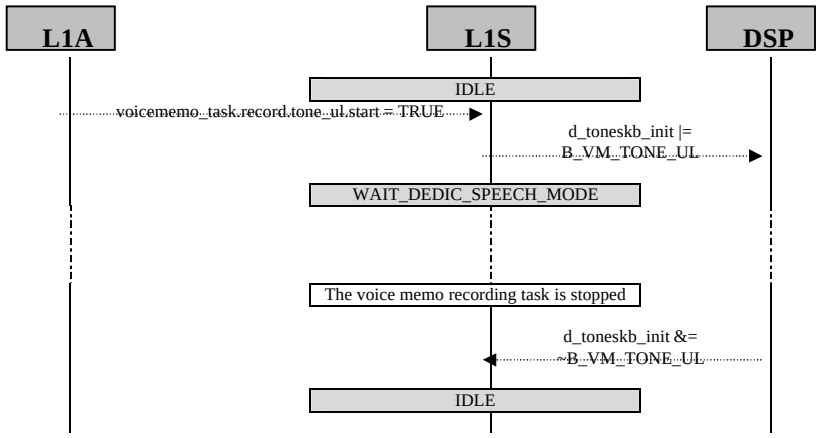
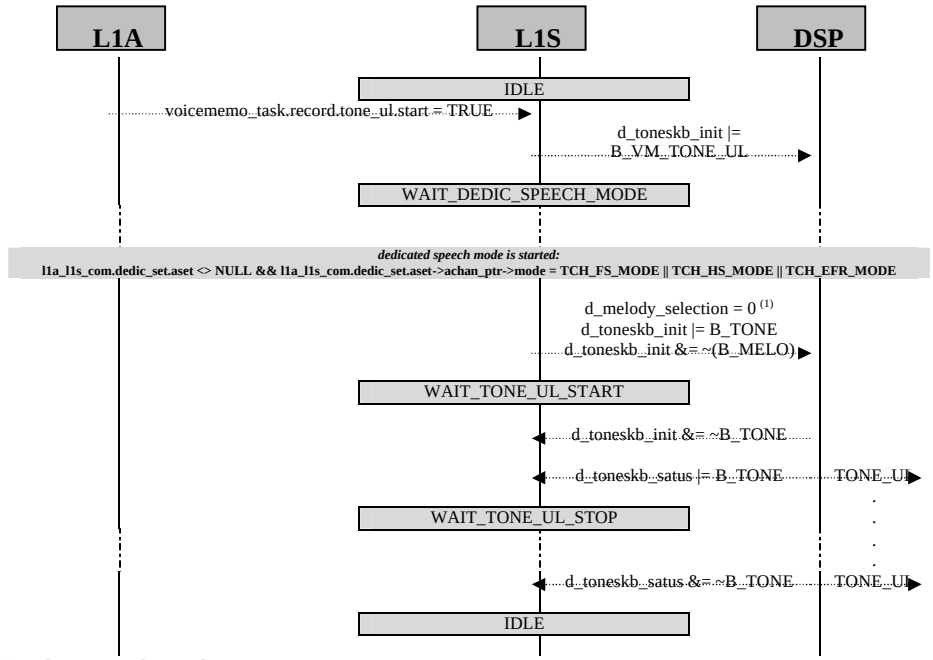
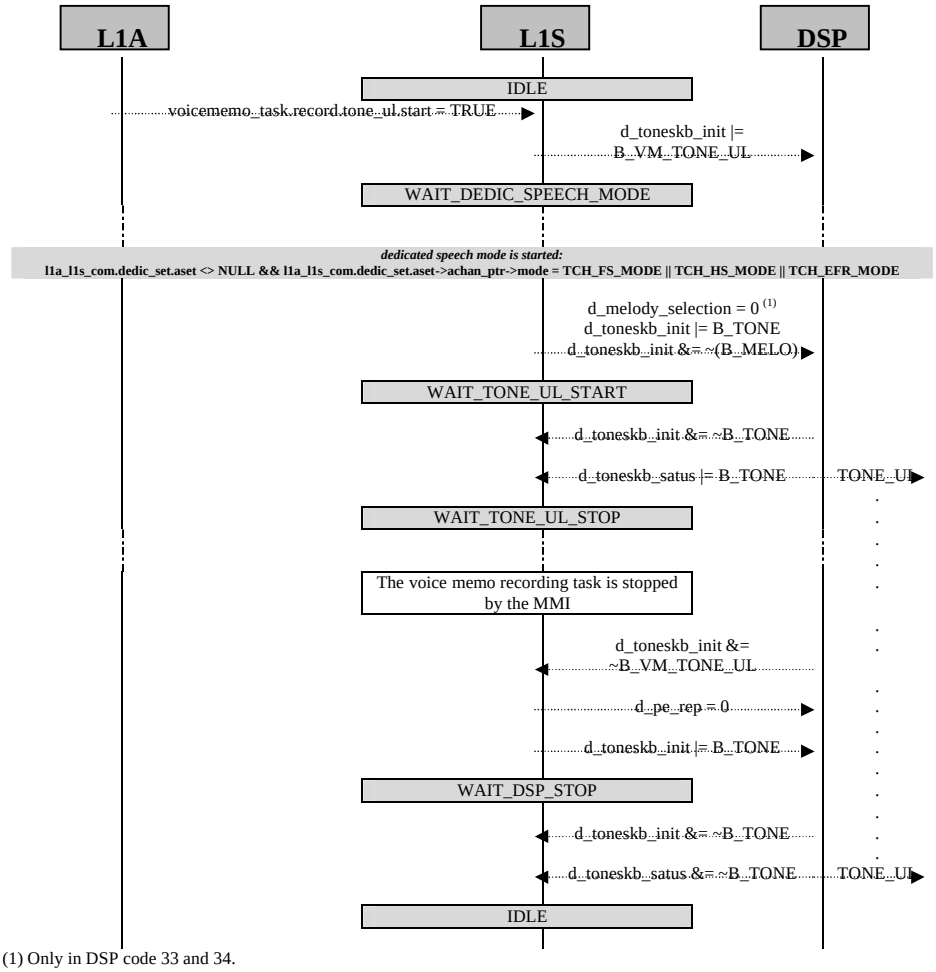


Figure 28: Voice memo tone uplink: recording in idle mode.



(1) Only in DSP code 33 and 34.

Figure 29: Voice memo tone uplink: recording in dedicated mode (automatic stop).



(1) Only in DSP code 33 and 34.

Figure 30: Voice memo tone uplink: recording in dedicated mode (resquested stop).

7.4.3 L1S state machine

Figure 31: Tone uplink L1S state machine

8 Voice memorisation AMR

8.1 Principle

The voice memorization AMR feature is a task to record and play some speech samples pronounced by the mobile user. So this task can be divided in two subtasks:

- Recording task: task to save the samples corresponding the to the user speech.
- Playing task: task to play the speech samples previously recorded.

The compression format used is based on the **AMR vocoders**. So, the rate ranges from 13 to 32 bytes per 20 ms. So to record 1 min. of speech, the MMI must allow 60s/0.02s * (13-32) = (39-96) 000bytes. Moreover if the DTX is used, the rate changes if the user speaks or not. If he doesn't, there will be a succession of SID_FIRST, NO_DATA and SID_UPDATE samples.
Voice Memo AMR is only available on AMR platforms (DSP code 34 and higher)

8.2 Format

Upper layers may need to play various AMR samples format (mainly MMS application). [1] defines Interface Format1 (IF1), which is a **generic frame format for frames of the AMR speech codec**. Applications related to AMR like MMS are derived from it. It is a **big-endian, byte-wise** (alignments are 8-bit) format. Therefore, DSP 16-bit granularity and little-endianess are not suited to upper layers development so it was decided that Layer1 would perform a small conversion step to be closer to IF1. Format shown below is detailed in this paragraph:

Header1	Data bits	Header2	Data bits	…	…	Header3	Data bits	END_MASK

8.2.1 Header

It is a byte containing the 8 LSB from DSP AMR header:

7	6	5	4	3	2	1	0
0	0	rx/tx_ type2	rx/tx_ type 1	rx/tx_ type 0	chan_ type2	chan_ type1	chan_ type0

Fields are defined as below:

Binary code	***b_amms_channel_type***	***B_amms_rx_type***	***B_amms_tx_type***
000	*4.75*	*SPEECH_GOOD*	*SPEECH_GOOD*
001	*5.15*	*Unused*	*Unused*
010	*5.9*	*Unused*	*Unused*
011	*6.7*	*SPEECH_BAD*	*Unused*
100	*7.4*	*SID_FIRST*	*SID_FIRST*
101	*7.95*	*SID_UPDATE*	*SID_UPDATE*
110	*10.2*	*SID_BAD*	*Unused*
111	*12.2*	*NO_DATA*	*NO_DATA*

This header is generic enough to identify all types of frame.

8.2.2 Data bits

The order and numbering notation of the bits are as specified for Interface Format 1 (IF1) in [1] for AMR. As specified there, the bits of speech frames have been rearranged in order of decreasing sensitivity, while bits of comfort noise frames are in the order produced by the encoder. These data are **8-bit aligned with a big-endian format**.
Note1: SID frames are handled specifically. Although SID_FIRST has no useful data bits and could be coded as a simple header, Layer1 format **includes the same number of comfort noise parameters (data bits) as SID_UPDATE/SID_BAD** in order to be close to IF1. However, Layer1 doesn't set these bits to "0" for SID_FIRST nor sets **STI and Mode Indication** bits defined in [1].

8.2.3 End mask

To mark automatic end of voice memo during play task, a 8-bit word must be added at the end of the Voice Memo (it is added by Layer1 during record). It is defined as the constant **SC_VM_AMR_END_MASK.**

8.2.4 Example

Here is the Layer1 format for SPEECH frame, 5.15 kbits/s, one 8-bit header + 13 data bytes noted from D1 to D13:

DSP Header (8 LSB bits)	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13

DSP format. Because of endianess, in memory it looks like:

DSP Header used bits	DSP Header Unused	Unused	Unused	Unused	Unused	… See data bits below

D2	D1	D4	D3	D6	D5	D8	D7	D10	D9	D12	D11	Unused	D13

8.3 Playing task

8.3.1 MMI rules

The MMI must respect the following rules

- When the MMI starts a voice memorization playing task (MMI_VM_AMR_PLAY_START_REQ message), it must wait the start confirmation (MMI_VM_AMR_PLAY_START_CON) before to stop this task.

- The MMI isn't allowed to start a new voice memorization AMR playing task before to receive a stop confirmation message (MMI_VM_AMR_PLAY_STOP_CON).
- The MMI can receive a stop confirmation message (MMI_VM_AMR_PLAY_STOP_CON) in two cases:
  - The MMI requested to stop the current voice memorization playing task (i.e. MMI_VM_AMR_PLAY_STOP_REQ) and the L1 confirms with the stop confirmation message.
  - The current task is stopped automatically (end of the voice buffer) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to run a voice memorization playing task with:
  - A melody 0.
  - A melody 1.
  - A voice memorization recording or playing.
  - A voice memorization AMR recording.
  - A speech recognition (enrollment, update, update-check, recognition).

- The MMI is allowed to run the voice memorization AMR playing task in GSM Idle mode and all GPRS modes, with TIDE feature not activated. Other modes are under investigation.

8.3.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	**MMI_VM_AMR_PLAY_START_REQ**	**T_MMI_VM_AMR_PLAY_REQ**
MMI->L1	**MMI_VM_AMR_PLAY_STOP_REQ**	**Trigger**
MMI<-L1	**MMI_VM_AMR_PLAY_START_CON**	**Trigger**
MMI<-L1	**MMI_VM_AMR_PLAY_STOP_CON**	**Trigger**

T_MMI_VM_AMR_PLAY_REQ message type
session_id (UWORD8) Specifies the customer data identification corresponding to this melody (for L1-upper layer communication).

8.3.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	**L1_VM_AMR_PLAY_START_CON**	**Trigger**
L1A<-L1S	**L1_VM_AMR_PLAY_STOP_CON**	**Trigger**

8.3.4 Process flow

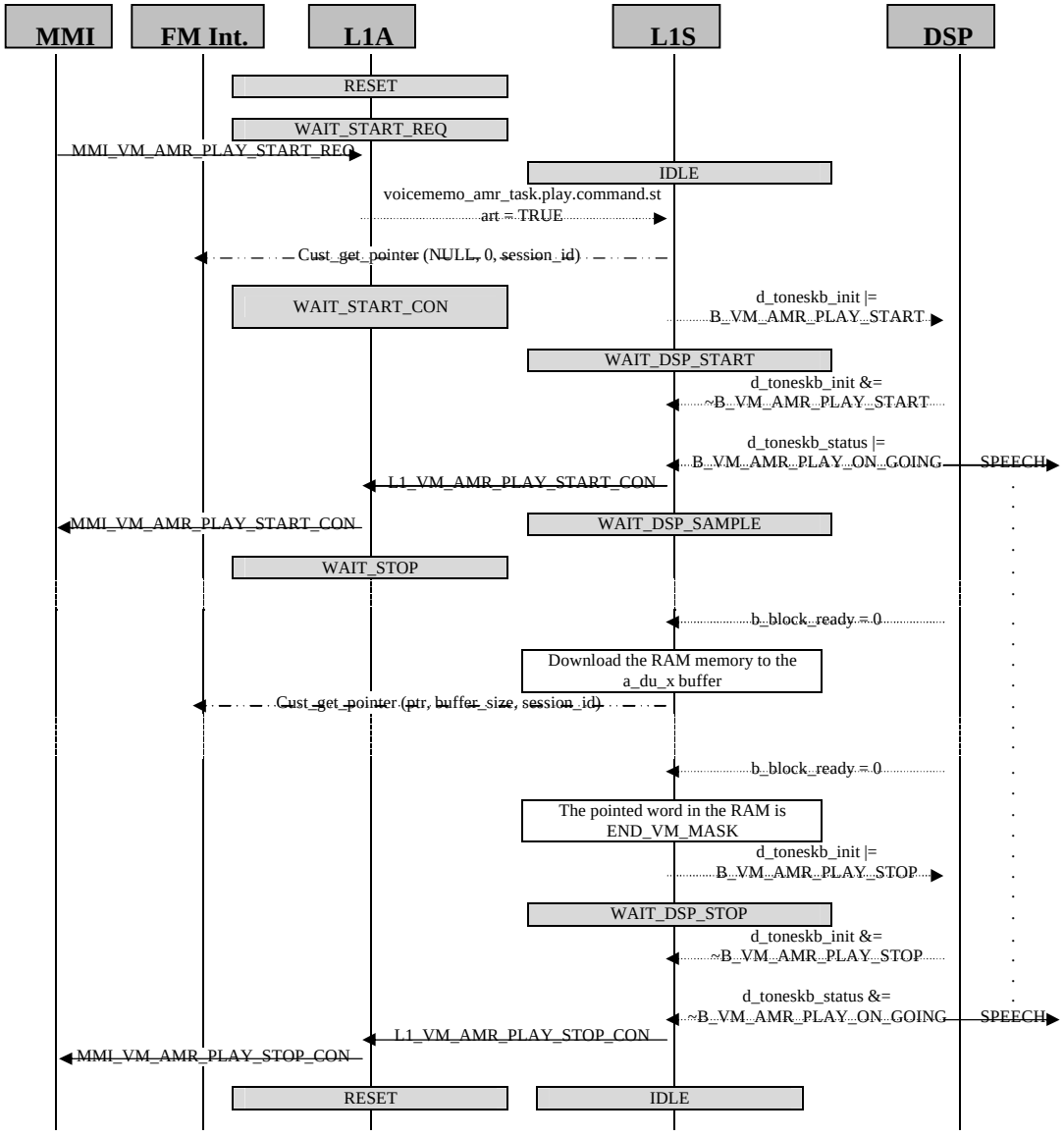


Figure 32: Voice memo AMR (playing in all mode with an automatic stop)

8.3.5 L1A state machine

Figure 33: Voice memo AMR playing (L1A state machine)

8.3.6 L1S state machine

Figure 34: Voice memo AMR playing (L1S state machine)

8.4 Recording task

8.4.1 MMI rules

The MMI must respect the following rules

- When MMI starts a voice memorization AMR recording task (MMI_VM_AMR_RECORD_START_REQ message), it must wait the start confirmation (MMI_VM_AMR_RECORD_START_CON) before to stop this task.
- The MMI isn't allowed to start a new voice memorization AMR task before to receive a stop confirmation message (MMI_VM_AMR_RECORD_STOP_CON).
- The MMI can receive a stop confirmation message (MMI_VM_AMR_RECORD_STOP_CON) in two cases:
  - The MMI requested to stop the current voice memorization AMR recording task (i.e. MMI_VM_AMR_RECORD_STOP_REQ) and the L1 confirms with the stop confirmation message.
  - The recording task is stopped automatically (end of the voice buffer) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to run a voice memorization recording task with:
  - A melody 0.
  - A melody 1.
  - A tone.
  - A speech recognition (enrollment, update, update-check, recognition).
  - A voice memorization AMR playing.
  - A voice memorization playing or recording.
- The MMI is allowed to run the voice memorization AMR recording task in GSM Idle mode and all GPRS modes, with TIDE feature not activated. Other modes are under investigation.

8.4.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_VM_AMR_RECORD_START_REQ	T_MMI_VM_AMR_RECORD_REQ
MMI->L1	MMI_VM_AMR_RECORD_STOP_REQ	Trigger
MMI<-L1	MMI_VM_AMR_RECORD_START_CON	Trigger
MMI<-L1	MMI_VM_AMR_RECORD_STOP_CON	T_MMI_VM_AMR_RECORD_CON

T_MMI_VM_AMR_RECORD_CON message type

recorded_size(UWORD32)
Specifies the size in 16-bit word unit of the recorded data..
Note: It takes into account the end mask put at the end of the recorded data (i.e. SC_VM_END_MASK).

T_MMI_VM_AMR_RECORD_REQ message type

session_id (UWORD8)
Specifies the customer data identification of this voice memo (for L1-Flash manager communication).

maximum_size(UWORD32)
Specifies the maximum size (in 8-bit word unit) of the recorded data. This value takes into account the end mask put at the end of the recorded data (i.e. SC_VM_AMR_END_MASK).
Therefore, if the recorded duration is 1 min (without DTX, rate 32 8-bit word/20ms), the size is
60s/20 * 1000 *32 + 1 = 94,7 Kbytes

dtx_used(BOOL)
Specifies the DTX mode for the voice memo record (0 = No DTX, 1 = DTX)

record_coeff_ul (UWORD16)
Coefficient added during the voice memo recording to the uplink TCH path in format Q8.8.
Example: record_coef_ul = 0x0100 = 1, record_coef_ul = 0x0080 = 0,5.

amr_vocoder(UWORD 8)
AMR codec to be used during recording. (0 -> 7)

Value	0	1	2	3	4	5	6	7
Rate (kbps)	4.75	5.15	5.60	6.45	7.45	7.95	10.2	12.2

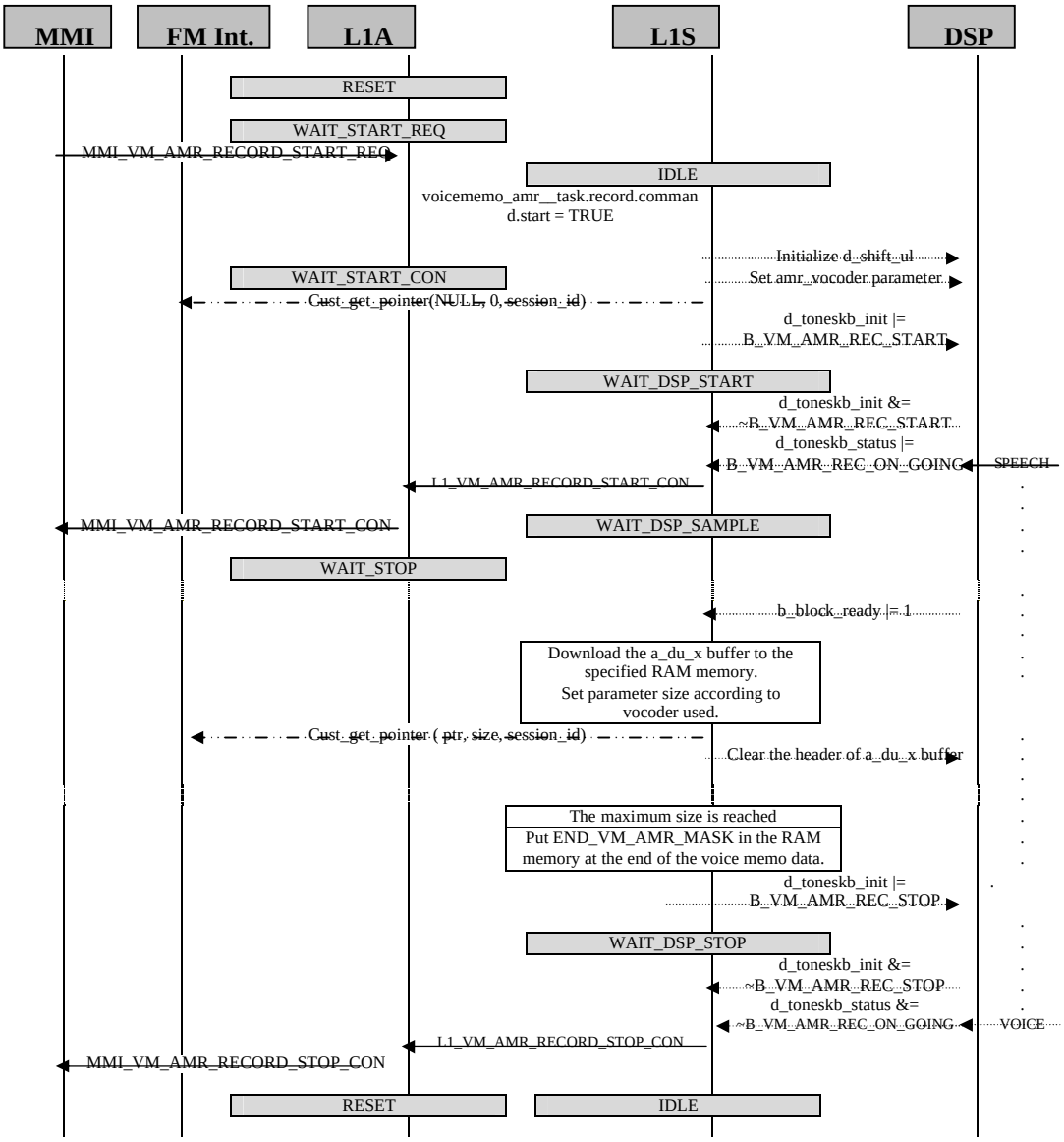


Figure 35: Voice memo (recording with an automatic stop)

8.4.3 L1A state machine

Figure 36: Voice memo AMR recording (L1A state machine)

8.4.4 L1S state machine

Figure 37: Voice memo AMR recording (L1S state machine)

9 Speech recognition

9.1 Principle

The speech recognition feature is used by the mobile user to execute some task thanks to its speech.
For example, with the speech recognition task, the user can pronounce the name of the person to call and the mobile can dial automatically the corresponding telephone number.
But to realize the example above some steps are needed:

- The user must enroll the word to recognize in a vocabulary database: it's the enrollment task. This task builds a model of the pronounced word and save it in the vocabulary database.
- The user can update each word of the vocabulary database to enhance the recognition of these words. It's he update task. This task averages the old model (i.e. enrollment phase) with a new model and save it in the vocabulary database.
- The user can update and the mobile can check if this updated word is close to the word to update. It's the update-check task. This task averages the old model with a new model and check if the averaged model is close to the old model or not.
- The mobile can recognize if the word pronounced by the user is in a vocabulary database and by the way execute the task corresponding to this word (name dialing, menu opening…). This task build a model of the word pronounced by the user and compare it with the model included in a vocabulary database.

Finally, a new task (Speech recognition background task) is created to exchange the model(between the MMI vocabulary database (generally in Flash memory) and the DSP via the API memory. The task is a preemptive task with a low priority (c.f. next chapter).

Locosto TCS3.2 doesn't support this feature.

9.2 Structure

As it said on the previous chapter, the speech recognition background database is created in order to load/save the model from/to the MMI vocabulary database and to/from the API memory (c.f. figure below). In fact, when a model must be saved/loaded to/from the MMI database, the L1A sends a message to the background task via the *Qsrback* queue to request to start a background task.
Some functions of the background task are customer dependent, so the background task call some customer function included in the MMI.
At the end of the background task, the speech recognition background task sends a message to the L1A via the *Q1* queue.

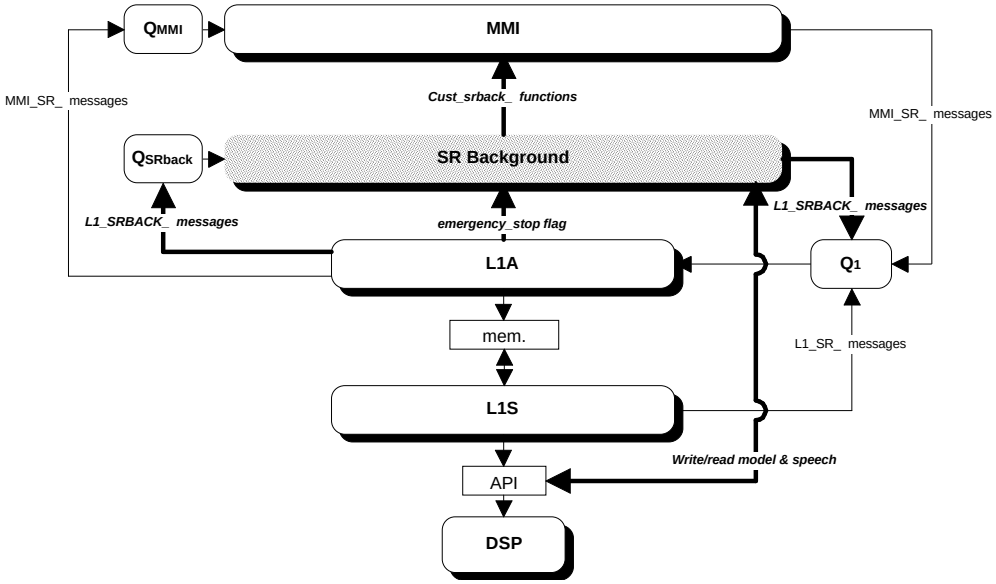


Figure 38: Speech recognition structure.

9.3 Speech recognition vocabulary database

The speech recognition database is customer dependent but the database must respect the following rules:

- The customer can create several vocabulary databases. It must only transmit to the L1 the ID of the database used.
- In the recognition phase, only the vocabulary of one database is used.
- The model of a database must be arrange from the index 0 to the index "*number of model in the current database – 1*".
- The maximum model size is 1041 16-bit words (i.e. SC_MMI_DB_MODEL_SIZE).
- The maximum speech size, in the database, is 1501 16-bit words (i.e. SC_MMI_DB_SPEECH_SIZE).
- The header of the model indicates the size of the model in 16 words unit (i.e. model frame unit).
- The speech sample is included between the beginning of the speech buffer to the end mask constant (i.e. SC_VM_END_MASK).
- The customer database can be in flash memory.

9.4 Enrollment task

The aim of this task is to acquire a new word for a MMI vocabulary database. So, the user says a word to enroll and the mobile builds a model of this word. This model is used as reference for the recognition phase.
Moreover, during the enrollment task, the user can record the speech sample of the word. So, thanks to the voice memorization playing task, the user can hear the word that he enrolls.

9.4.1 MMI rules

The MMI must respect the following rules:

- When the MMI starts a speech recognition enrollment task (MMI_SR_ENROLL_START_REQ message), it must wait the start confirmation (MMI_SR_ENROLL_START_CON) before to stop this task.
- The MMI can't start any speech recognition task before to receive a stop enroll confirmation message (MMI_SR_ENROLL_STOP_CON).
- The MMI can receive a stop enroll confirmation message (MMI_SR_ENROLL_STOP_CON) in two cases:
  - The MMI requested to stop the current speech recognition enroll task (i.e. MMI_SR_ENROLL_STOP_REQ) and the L1 confirms the stop enroll confirmation message.
  - The current task is stopped automatically (end of the enrollment or an error is occurred) and the L1 informs the MMI with the stop enroll confirmation message.
- The MMI isn't allowed to run a speech recognition enroll task with:
  - A voice memorization playing and recording.
  - An another speech recognition tasks (update, update-check and recognition).
  - Melody 0.
  - Melody 1.
- With the DSP code 17, 31 and 32, the enrollment task can't run in all GSM dedicated modes. And it can't run in all GPRS modes.
- With the DSP code 33, the enrollment task can't run in all dedicated modes, excepted the dedicated signaling only mode. And it can run in all GPRS modes (if the DSP background task is present (c.f. spec. S924).
- The UL gain must be constant during the speech recognition enroll task and not too high (no saturation).
- For the DSP code 32, during a transition between the GSM and the GPRS, the speech recognition enrollment task is stopped.

9.4.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	**MMI_SR_ENROLL_START_REQ**	**T_MMI_SR_ENROLL_REQ**
MMI<-L1	**MMI_SR_ENROLL_START_CON**	**Trigger**
MMI->L1	**MMI_SR_ENROLL_STOP_REQ**	**Trigger**
MMI<-L1	**MMI_SR_ENROLL_STOP_CON**	**T_MMI_ENROLL_STOP_CON**

T_MMI_SR_ENROLL_REQ message type

database_id (UWORD8)
Specifies which database is used.

word_index (UWORD8)
Specifies the index in the *"database_id"* database of the model to enroll.

speech (BOOL)
Specifies if the speech must be recorded or not.

speech_address (*UWORD16)
Specifies the address of the working circular buffer for the speech recording task.
Note: the size of this buffer is fixed and equals to SC_SR_MMI_2_L1_SPEECH_SIZE.
Moreover, the size of this buffer is bigger than the size of the speech buffer of the MMI vocabulary database: SC_SR_MMI_DB_SPEECH_SIZE < SC_SR_MMI_2_L1_SPEECH_SIZE.

T_MMI_SR_ENROLL_STOP_CON message type

Formatted: English (U.S.)

error_id (UWORD8)
Specifies the type of error.

value	id	type
0	SC_NO_ERROR	No error is occurred.
1	SC_BAD_ACQUISITION	Bad acquisition of the word. The word is too long or too short.
2	SC_TIME_OUT	The DSP task to acquire the word takes to much time.

9.4.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	L1_SR_ENROLL_START_CON	Trigger
L1A<-L1S	L1_SR_ENROLL_STOP_CON	T_L1_ENROLL_STOP_CON

T_L1_SR_ENROLL_STOP_CON message type

error_id (UWORD8)
Specifies the type of error.

value	id	type
0	SC_NO_ERROR	No error is occurred.
1	SC_BAD_ACQUISITION	Bad acquisition of the word. The word is too long or too short.
2	SC_TIME_OUT	The DSP task to acquire the word takes to much time.

Formatted: English (U.S.)

9.4.4 Process flow

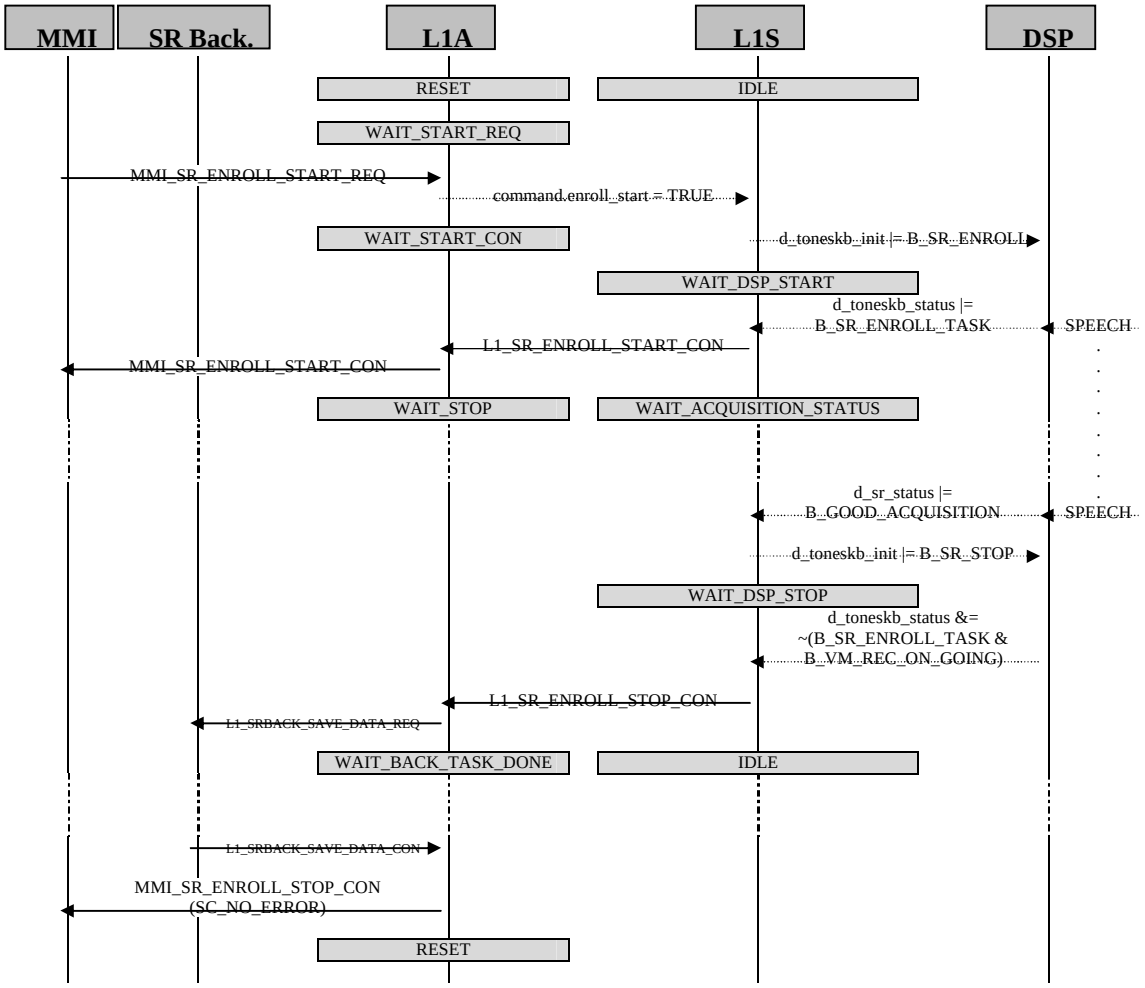


Figure 39: Enrollment task: enroll no error and no resquested stop.

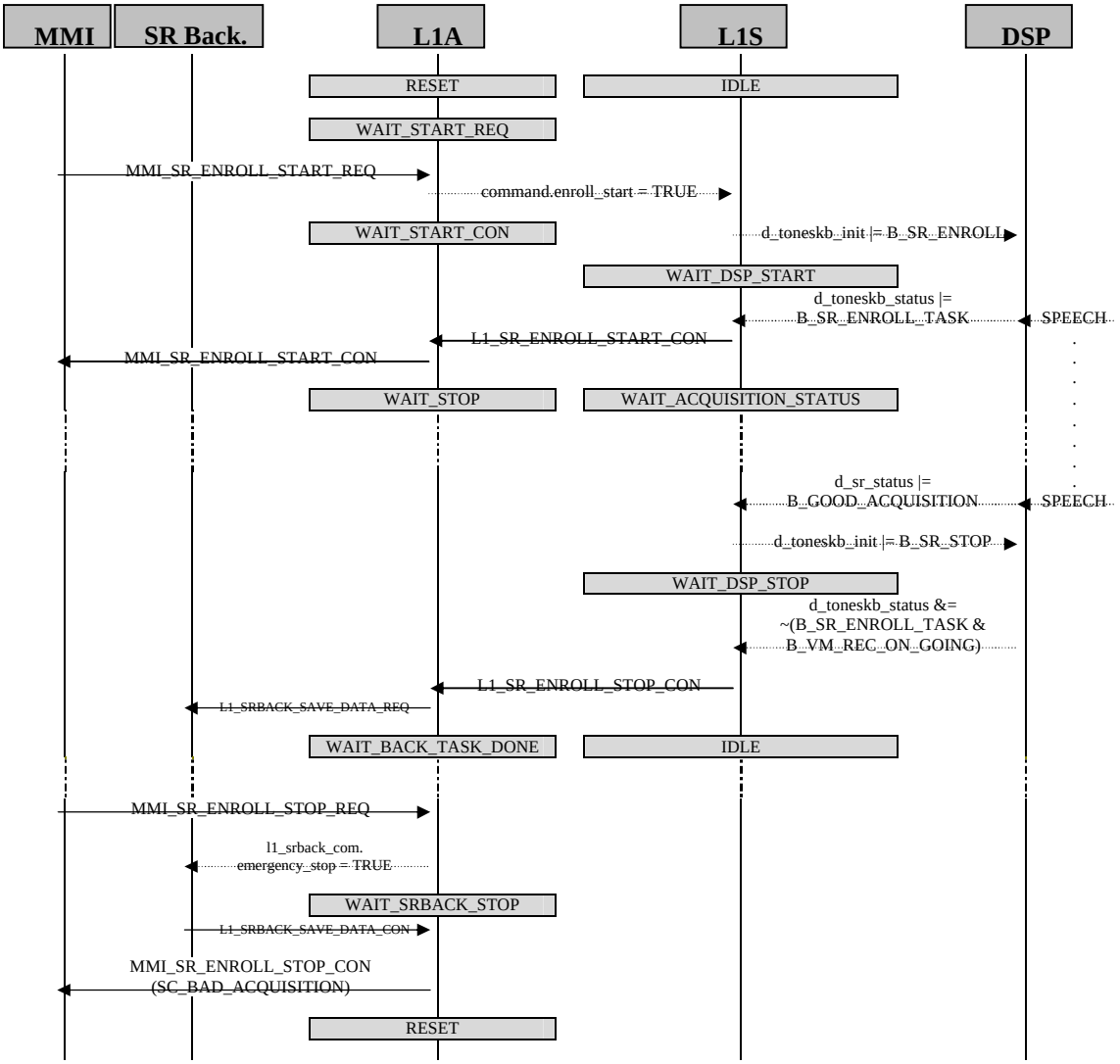


Figure 40: Enrollment task: enroll no error and resquested stop (L1S was stopped).

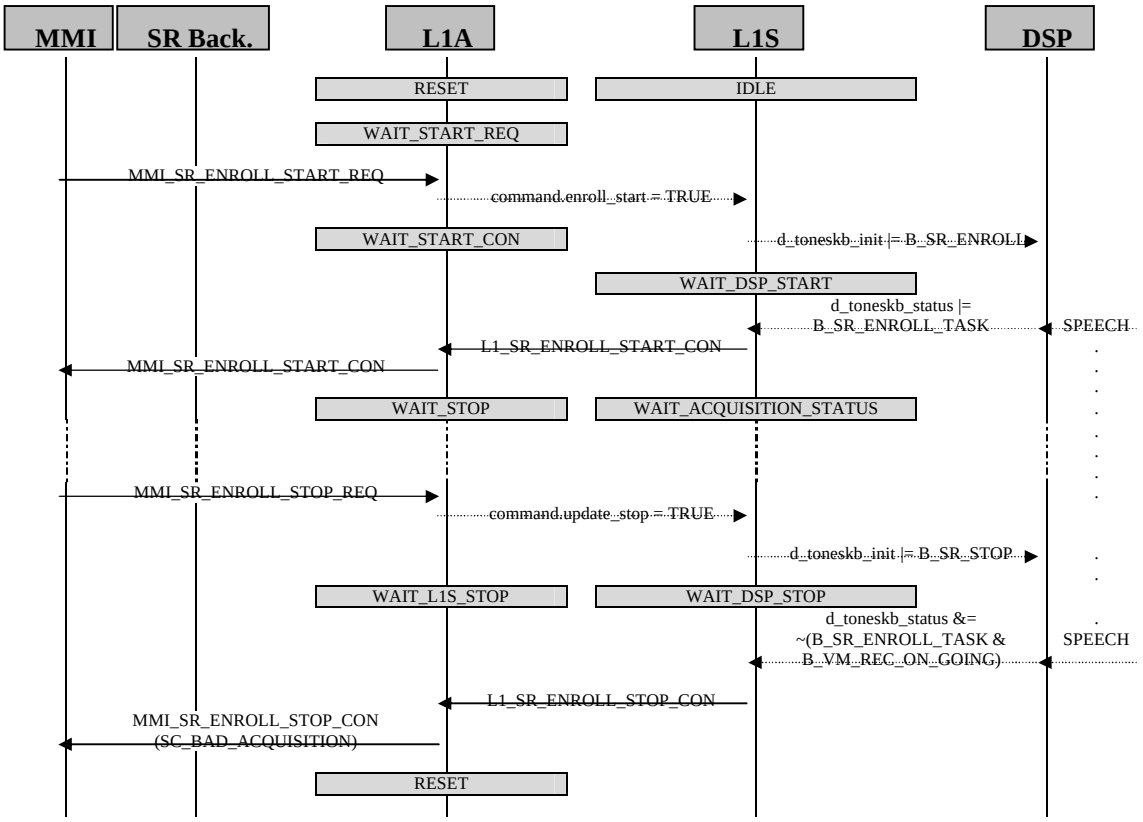


Figure 41: Enrollment task: enroll no error and resquested stop (L1S is running).

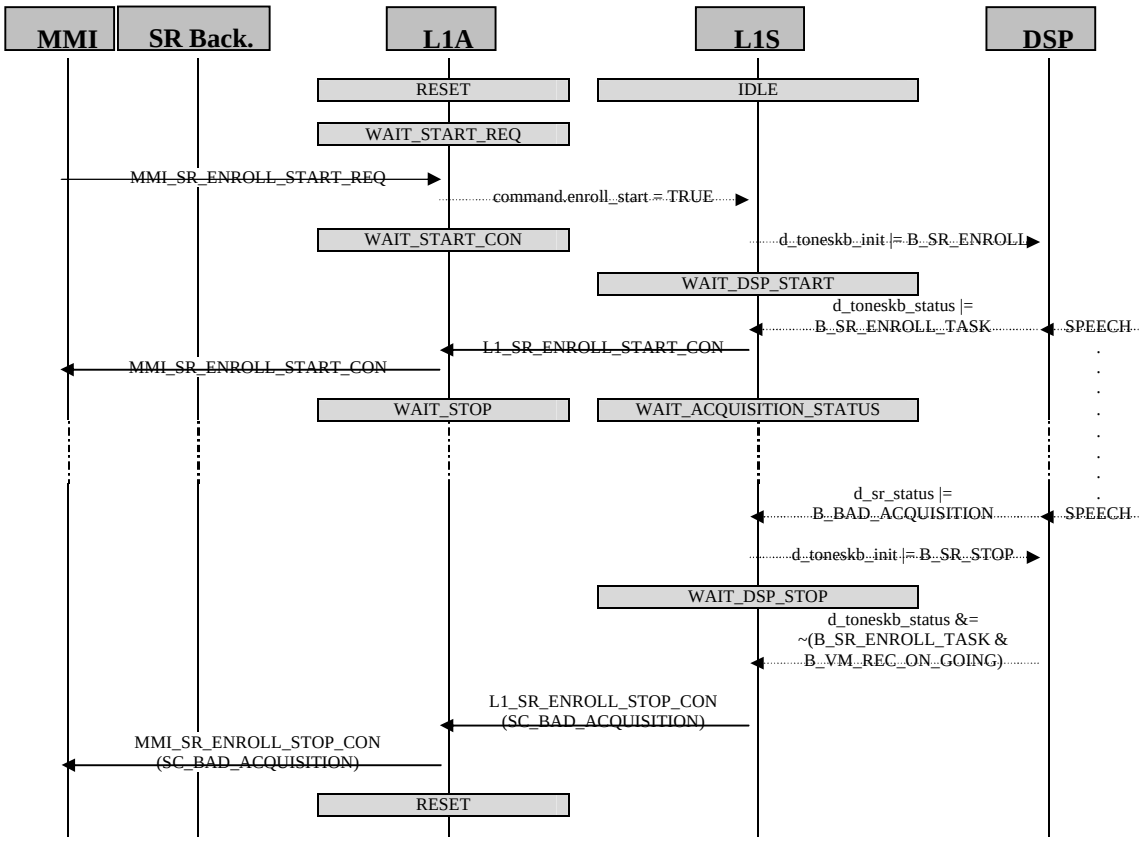


Figure 42: Enrollment task: enroll error (bad acquisition) and no resquested stop.

9.4.5 L1A state machine

9.4.6 L1S state machine

9.5 Update task

The aim of this task is to update a model of a word of a MMI vocabulary database. So, the user pronounces again a word contained in the MMI database. And the mobile builds a new model of this word thanks to the old model and the new model just acquired. This model is a king of average of two models of the same word. It permits to have a better model than the model acquired by the enrollment task.
Moreover, during the update task, the user can record again the speech sample of the word. This new speech samples replace the samples acquired during the enroll task. And, thanks to the voice memorization playing task, the user can hear this word that he updates.

9.5.1 MMI rules

The MMI must respect the following rules:

- When the MMI starts a speech recognition update task (MMI_SR_UPDATE_START_REQ message), it must wait the start confirmation (MMI_SR_UPDATE_START_CON) before to stop this task.
- The MMI isn't allowed to start a new speech recognition task before to receive a stop update confirmation message (MMI_SR_UPDATE_STOP_CON).
- The MMI can receive a stop update confirmation message (MMI_SR_UPDATE_STOP_CON) in two cases:
  - The MMI requested to stop the current speech recognition update task (i.e. MMI_SR_UPDATE_STOP_REQ) and the L1 confirms with the stop confirmation message.
  - The current task is stopped automatically (end of the update or an error is occurred) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to run a speech recognition update task with:
  - A voice memorization playing and recording.
  - An another speech recognition tasks (enroll, update-check and recognition).
  - A melody 1
  - A melody 0
- With the DSP code 17, 31 and 32, the update task can't run in all GSM dedicated modes. And it can't run in all GPRS modes.
- With the DSP code 33, the update task can't run in all dedicated modes, excepted the dedicated signaling only mode. And it can run in all GPRS modes (if the DSP background task is present (c.f. spec. S924).
- The speech recognition update task can't run if the user database is empty.
- The update task must have the same audio configuration (FIR coefficients…) than the enrollment task.
- The UL gain must be constant and the same than the enrollment task and not too high (no saturation).
- For the DSP code 32, during a transition between the GSM and the GPRS, the speech recognition update task is stopped.

9.5.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_SR_UPDATE_START_REQ	T_MMI_SR_UPDATE_REQ
MMI<-L1	MMI_SR_UPDATE_START_CON	Trigger
MMI->L1	MMI_SR_UPDATE_STOP_REQ	Trigger
MMI<-L1	MMI_SR_UPDATE_STOP_CON	T_MMI_UPDATE_STOP_CON

T_MMI_SR_UPDATE_REQ message type

database_id (UWORD8)
Specifies which database is used.

word_index (UWORD8)
Specifies the index in the *"database_id"* database of the model to update.

speech (BOOL)
Specifies if the speech must be recorded or not.

speech_address (*UWORD16)
Specifies the address of the working circular buffer used by the speech recording task.
Note: the size of this buffer is fixed by the value SC_SR_MMI_2_L1_SPEECH_SIZE.
Moreover, the size of this buffer is bigger than the size of the speech buffer for each word of the MMI vocabulary database: SC_SR_MMI_DB_SPEECH_SIZE < SC_SR_MMI_2_L1_SPEECH_SIZE.

T_MMI_SR_UPDATE_STOP_CON message type

error_id (UWORD8)
Specifies the type of error.

value	id	type
0	SC_NO_ERROR	No error is occurred.
1	SC_BAD_ACQUISITION	Bad acquisition of the word. The word is too long or too short.
2	SC_TIME_OUT	The time to acquire or to update the word is too long.
3	SC_BAD_UPDATE	Bad update of the model. The model from database is too different than the new model (built during the acquisition).

9.5.3 L1A-L1S message interface

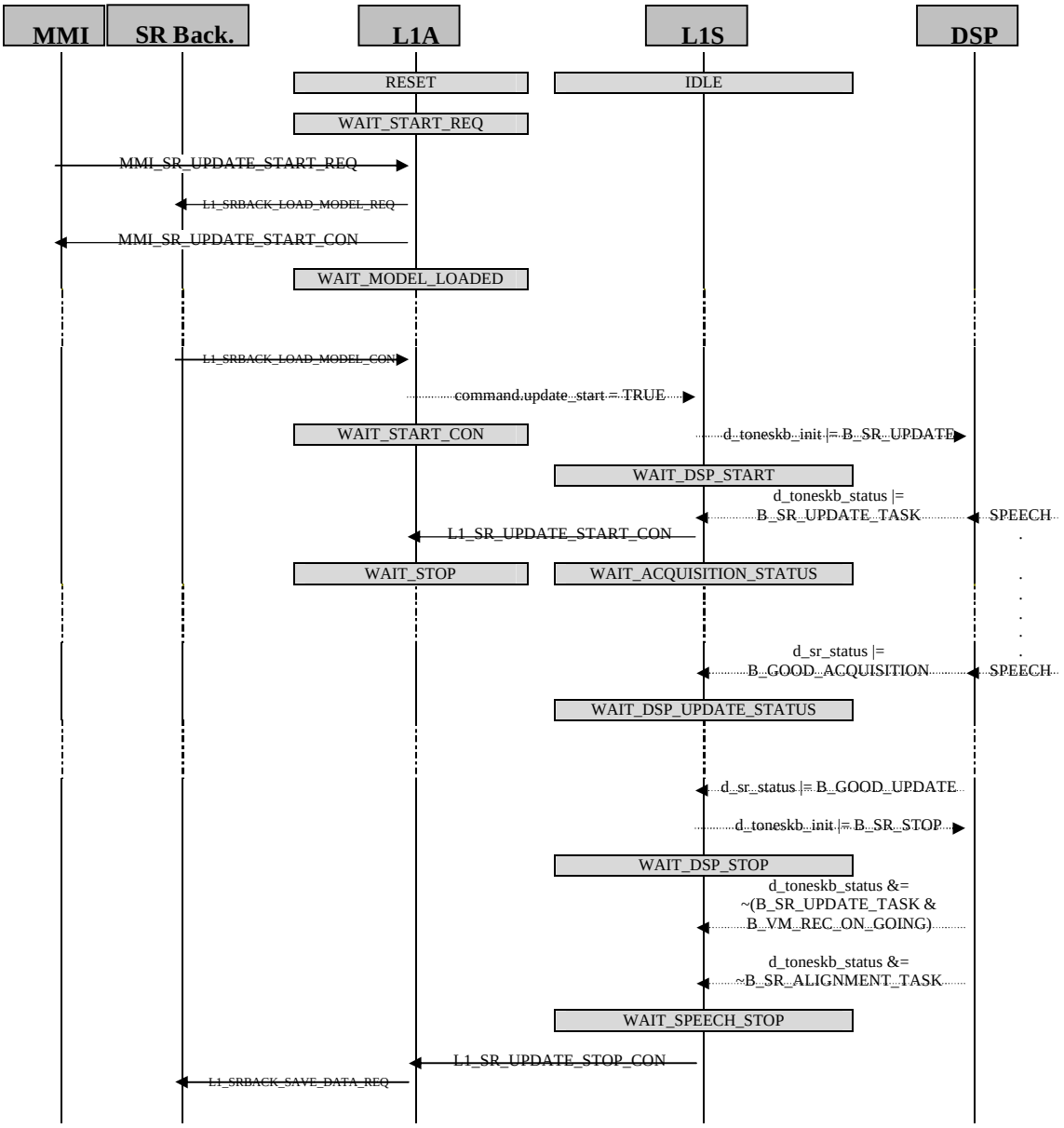
Direction	Message name	Type
L1A->L1S	**L1_SR_UPDATE_START_CON**	**Trigger**
L1A->L1S	**L1_SR_UPDATE_STOP_CON**	**T_L1_UPDATE_STOP_CON**

T_L1_SR_UPDATE_STOP_CON message type

error_id (UWORD8)
Specifies the type of error.

value	id	type
0	SC_NO_ERROR	No error is occurred.
1	SC_BAD_ACQUISITION	Bad acquisition of the word. The word is too long or too short.
2	SC_TIME_OUT	The time to acquire or to update the word is too long.
3	SC_BAD_UPDATE	Bad update of the model. The model from database is too different than the new model (built during the acquisition).

9.5.4 Process flow



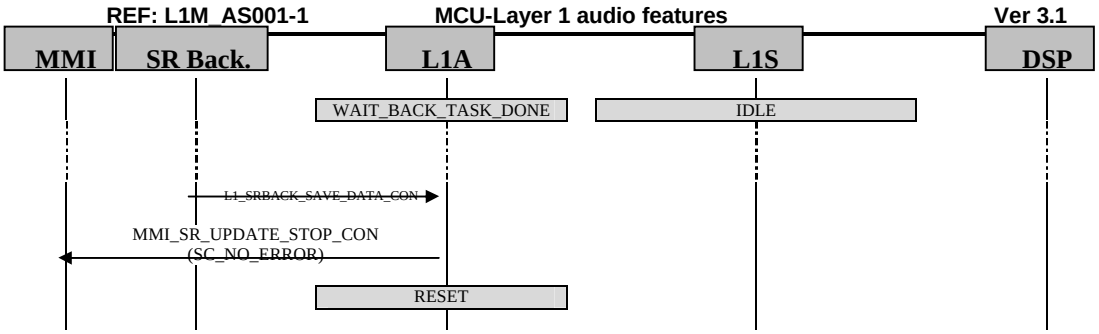


Figure 47: Update task: Update no error and no resquested stop.

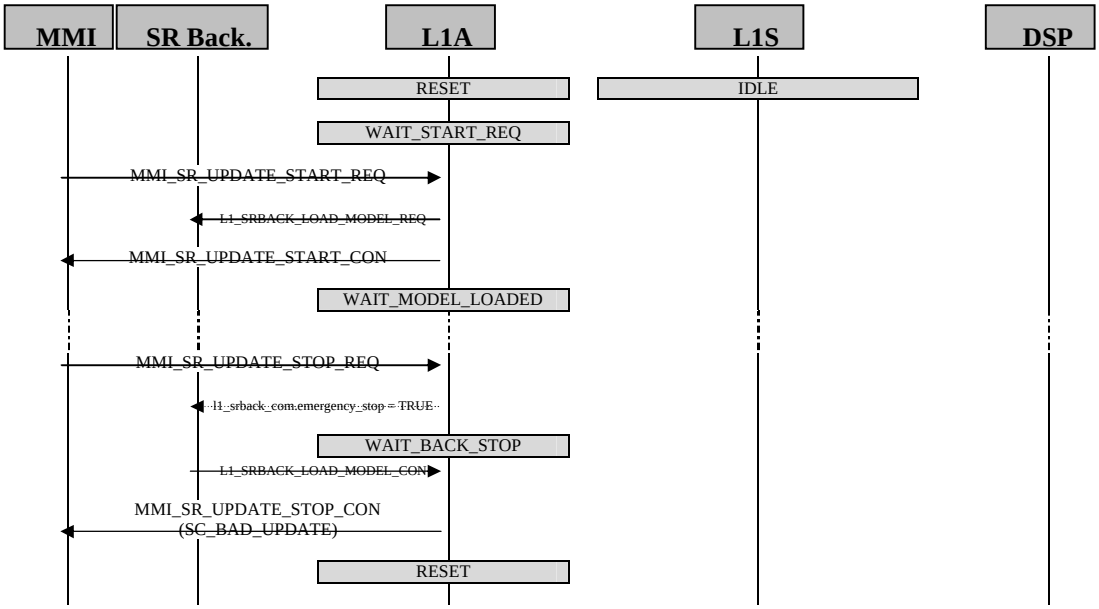


Figure 48: Update task: Update no error and resquested stop (L1S isn't running).

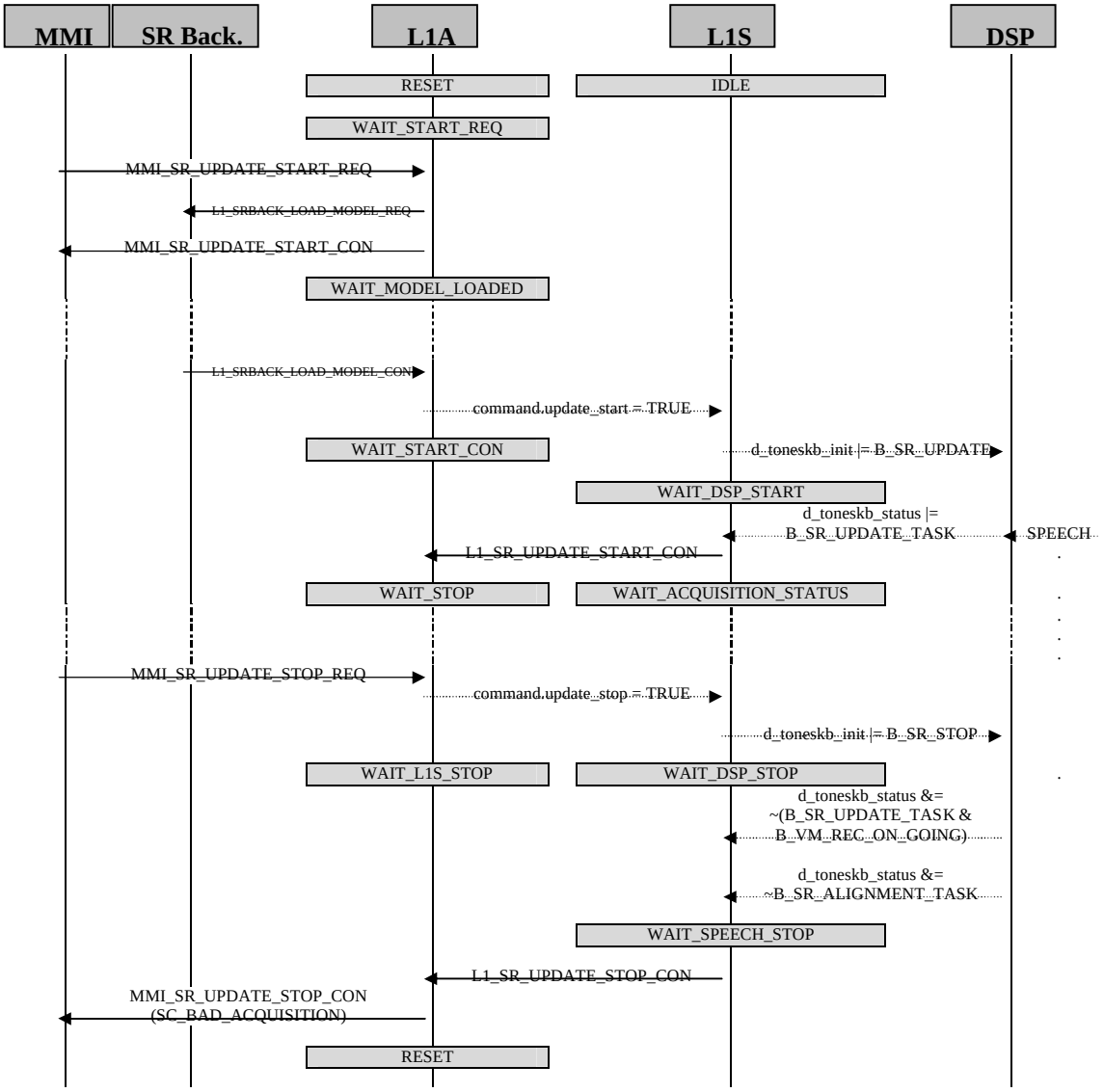


Figure 49: Update task: Update no error and resquested stop (L1S is running).

9.5.5 L1A state machine

9.5.6 L1S state machine

9.6 Reco task

The aim of this task is to find the model included in the MMI vocabulary database corresponding to the word pronounced by the user. So, the user says a word and the mobile builds its model. And it compares this model to the models contained in the MMI vocabulary database. For each comparison, a score is attributed. Finally, after these comparisons, the word corresponding to the best score is the elected model. Note that a CTO (Close Too Original) algorithm is activated if the number of model in the database is low (<4). This algorithm builds some dummy models to increase the performance of the recognition algorithm when the number of model in the vocabulary is to low.

9.6.1 MMI rules

The MMI must respect the following rules:

- When the MMI starts a speech recognition task (MMI_SR_RECO_START_REQ message), it must wait the start confirmation (MMI_SR_RECO_START_CON) before to stop this task.
- The MMI isn't allowed to start a new speech recognition task before to receive a stop recognition confirmation message (MMI_SR_RECO_STOP_CON).
- The MMI can receive a stop recognition confirmation message (MMI_SR_RECO_STOP_CON) in two cases:
  - The MMI requested to stop the current speech recognition task (i.e. MMI_SR_RECO_STOP_REQ) and the L1 confirms with the stop confirmation message.
  - The current task is stopped automatically (end of the recognition or an error is occurred) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to run a speech recognition update task with:
  - A voice memorization playing and recording.
  - An another speech recognition tasks (update, update-check and enrollment).
  - Melody 1.
  - Melody 0.
- With the DSP code 17, 31 and 32, the recognition task can't run in all GSM dedicated modes. And it can't run in all GPRS modes.
- With the DSP code 33, the recognition task can't run in all dedicated modes, excepted the dedicated signaling only mode. And it can run in all GPRS modes (if the DSP background task is present (c.f. spec. S924).
- The speech recognition task can't run if the user database is empty.
- The speech recognition task must have the same audio configuration (FIR coefficients…) than the enrollment/update task.
- The UL gain must be constant during the speech recognition task and not too high (no saturation).
- The index of the words included in the MMI database must start from 0 to the "*number of model*".
- For the DSP code 32, during a transition between the GSM and the GPRS, the recognition task is stopped.

9.6.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_SR_RECO_START_REQ	T_MMI_SR_RECO_REQ
MMI<-L1	MMI_SR_RECO_START_CON	Trigger
MMI->L1	MMI_SR_RECO_STOP_REQ	Trigger
MMI<-L1	MMI_SR_RECO_STOP_CON	T_MMI_SR_RECO_STOP_CON

T_MMI_SR_RECO_REQ message type

database_id (UWORD8)
Specifies which user database is used.

vocabulary_size (UWORD8)
Specifies the number of word included in the *"database_id"* database.

T_MMI_SR_RECO_STOP_CON message type

error_id (UWORD8)
Specifies the type of error.

value	id	type
0	SC_NO_ERROR	No error is occurred.
1	SC_BAD_ACQUISITION	Bad acquisition of the word. The word is too long or too short.
2	SC_TIME_OUT	The time to acquire or to compare a word is too long.
4	SC_BAD_RECOGNITION	The word is out of vocabulary or the best words are too close.
5	SC_CTO_WORD	A word generated by the CTO algorithm is the best word.

best_word_index (UWOR16)
Gives the index in the user database of the recognized word.
Note: this value is valid only if there's no error.

best_word_score (UWORD32)
Gives the scores of the recognized word. The DSP OOV algorithm calculated this score.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

second_best_word_index (UWOR16)
Gives the index in the user database of the second recognized word.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

second_best_word_score (UWORD32)
Gives the scores of the second recognized word. The DSP OOV algorithm calculated this score.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

third_best_word_index (UWOR16)
Gives the index in the user database of the third recognized word.
Note: this value is valid only if the MMI doesn't stop the speech recognition task.
(useful for a customer OOV algorithm)

third_best_word_score (UWORD32)
Gives the scores of the third recognized word. The DSP OOV algorithm calculated this score.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

fourth_best_word_index (UWOR16)
Gives the index in the user database of the third recognized word.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

fourth_best_word_score (UWORD32)
Gives the scores of the third recognized word. The DSP OOV algorithm calculated this score.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

d_sr_db_level (UWORD16)
Gives the estimate voice level in dB. This information is used to tune the speech recognition task.
Note: this value is valid only if there's no error.
(useful to tune the speech recognition)

d_sr_db_noise (UWORD16)
Gives the estimate noise level in dB. This information is used to tune the speech recognition task.
Note: this value is valid only if there's no error.
(useful to tune the speech recognition)

d_sr_model_size (UWORD16)
Gives the size of the model acquired during the speech recognition task. This information is used to tune the speech recognition task.
Note: this value is valid only if there's no error.
(useful to tune the speech recognition)

9.6.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	**L1_SR_RECO_START_CON**	**Trigger**
L1A<-L1S	**L1_SR_RECO_STOP_CON**	**T_L1_SR_RECO_STOP_CON**
L1A<-L1S	**L1_SR_PROCESSING_STOP_CON**	**T_L1_SR_PROCESSING_STOP_CON**
L1A<-L1S	**L1_SR_RECO_STOP_IND**	**T_L1_SR_RECO_STOP_IND**

T_L1_SR_RECO_STOP_CON message type

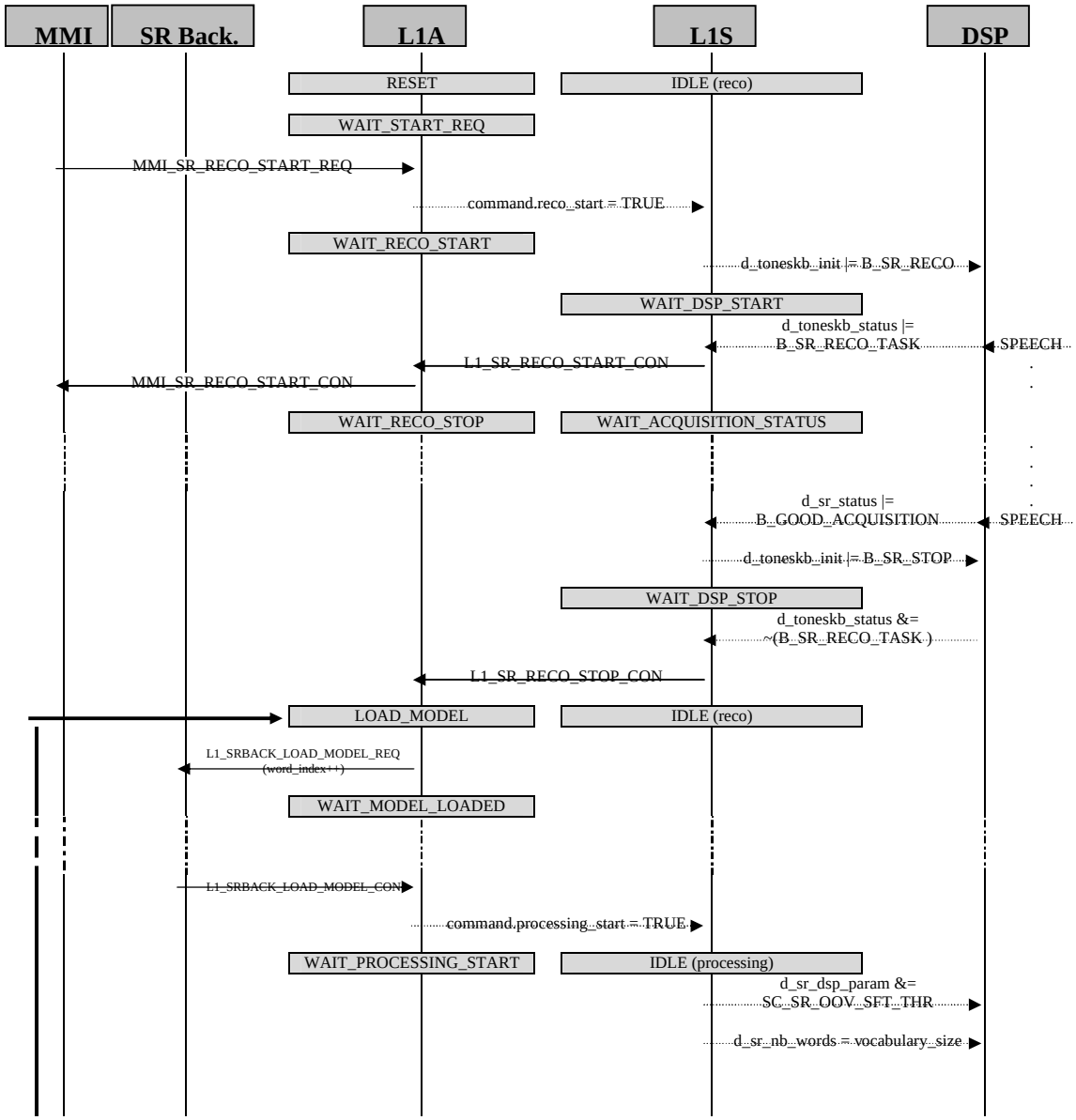
error_id (UWORD8)
Specifies the type of error.

value	id	type
0	SC_NO_ERROR	No error is occurred.
1	SC_BAD_ACQUISITION	Bad acquisition of the word. The word is too long or too short.
2	SC_TIME_OUT	The time to acquire or to compare a word is too long.
4	SC_BAD_RECOGNITION	The word is out of vocabulary or the best words are too close.
5	SC_CTO_WORD	A word generated by the CTO algorithm is the best word.

T_L1_SR_RECO_STOP_IND message type:
C.f. T_MMI_SR_RECO_STOP_CON message type.

T_L1_SR_PROCESSING_STOP_CON message type:
C.f. T_MMI_SR_RECO_STOP_CON message type.

9.6.4 Process flow



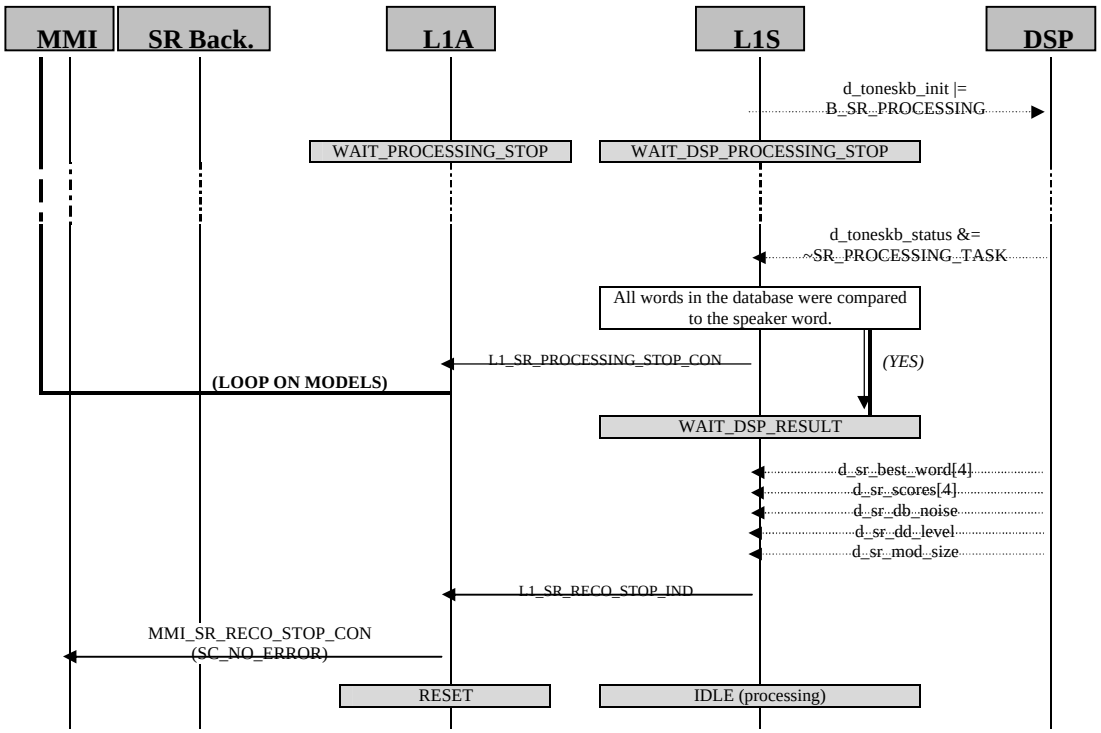


Figure 54: Reco task: reco no error and no resquested stop.

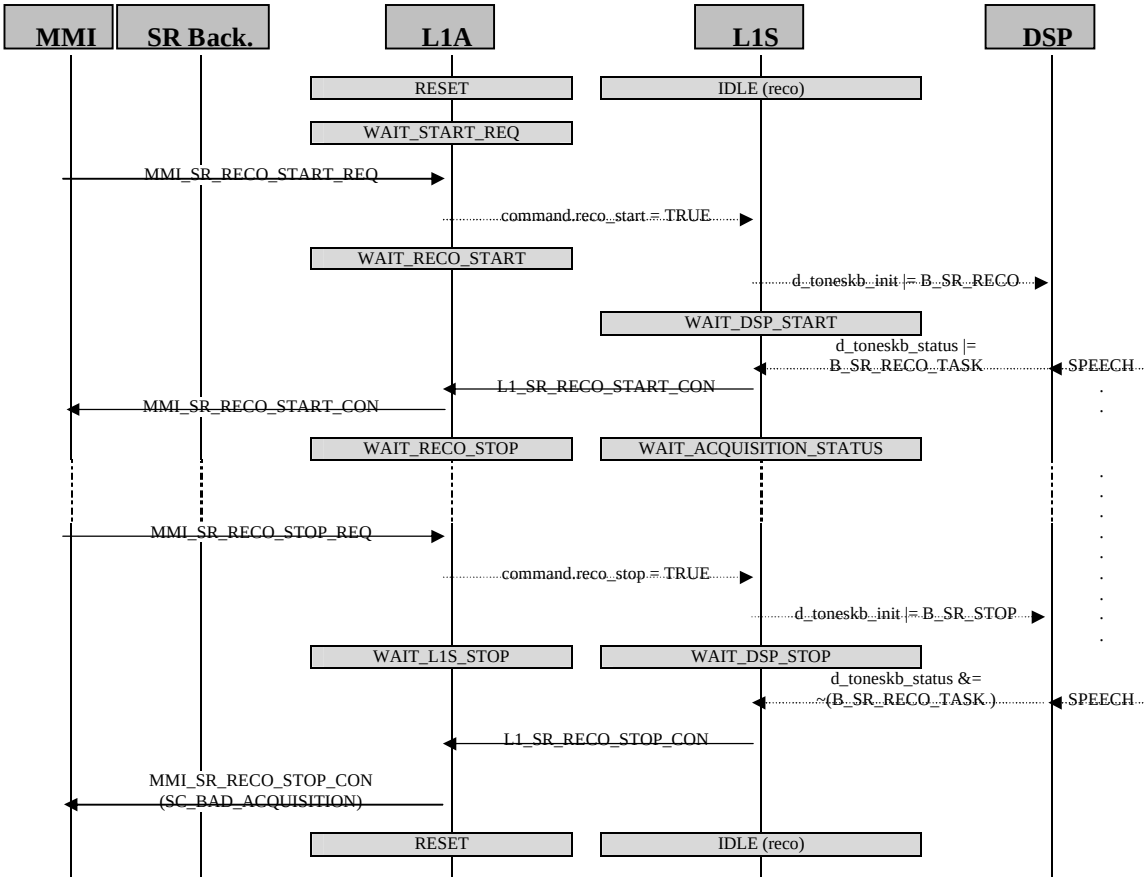


Figure 55: Reco task: reco no error and resquested stop (L1S is in acquisition phase).

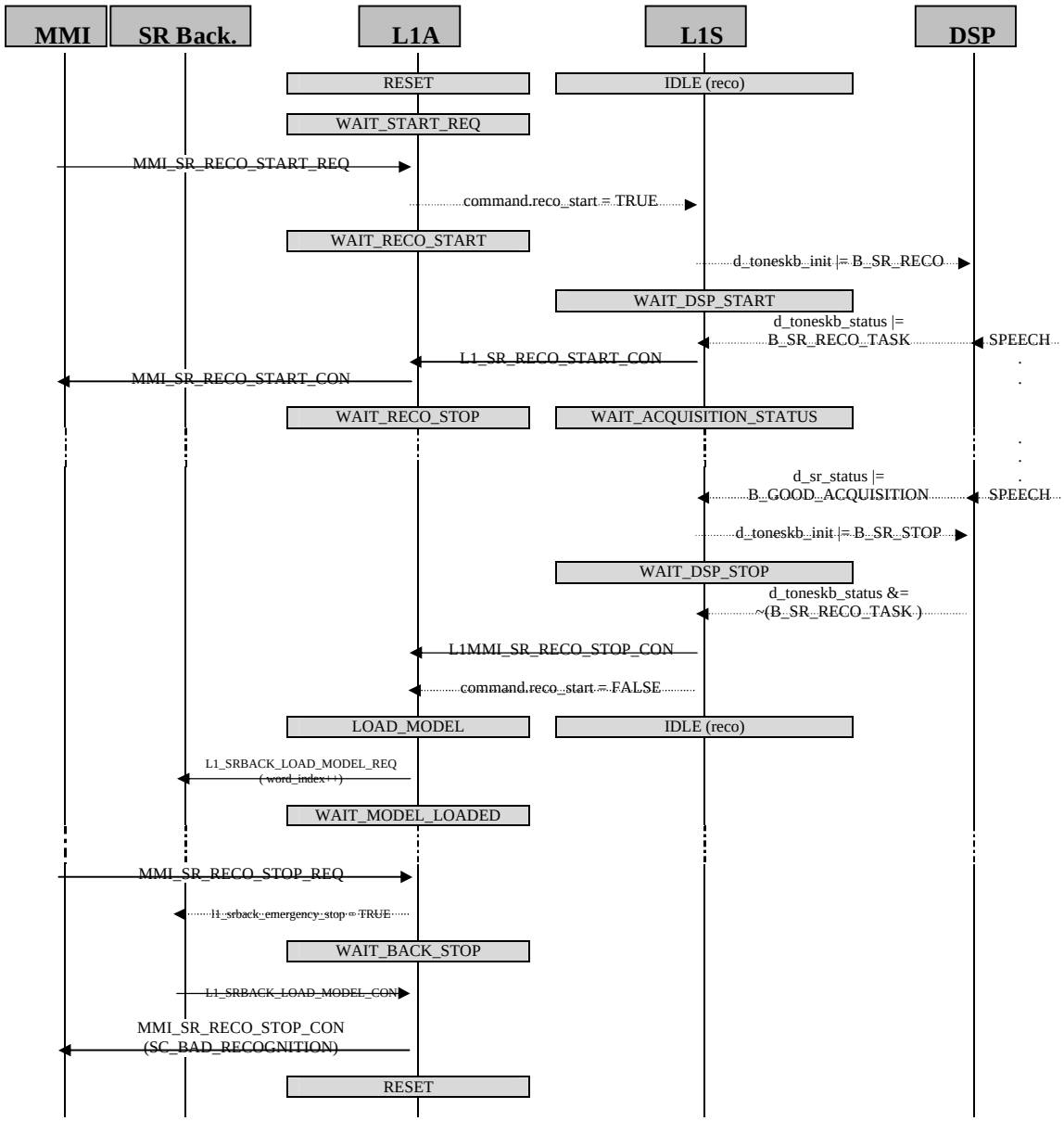
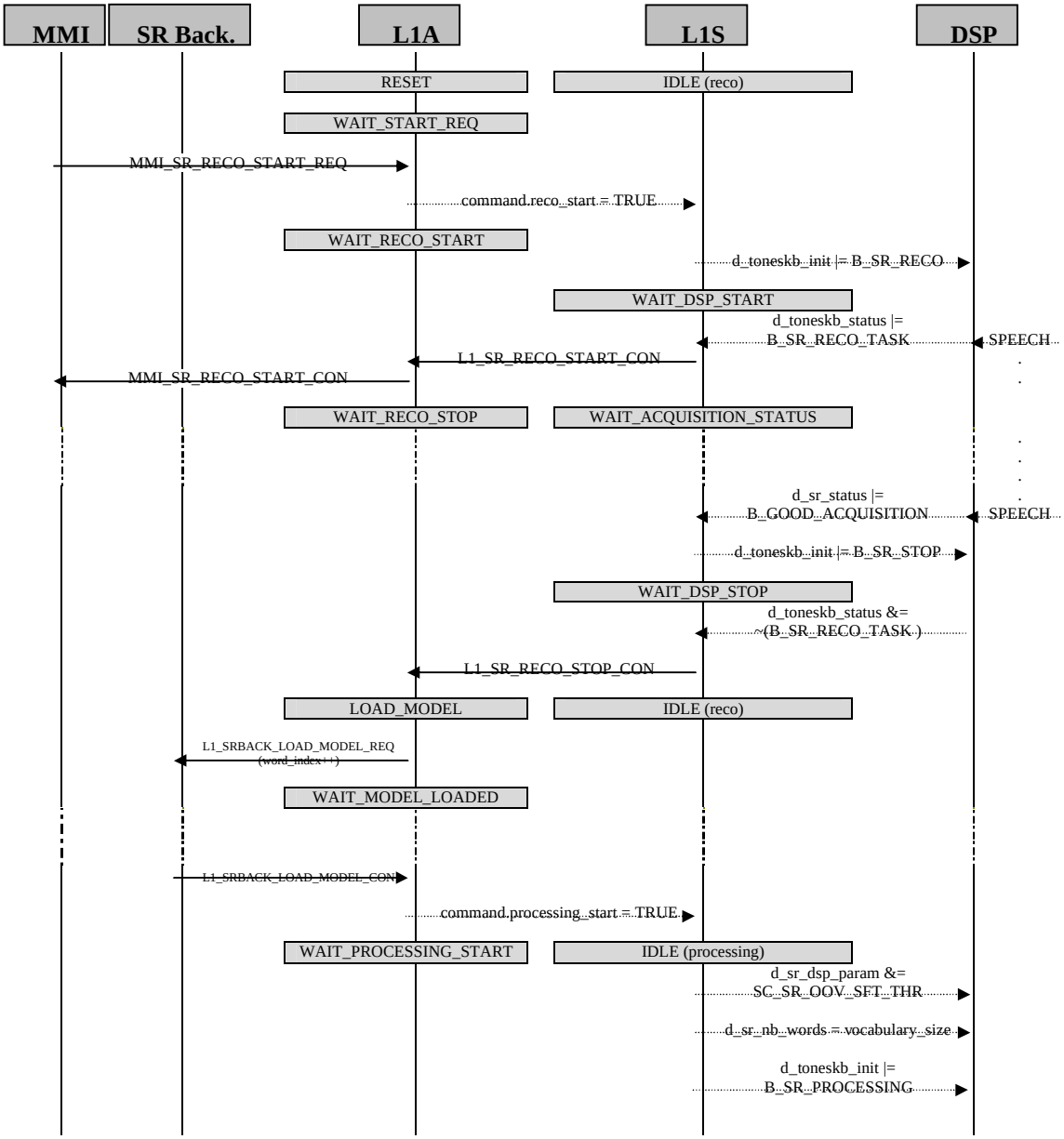


Figure 56: Reco task: reco no error and resquested stop (Background task is running).



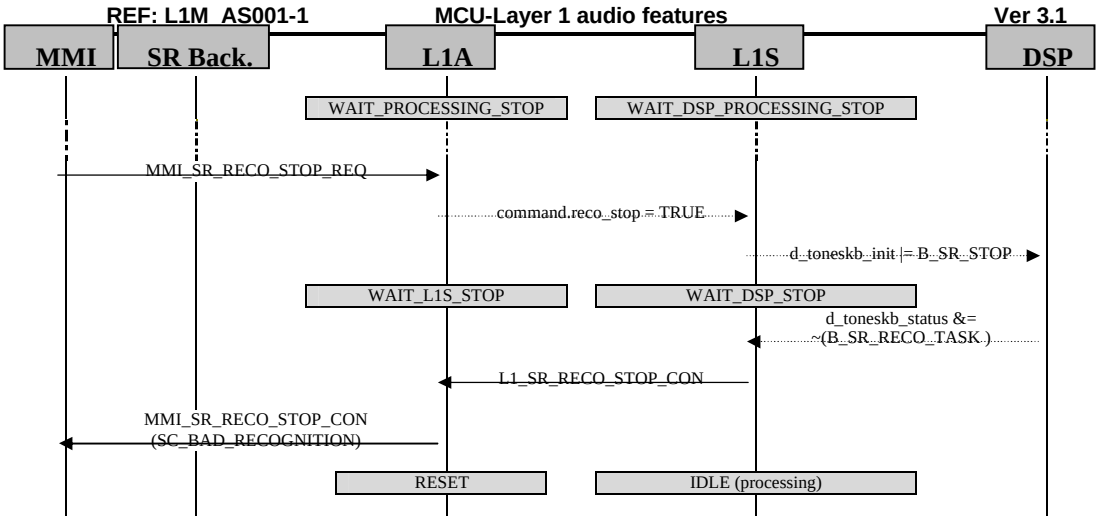


Figure 57: Reco task: reco no error and resquested stop (L1 is in processing phase).

9.6.5 L1A state machine

9.6.6 L1S state machine

9.7 Update-check task

The aim of this task is to update a model included in a MMI vocabulary database and to check if the updated model is close to the model to update.
Therefore, this task is composed by the update task followed by a recognition task.
Like the update task, the user can record again the speech sample of the word. This new speech samples replace the samples acquired during the enroll task or the update task. And, thanks to the voice memorization playing task, the user can hear the word that he pronounced.
If the updated model doesn't correspond to the model to update, this new model is ignored and not saved in the MMI vocabulary database and the speech samples too

9.7.1 MMI rules

The MMI must respect the following rules:

- When the MMI starts a speech recognition update-check task (MMI_SR_UPDATE_CHECK_START_REQ message), it must wait the start confirmation (MMI_SR_UPDATE_CHECK_START_CON) before to stop this task.
- The MMI isn't allowed to start a new speech recognition task before to receive a stop update-check confirmation message (MMI_SR_UPDATE_CHECK_STOP_CON).
- The MMI can receive a stop confirmation message (MMI_SR_UPDATE_CHECK_CON) in two cases:
  - The MMI requested to stop the current update-check task (i.e. MMI_SR_UPDATE_CHECK_STOP_REQ) and the L1 confirms with the stop confirmation message.
  - The current task is stopped automatically (end of the update-check or an error is occurred) and the L1 informs the MMI with the stop confirmation message.
- The MMI isn't allowed to run a speech recognition update-check task with:
  - A voice memorization playing and recording.
  - An another speech recognition tasks (enroll, update and recognition).
  - Melody 1.
  - Melody 0.
- With the DSP code 17, 31 and 32, the update-check task can't run in all GSM dedicated modes. And it can't run in all GPRS modes.
- With the DSP code 33, the update-check task can't run in all dedicated modes, except the dedicated signaling only mode. And it can run in all GPRS modes (if the DSP background task is present (c.f. spec. S924).
- The speech recognition update-check task can't run if the user database contains no word.
- The update-check task must have the same audio configuration (FIR coefficients…) than the enrollment task.
- The same audio configuration (FIR coefficients…) must be set for the update-check than the enroll task.
- The UL gain must be constant and the same than the enrollment task and not too high (no saturation).
- The index of the words included in the MMI database must start from 0 to the "*number of model*".
- For the DSP code 32, during a transition between the GSM and the GPRS, the update-check task is stopped.

9.7.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_SR_UPDATE_CHECK_START_REQ	T_MMI_SR_UPDATE_CHECK_REQ
MMI<-L1	MMI_SR_UPDATE_CHECK_START_CON	Trigger
MMI->L1	MMI_SR_UPDATE_CHECK_STOP_REQ	Trigger
MMI<-L1	MMI_SR_UPDATE_CHECK_STOP_CON	T_MMI_UPDATE_CHECK_STOP_CON

T_MMI_SR_UPDATE_CHECK_REQ message type
database_id (UWORD8) Specifies which user database is used.
word_index (UWORD8) Specifies the index in the *"database_id"* database of the model to update-check.
model_address (*UWORD16) Specifies the address where the model of the updated must be saved. Note: the size of this buffer is fixed by the value SC_SR_MMI_DB_MODEL_SIZE.
speech (BOOL) Specifies if the speech must be recorded or not.
speech_address (*UWORD16) Specifies the address of the working circular buffer used by the speech recording task. Note: the size of this buffer is fixed by the value SC_SR_MMI_2_L1_SPEECH_SIZE. Moreover, the size of this buffer is bigger than the size of the speech buffer for each word of the MMI vocabulary database: SC_SR_MMI_DB_SPEECH_SIZE < SC_SR_MMI_2_L1_SPEECH_SIZE.
vocabulary_size (UWORD8) Specifies the number of word included in the user database.

T_MMI_SR_UPDATE_CHECK_STOP_CON message type

error_id (UWORD8)
Specifies the type of error.

value	id	type
0	SC_NO_ERROR	No error is occurred.
1	SC_BAD_ACQUISITION	Bad acquisition of the word. The word is too long or too short.
2	SC_TIME_OUT	The time to acquire or to update or to check a word is too long.
3	SC_BAD_UPDATE	Bad update of the word.
4	SC_BAD_RECOGNITION	The updated word is out of vocabulary or the best words are too close.
5	SC_CTO_WORD	A word calculated by the CTO algorithm is the best word.
6	SC_CHECK_ERROR	The best word isn't the word to update.

best_word_index (UWOR16)
Gives the index in the user database of the recognized word.
Note: this value is valid only if the MMI doesn't stop the speech recognition task.
(useful for a customer OOV algorithm)

best_word_score (UWORD16)
Gives the scores of the recognized word. The DSP OOV algorithm calculated this score.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

second_best_word_index (UWOR16)
Gives the index in the user database of the second recognized word.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

second_best_word_score (UWORD16)
Gives the scores of the second recognized word. The DSP OOV algorithm calculated this score.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

third_best_word_index (UWOR16)
Gives the index in the user database of the third recognized word.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

third_best_word_score (UWORD16)
Gives the scores of the third recognized word. The DSP OOV algorithm calculated this score.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

fourth_best_word_index (UWOR16)
Gives the index in the user database of the third recognized word.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

fourth_best_word_score (UWORD16)
Gives the scores of the third recognized word. The DSP OOV algorithm calculated this score.
Note: this value is valid only if there's no error.
(useful for a customer OOV algorithm)

d_sr_db_level (UWORD16)
Gives the estimate voice level in dB. This information is used to tune the speech recognition task.
Note: this value is valid only if there's no error.
(useful to tune the speech recognition)

d_sr_db_noise (UWORD16)
Gives the estimate noise level in dB. This information is used to tune the speech recognition task.
Note: this value is valid only if there's no error.
(useful to tune the speech recognition)

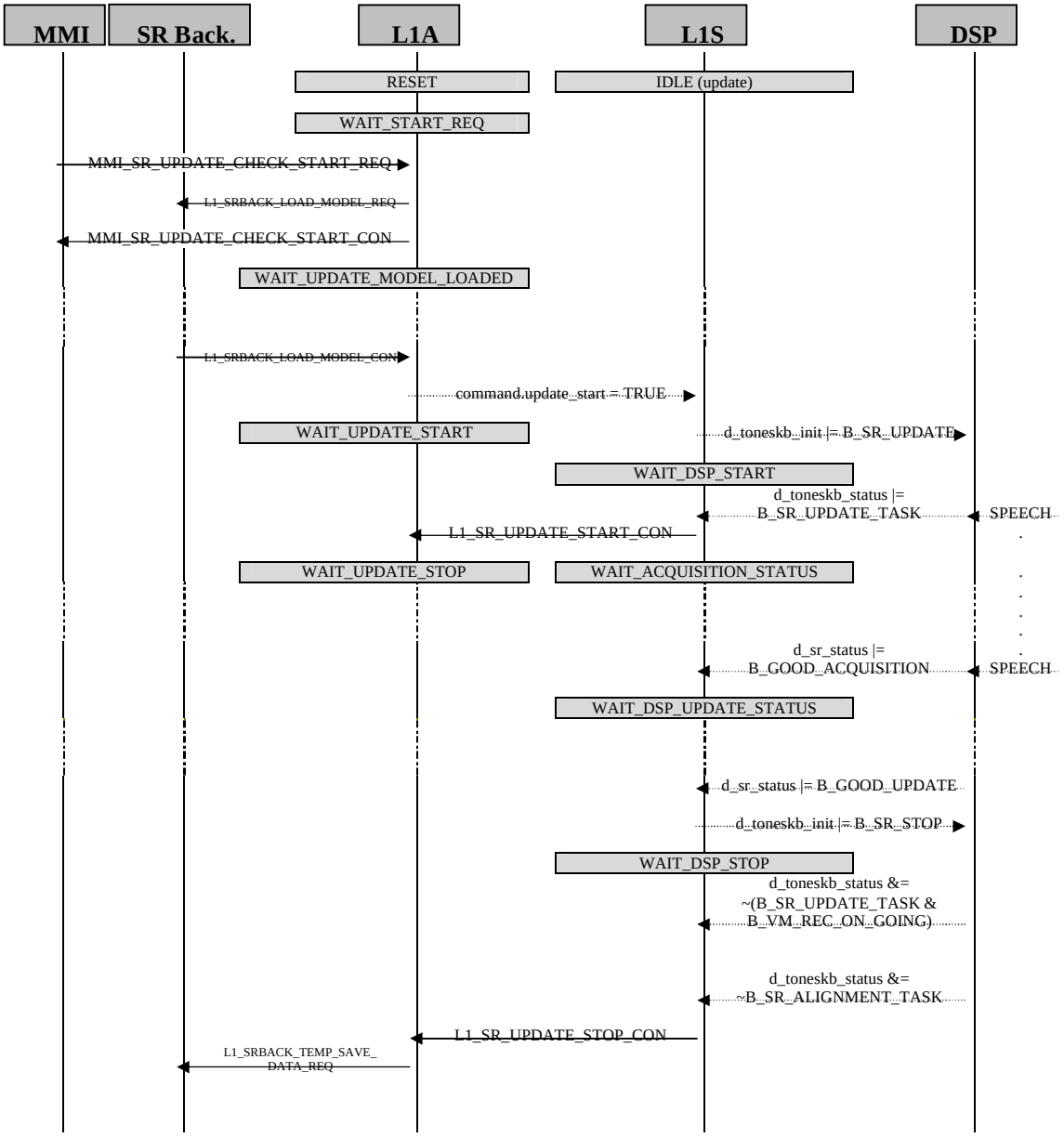
d_sr_model_size (UWORD16)
Gives the size of the model acquired during the speech recognition task. This information is used to tune the speech recognition task.
Note: this value is valid only if there's no error.
(useful to tune the speech recognition)

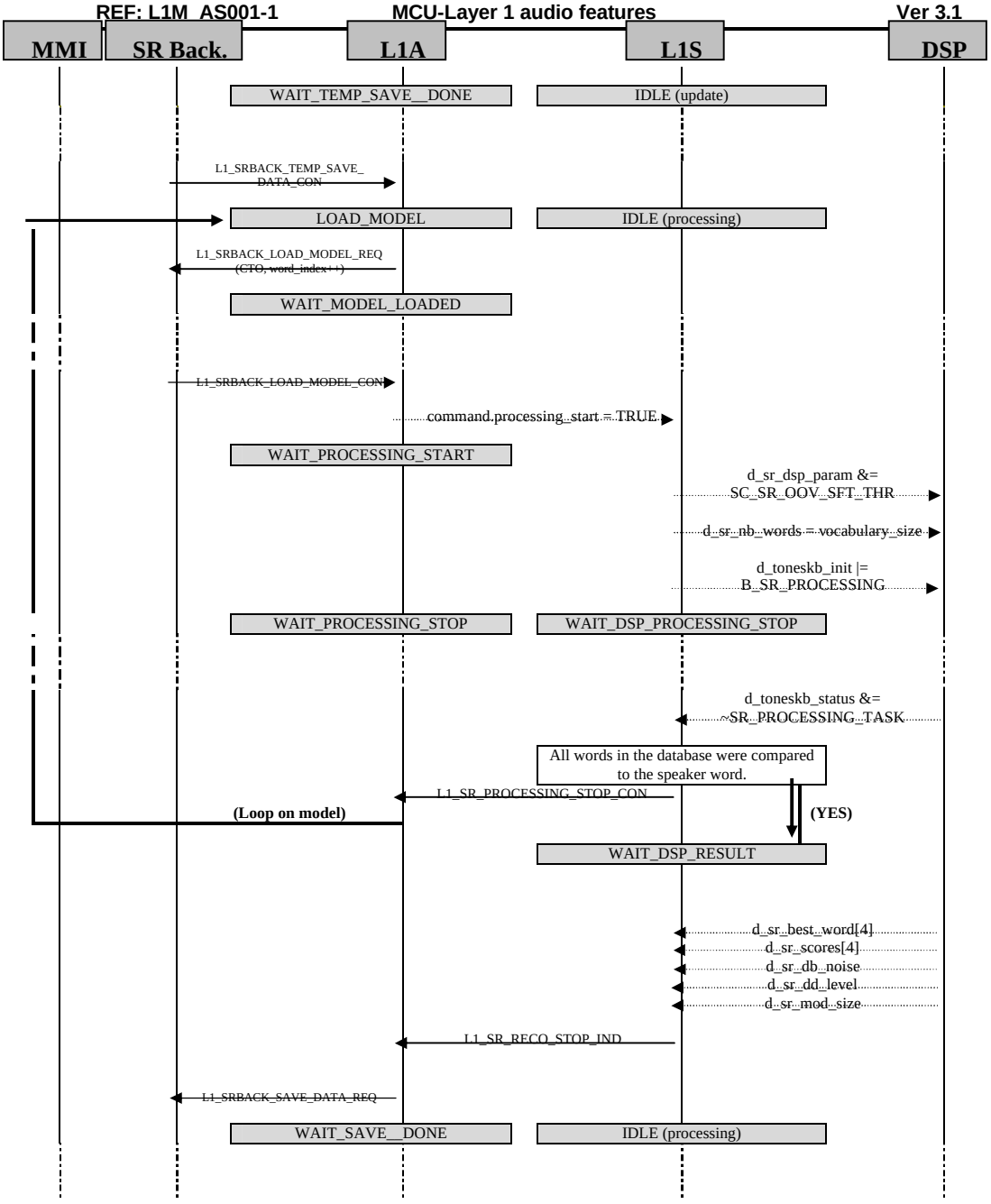
9.7.3 L1A-L1S message interface

For the structure of the message, please refer to the "reco task" and "update task" chapters.

Direction	Message name	Type
L1A<-L1S	L1_SR_UPDATE_START_CON	Trigger
L1A<-L1S	L1_SR_UPDATE_STOP_CON	T_L1_UPDATE_STOP_CON
L1A<-L1S	L1_SR_RECO_START_CON	Trigger
L1A<-L1S	L1_SR_RECO_STOP_CON	T_L1_RECO_STOP_CON
L1A<-L1S	L1_SR_PROCESSING_STOP_CON	T_L1_PROCESSING_STOP_CON
L1A<-L1S	L1_SR_RECO_STOP_IND	T_L1_RECO_STOP_IND

9.7.4 Process flow





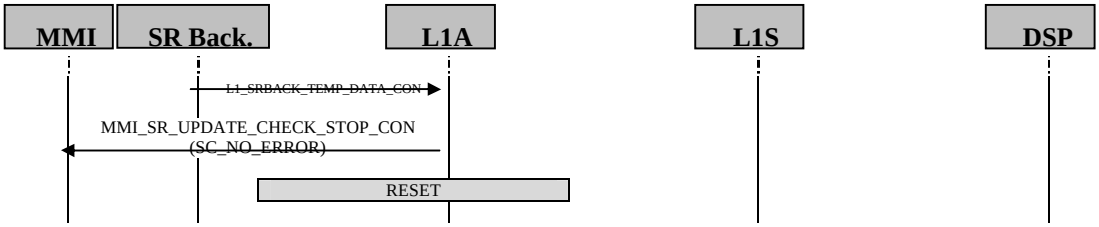


Figure 65: Update-check task: update-check with no error, no resquested stop.

9.7.5 L1A state machine

9.7.6 L1S state machine

C.f. L1S state machine of the update task and the L1S state machine of the processing task:

9.8 Speech recording task

The speech recording task is a task that runs in the same time than the acquisition phase of the enrollment, update and update-check task. This task is started if the MMI specified in the start message that the speech sample of the word to acquire must be recorded. So, this task doesn't have an MMI/L1 communication, it starts by thew L1A and stops automatically when the result of the speech recognition acquire phase is known (i.e. status = bad or good acquisition).
Note that the speech is recorded in a circular buffer specified by the MMI (i.e. speech_address message parameter).

9.8.1 Process Flow

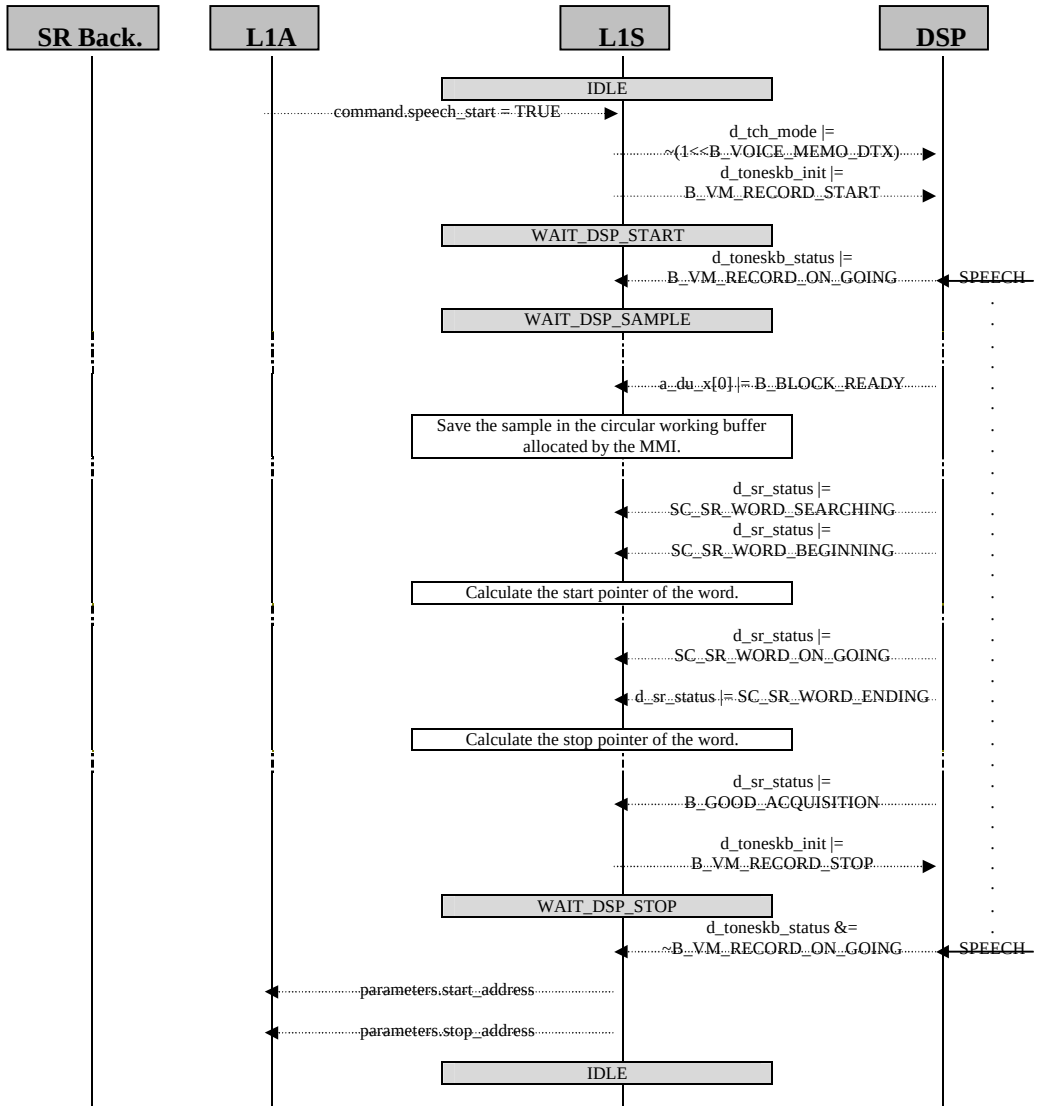


Figure 75: Speech recognition recording task: record speech with an automatic stop.

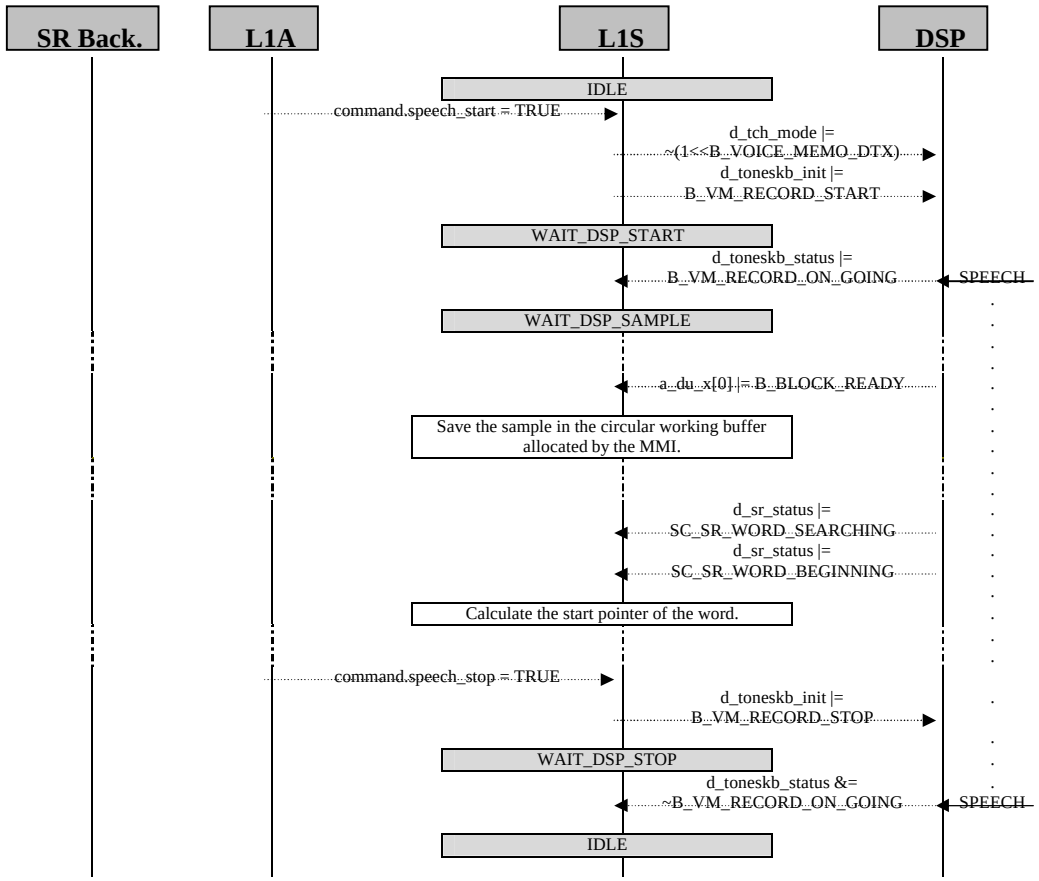


Figure 76: Speech recognition recording task: record speech with a requested stop.

9.8.2 L1S state machine

Figure 77: Speech recognition: L1S state machine speech recording task (1/2).

Figure 78: Speech recognition: L1S state machine speech recording task (2/2).

9.9 Speech recognition Background task

The background task is a preemptive task with the lowest priority. The aim of this task is to exchange the model between the DSP API memory and a customer vocabulary database put in Flash. And this task has an another role: to save the speech sample from the L1 buffer (allocated by the MMI) and a speech recognition dabase.
Note that: writing in flash is a customer dependencies therefore the background task calls some customer functions (c.f. included in the file l1audio_cust.c).

9.9.1 L1-Background interface

Direction	Message name	Type
L1A->BACK	**L1_SRBACK_SAVE_DATA_REQ**	**T_L1_SRBACK_SAVE_DATA**
L1A<-BACK	**L1_SRBACK_SAVE_DATA_CON**	**Trigger**
L1A->BACK	**L1_SRBACK_TEMP_SAVE_DATA_REQ**	**T_L1_SRBACK_TEMP_SAVE_DATA**
L1A<-BACK	**L1_SRBACK_TEMP_SAVE_DATA_CON**	**Trigger**
L1A->BACK	**L1_SRBACK_LOAD_MODEL_REQ**	**T_L1_SRBACK_LOAD_MODEL**
L1A<-BACK	**L1_SRBACK_LOAD_MODEL_CON**	**Trigger**

T_L1_SRBACK_SAVE_DATA message type
database_id (UWORD8) Specifies which user database is used.
model_index (UWORD8) Specifies the index of the model into the user database.
model_RAM_address (*API) Specifies the RAM address where the model is. Note: the size of the model is included in the header of the model.
speech (BOOL) Specifies if the speech was recorded or not.
start_buffer (*UWORD16) Specifies the start address of the RAM circular buffer that contains the speech samples. Note: this address is available only if the parameter *speech* = TRUE.
stop_buffer (*UWORD16) Specifies the stop address of the RAM circular buffer that contaisn the speech samples. Note: this address is available only if the parameter *speech* = TRUE.
start_address (*UWORD16) Specifies the start address of the recorded speech into the RAM buffer. Note: this address is available only if the parameter *speech* = TRUE.
stop_address (*UWORD16) Specifies the stop address of the recorded speech into the RAM circular buffer. Note: this address is available only if the parameter *speech* = TRUE.

T_L1_SRBACK_TEMP_SAVE_DATA message type
model_RAM_address_input (*API) Specifies the RAM address where the model is. Note: the size of the model is included in the header of the model.
model_RAM_address_output (*UWORD16) Specifies the RAM address where the model must be temporary saved

T_L1_SRBACK_LOAD_MODEL message type
database_id (UWORD8) Specifies which user database is used.
model_index (UWORD8) Specifies the index of the model into the user database.
model_RAM_address (*API) Specifies the RAM address where the model must be loaded
CTO_enable (BOOL) Specifies if the CTO algorithm is used (CTO_enable = TRUE) or not (CTO_enable = FALSE).

9.9.2 Process flow

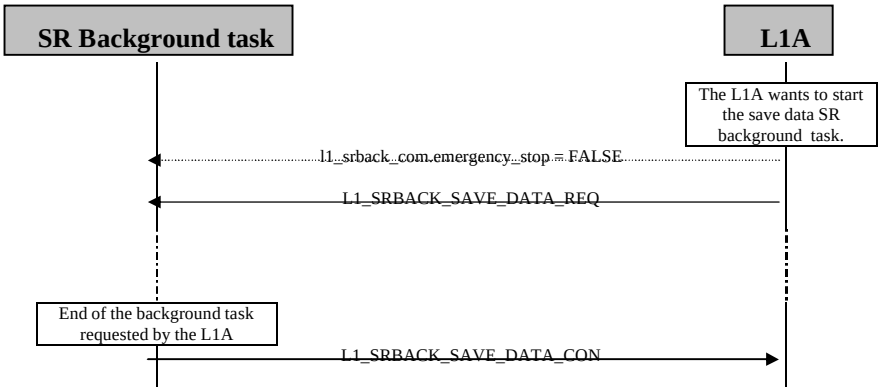


Figure 79: Background task: L1S/background task communication (without emercency stop)

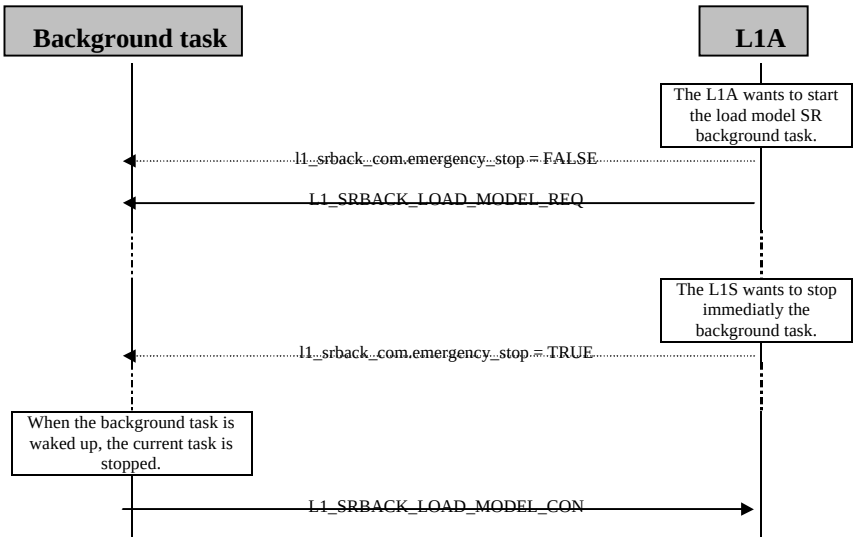


Figure 80: Background task: L1S/background task communication (with emergency stop)

9.9.3 Background state machine

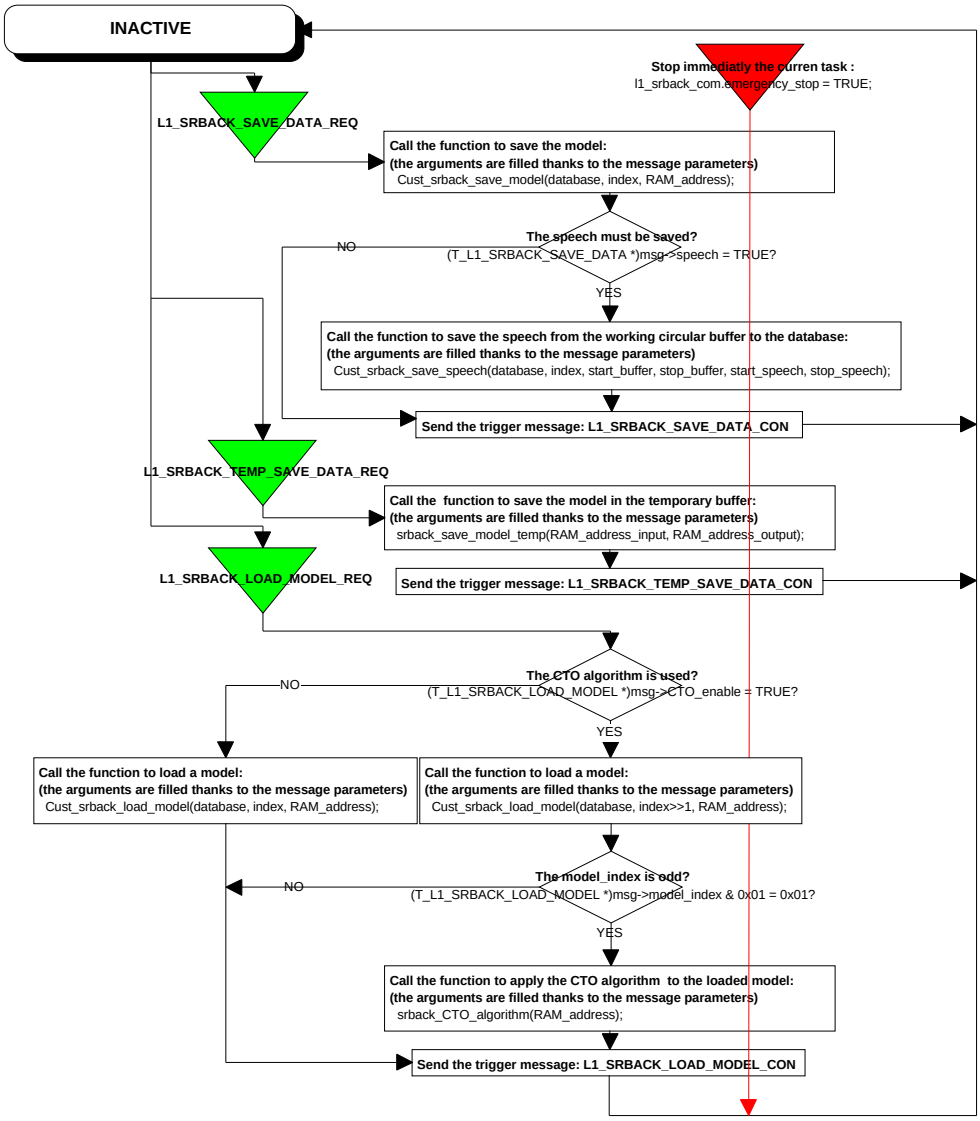


Figure 81: Background task: background state machine.

9.9.4 Called functions

Cust_srback_load_model(database, index, RAM_address) This customer function is used to load the model from the user database (specified by the arguments *database* and *index*) to the buffer (specified by the argument *address*). **Note: this function must immediately stop when an emergency stop is requested by L1**
database (UWORD8) Specifies which user database is used.
index (UWORD8) Specifies the index of the model into the user database.
RAM_address (*API) Specifies the RAM address where the model must be loaded. Note: the size of the model is included in the header of the model.

Cust_srback_save_model(database, index, RAM_address) This customer function is used to save the model from the RAM buffer (specified by the argument *RAM_address*) to the user database (specified by the arguments *database* and *index*). **Note: this function must immediately stop when an emergency stop is requested by L1.**
database (UWORD8) Specifies which user database is used.
index (UWORD8) Specifies the index of the model into the user database.
RAM_address (*API) Specifies the RAM address where the model must be loaded. Note: the size of the model is included in the header of the model.

srback_save_model_temp(RAM_address_input, RAM_address_ouput)
This function is used to save the model from the API memory (specified by the argument *RAM_address_input*) to the memory (specified by the argument *RAM_address_ouput*).
Note: this function must immediately stop when an emergency stop is requested by L1S.

RAM_address_input (*API)
Specifies the RAM memory where the model is.

RAM_address_output (*UWORD16)
Specifies the address in the API memory where the model must be temporary saved.
Note: the size of the model is included in the header of the model.

srback_CTO_algorithm(RAM_address)
This function applies the CTO algorithm to the model saved in a RAM buffer (specified by the argument *RAM_address*).
Note: this function must immediately stop when an emergency stop is requested by L1S.

RAM_address (*API)
Specifies the address where the model is.

Cust_srback_save_speech(database, index, start_buffer, stop_buffer, start_speech, stop_speech)
This customer function is used to save the speech (specified by the argument *start_speech* and *stop_speech*) from the circular buffer (specified by the argument *start_buffer* and *stop_buffer*) to the user database (specified by the arguments *database* and *index*). And this function adds the end voice memorisation mask (i.e. SC_VM_END_MASK) after the last speech samples put in the database.
Note: this function must immediately stop when an emergency stop is requested by L1S.

database (UWORD8)
Specifies which user database is used.

index (UWORD8)
Specifies the index in user database of the speech.

start_buffer (*UWORD16) Specified the start address of the circular buffer that contains the speech samples. **Note: the pointed value is included in the buffer. It's the first word of the buffer.**
stop_buffer (*UWORD16) Specified the end address of the circular buffer that contains the speech samples. **Note: the pointed value isn't included in the buffer. It's the first memory after the last word of the buffer.**
start_speech (*UWORD16) Specified the start address of the speech samples contained in the circular buffer. **Note: the pointed value is included in the buffer. It's the first sample of the word.**
stop_speech (*UWORD16) Specified the end address of the speech samples contained in the circular buffer. **Note: the pointed value isn't included in the buffer. It's the next value after the last sample of the word.**

So, we can represent the roles of the *Cust_srback_save_speech* function by the following pictures:

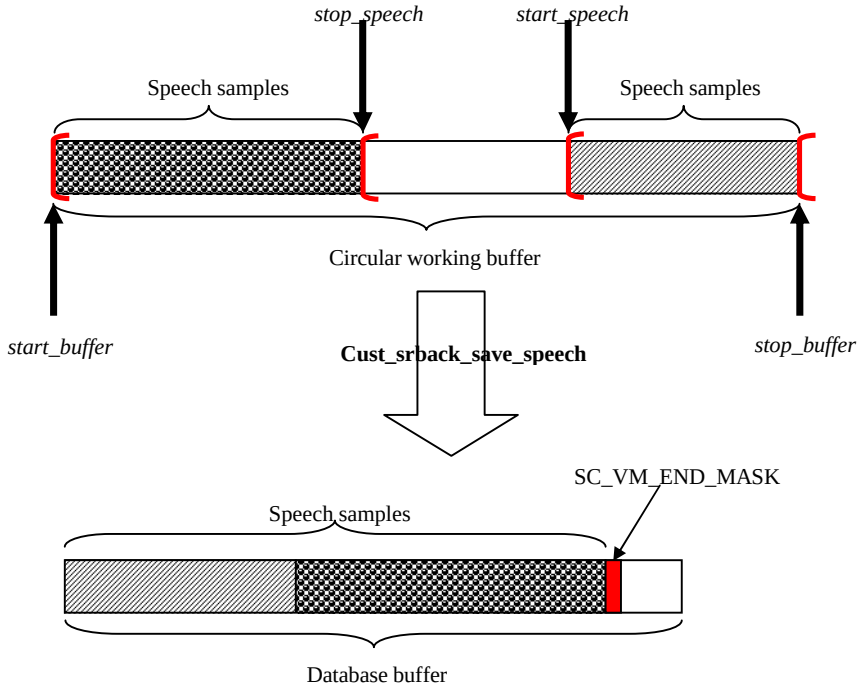


Figure 82: Background task: Cust_srback_save_speech function.

10 AEC

The AEC feature is the task to enable/disable and to tune the Echo canceller included in the audio path. In DSP Codes >= 33 (patch >= 2100), 34, 35 and 36, it has been replaced by the New AEC feature and can no longer be used.

Locosto TCS3.2 doesn't support this feature.

10.1 MMI rules

The MMI must respect the following rules:

- The MMI can't request a new AEC setting (MMI_AEC_REQ) before to receive the previous update indication message (MMI_AEC_CON) of the last AEC setting.
- The MMI is allowed to request an AEC setting (MMI_AEC_REQ) in all modem mode.

10.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	**MMI_AEC_REQ**	**T_MMI_AEC_REQ**
MMI->L1	**MMI_AEC_CON**	**Trigger**

T_MMI_AEC_REQ message type

aec_control (UWORD16)
The bit field register is used to configure the echo canceller (AEC), the speech enhancement control (SPENH) and noise suppression algorithms:

Bit number	Rule		
0	Not used.		
1	1: enable the AEC algorithm, 0: disable the AEC algorithm.		
2	1: enable the SPENH algorithm, 0: disable the SPENH algorithm.		
3-4	Set the additional echo suppression level:	00	0 dB
		01	6 dB
		10	12 dB
		11	18 dB
5-6	Set the noise suppression level limitation:	00	none
		01	-6 dB
		10	-12 dB
		11	-18 dB
7	1: Reset the AEC algorithm, 0: no action.		
8	1: Reset the SPENH algorithm, 0: no action.		

10.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	L1_AEC_CON	Trigger

10.4 Process flow

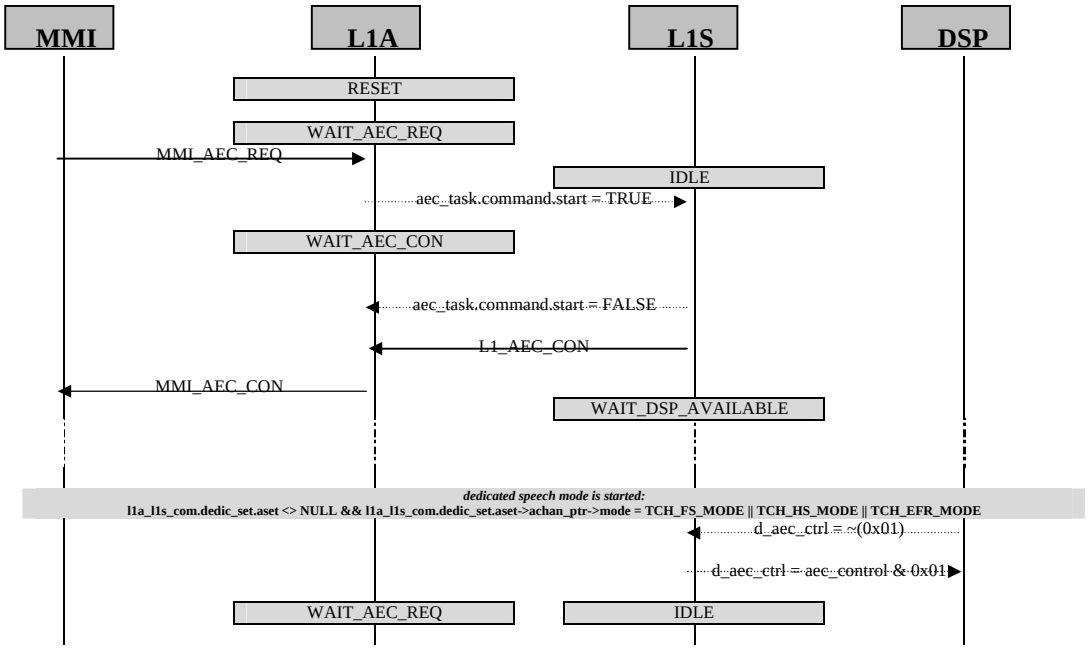


Figure 83: AEC with one request in idle mode.

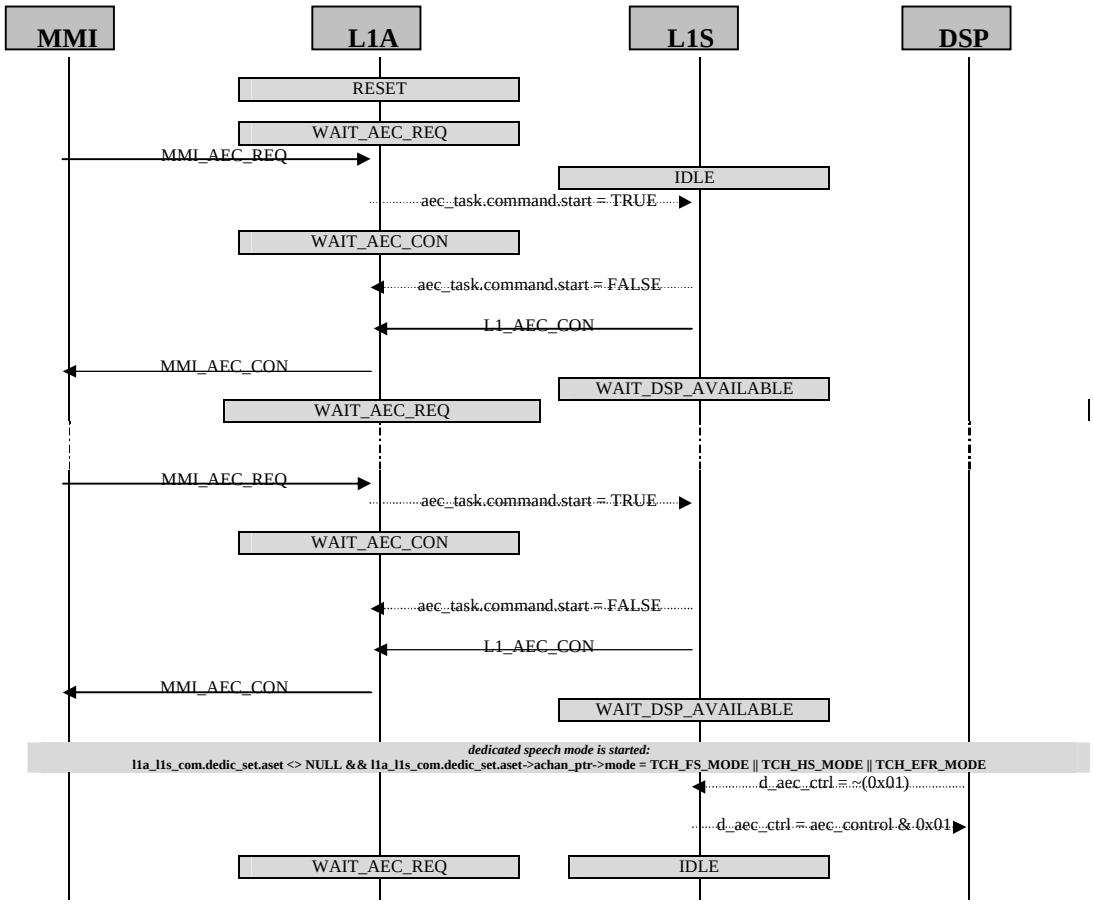


Figure 84: AEC with two requests in idle mode.

10.5 L1A state machine

Figure 85: AEC L1A state machine

10.6 L1S state machine

Figure 86: AEC L1S state machine

11 New AEC

The new AEC feature is the task to enable/disable and to tune the Echo canceller included in the audio path. It is an improvement that gives more visibility and tuning possibilities. It is available in DSP codes 33 (patch >= 2100), 34, 35 and 36 and replaces previous AEC algorithm, which can no longer be used.
AEC is enabled through L1_NEW_AEC compilation flag.
Overview and method to tune the new parameters are explained in [4].

11.1 MMI rules

11.1.1 AEC rules

The MMI must respect the following rules:

- The MMI can't request a new AEC setting (MMI_AEC_REQ) before to receive the previous update indication message (MMI_AEC_CON) of the last AEC setting.
- The MMI is allowed to request an AEC setting (MMI_AEC_REQ) in all modem mode.

11.1.2 Visibility

- Visibility is set through a bit of aec_control (see below). It has been designed for **debugging** purposes only. A message is sent to layer1 and traced every **SC_AEC_VISIBILITY_INTERVAL** frames (defined in l1_audio_cust.h).
- When visibility is enabled, it can only be disabled by a new AEC request, i.e. when the modem goes from dedicated mode to idle, visibility will still be enabled when going back to dedicated mode (even if there was no AEC request).

11.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	**MMI_AEC_REQ**	**T_MMI_AEC_REQ**
MMI->L1	**MMI_AEC_CON**	**Trigger**

T_MMI_AEC_REQ message type

aec_control (UWORD16)
The bit field register is used to configure the echo canceller (AEC), the speech enhancement control (SPENH) and noise suppression algorithms:

Bit number	Rule
0	MCU/DSP Ack.
1	1: enable the AEC algorithm, 0: disable the AEC algorithm.
2	1: enable the SPENH algorithm, 0: disable the SPENH algorithm.
3-4	Unused
5-6	Set the noise suppression level limitation: 00 None

01 -6 dB 10 -12 dB 11 -18 dB 7 1: Reset the AEC algorithm, 0: no action. 8 1: Reset the SPENH algorithm, 0: no action. 9 1: Enable AEC visibility, 0: disable AEC visibility.
cont_filter (UWORD16) Enable (1) or disable (0) continuous mode filtering
granularity_att (UWORD16) granularity of the smoothed attenuation
coef_smooth (UWORD16) smoothing coefficient
es_level_max (UWORD16) maximum attenuation level
fact_vad (UWORD16) VAD factor relative to the current estimated energy
thrs_abs (UWORD16) VAD absolute offset relative to the current estimated energy
fact_asd_fil (UWORD16) modifying factor of d_far_end_noise for filtering decision
fact_asd_mut (UWORD16) modifying factor of d_far_end_noise for muting decision

11.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	L1_AEC_CON	Trigger
L1A<-L1S	L1_AEC_IND	T_L1_AEC_IND

T_L1_AEC_IND message type
es_level (UWORD16) current attenuation
far_end_pow (UWORD32) current estimate of the far end energy
far_end_noise (UWORD32) current estimate of the far end noise

11.4 Process flow

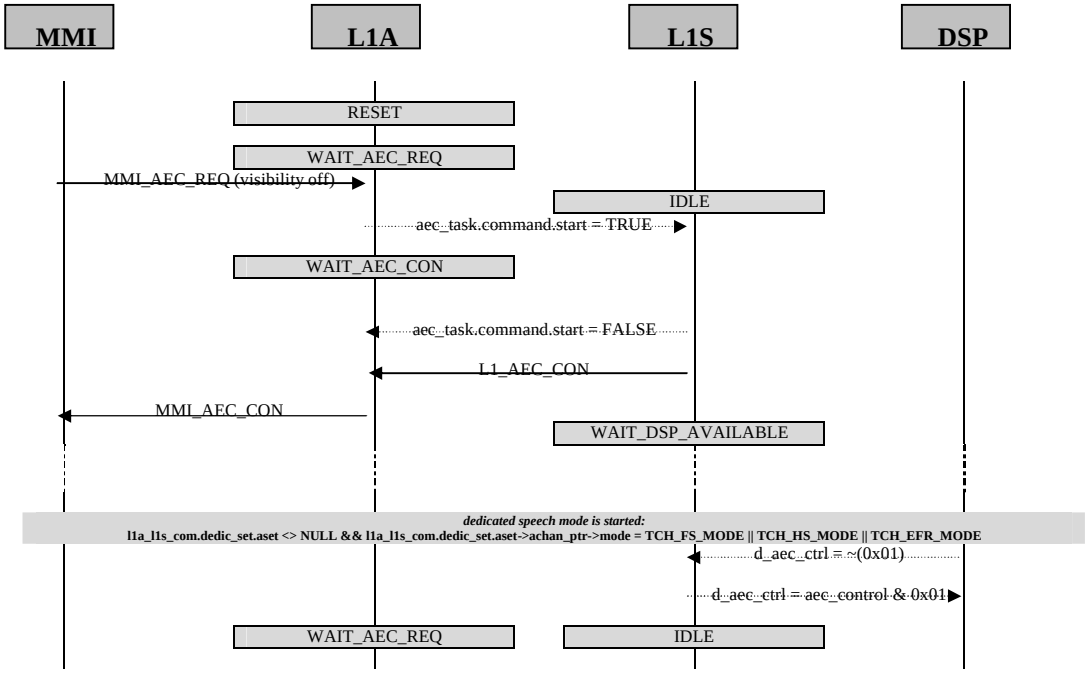


Figure 87: AEC with one request in idle mode, no AEC visibility

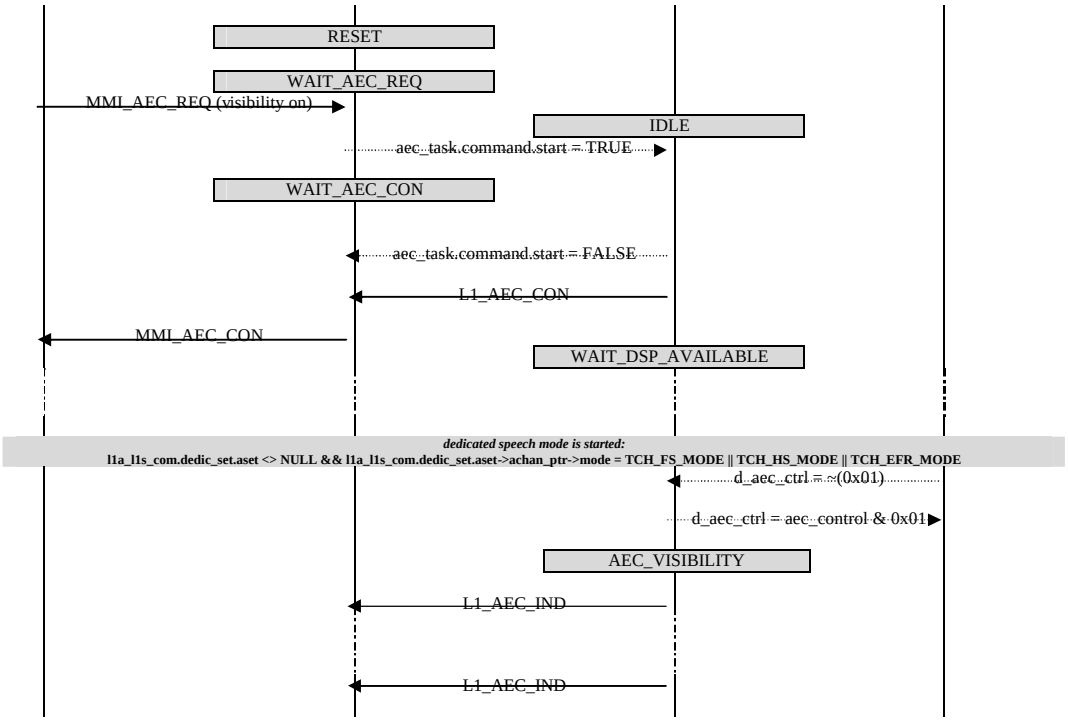


Figure 88: AEC with one request in idle mode, AEC visibility

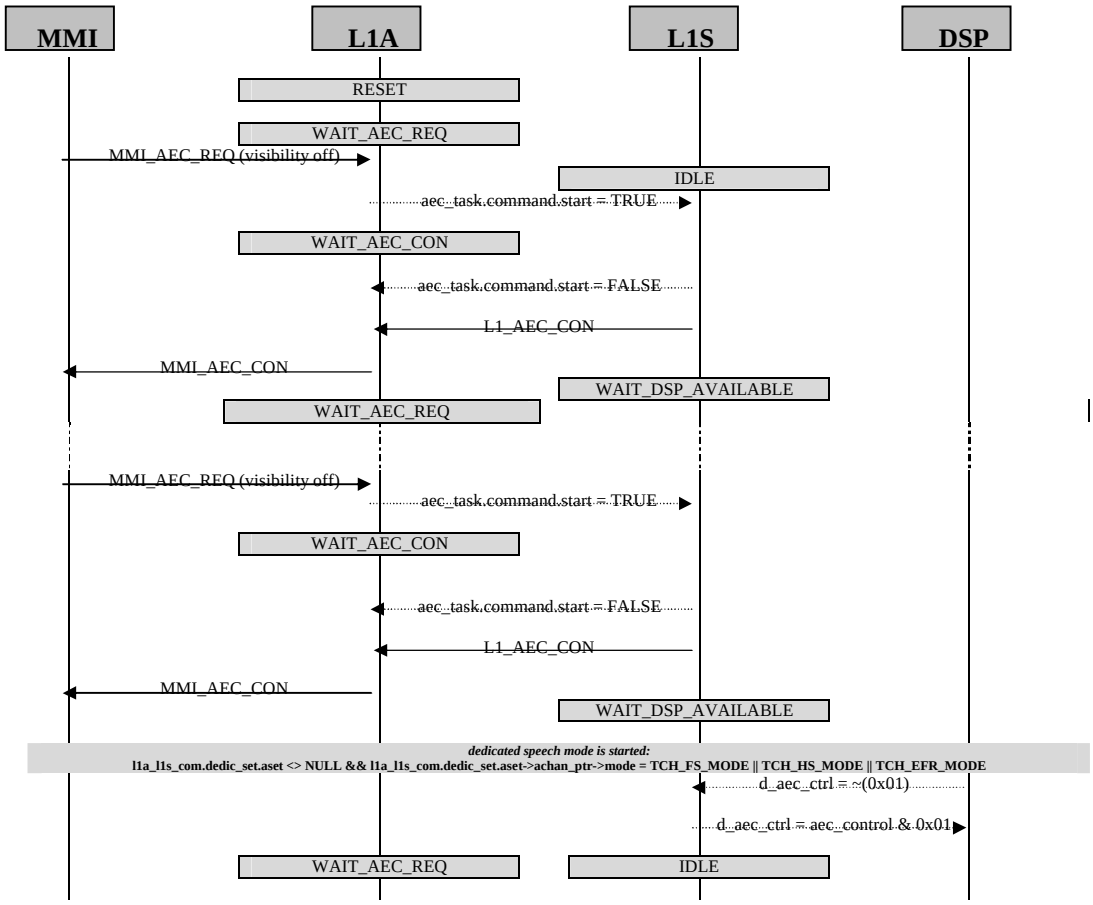


Figure 89: AEC with two requests in idle mode, no visibility

11.5 L1A state machine

Figure 90: AEC L1A state machine

11.6 L1S state machine

Figure 91: AEC L1S state machine

12 FIR configuration

12.1 MMI rules

The MMI must respect the following rules:

- The MMI can't request a new FIR configuration (MMI_FIR_CFG_REQ) before to receive the previous confirmation message (MMI_FIR_CFG_CON) of the last configuration.
- For the DSP code 32, the MMI is allowed to use the FIR task in all GSM modes but not in all GPRS modes.
- For the DSP code 33, the MMI is allowed to use the FIR task in all GSM modes and in all GPRS modes.

Note:
We can say that, for instance, the DSP/L1 communication for the FIR isn't really robust. Indeed, the FIR coefficient puts in the API are used directly by the DSP. So, when the MCU changes the coefficient in the same time than DSP used it, the FIR isn't really correct.

Locosto TCS3.2 doesn't support the FIR in downlink.

12.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	MMI_AUDIO_FIR_REQ	T_MMI_AUDIO_FIR_REQ
MMI->L1	MMI_AUDIO_FIR_CON	Trigger.

T_MMI_AUDIO_FIR_REQ message type

fir_loop (BOOL)
This flag indicates if the internal audio loop with the FIR filter must be closed.
0 No loop.
1 Samples from the microphone are looped back to the loudspeaker.
This loop can be used to determine the FIR coefficients.

update_fir (UWORD8)
Indicates which FIR must be updated

Value	Name	Indication
1	DL_FIR	DL FIR must be updated.
2	UL_FIR	UL FIR must be updated.
3	UL_DL_FIR	UL&DL FIR must be updated.

fir_ul_coefficient (*UWORD16)
Pointer to the 31 coefficients of the uplink path FIR.
The format of each coefficient is F2.14. Moreover the coefficients are signed, the MSB indicates the signed (1:-, 0:+).
For example: 0,5 = 0x2000, 1 = 0x4000 , –1=0xc000 and -0.5=-0x2000=-(0010 0000 0000 0000)= NOT(0010 0000 0000 0000)+1 (it's the modulo 2 addition)= 1101 1111 1111 1111+1=1110 0000 0000 0000 = 0xE000.

fir_dl_coefficient (*UWORD16)
Pointer to the 31 coefficients of the downlink path FIR.
The format of each coefficient is F2.14. Moreover the coefficients are signed, the MSB indicates the signed (1:-, 0:+).
For example: 0,5 = 0x0200, 1 = 0x4000 and –1=0xc000.

12.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	L1_AUDIO_FIR_CON	Trigger.

12.4 Process flow

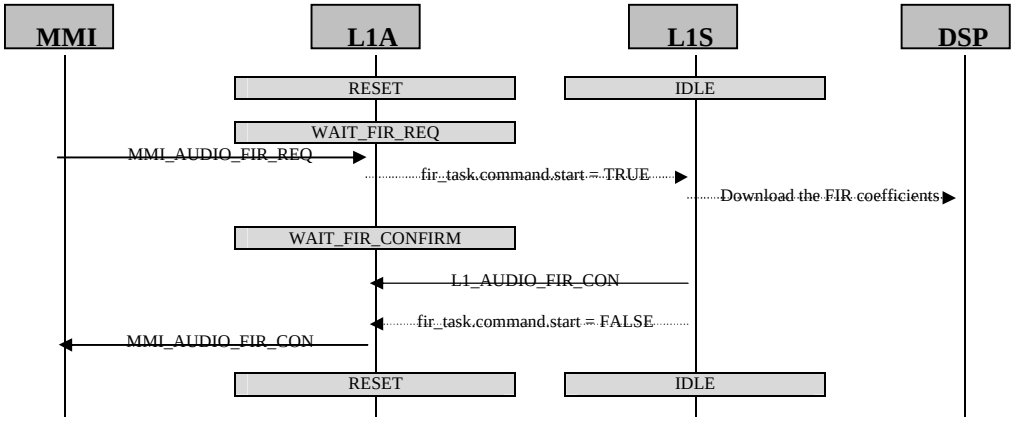


Figure 92: FIR process flow.

12.5 L1A state machine

Figure 93: FIR L1A state machine.

12.6 L1S state machine

Figure 94: FIR L1S state machine.

–

13 Audio mode configuration

13.1 MMI rules

The MMI must respect the following rules:

- The MMI can't request a new mode configuration (MMI_AUDIO_MODE_REQ) before to receive the previous confirmation message (MMI_AUDIO_MODE_CON) of the last configuration.
- This feature is only available in DSP code 33 with a bluetooth device.

13.2 MMI-L1 interface

Direction	Message name	Type
MMI->L1	**MMI_AUDIO_MODE_REQ**	**T_MMI_AUDIO_MODE**
MMI->L1	**MMI_AUDIO_MODE_CON**	**Trigger.**

T_MMI_AUDIO_MODE message type

audio_mode (UWORD8)
Indicates which audio mode must be applied.

Value	Name	Indication
0	GSM_ONLY	GSM normal mode.
1	BT_CORDLESS	Bluetooth cordless mode.
2	BT_HEADSET	Bluetooth headset mode.

For information:

- **GSM normal mode**: voice samples are exchanged between GSM network and the analog base band (microphone and speaker). The path is between the demodulated samples of the GSM network and the VSP of the analog base band.
- **Bluetooth Cordless**: voice samples are exchanged between analog base band (microphone and speaker) and Bluetooth CODEC interfaces: The path is between Voice Serial Port of analog base band and Multi-Channel Serial Interface of bluetooth device.
- **Bluetooth Headset**: voice samples are exchanged between GSM network and Bluetooth CODEC interfaces. The path is between the demodulated samples of the GSM network and the Multi-Channel Serial Interface of the bluetooth device.

13.3 L1A-L1S message interface

Direction	Message name	Type
L1A<-L1S	**L1_AUDIO_MODE_CON**	**Trigger.**

13.4 Process flow

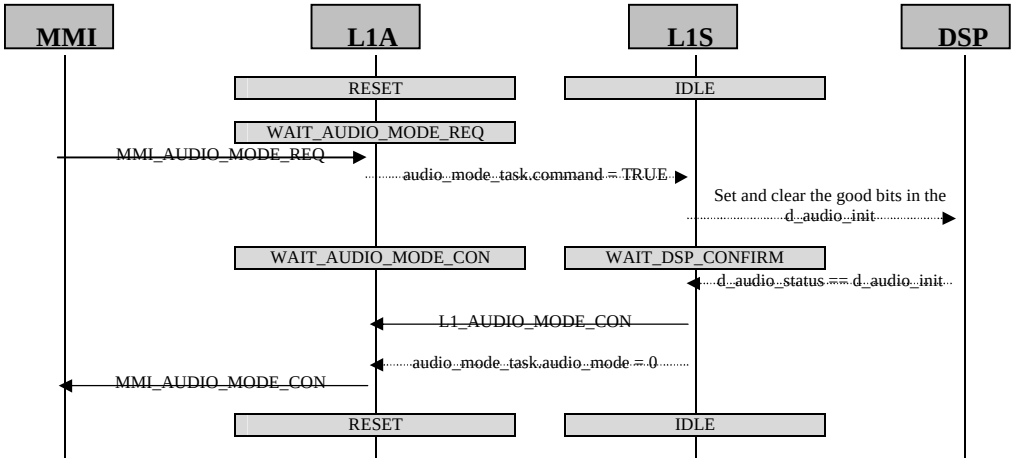


Figure 95: Audio mode process flow.

13.5 L1A state machine

Figure 96: Audio mode L1A state machine.

13.6 L1S state machine

Figure 97: Audio mode L1S state machine.

14 Vocoder mute functions

These two following functions are use to mute/unmute the uplink or the downlink vocoder during the dedicated mode. The MMI can call directly these drivers at any time.
In fact, the" vocoder is mute" means that the samples from the RF is clear and the sample from the earphone is clear too. Only the samples from the audio tasks are valid.
This driver function are active only in TCH speech mode.

Locosto TCS3.2 doesn't support this feature.

14.1 vocoder_mute_dl

This function is used to mute the DSP downlink vocoder. Its prototype is:

VOID vocoder_mute_dl(BOOL mute)

If *mute* equals TRUE, the downlink vocoder is muted and if *mute* equals FALSE the downlink vocoder is unmuted.
This function can be called by the MMI at anytime.

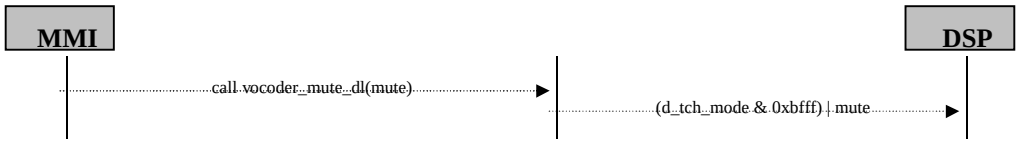


Figure 98: vocoder_mute_dl process.

14.2 vocoder_mute_ul

This function is used to mute the DSP uplink vocoder. Its prototype is:

VOID vocoder_mute_ul(BOOL mute)

If *mute* equals TRUE, the uplink vocoder is muted and if *mute* equals FALSE the uplink vocoder is unmuted.
This function can be called by the MMI at anytime.

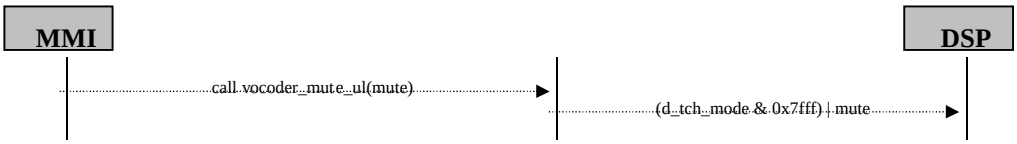


Figure 99: vocoder_mute_ul process.

15 Analogue Base Band audio drivers

This document describes API for NAUSICA, IOTA and SYREN ABBs. Audio functions such as volume control and side tone control are implemented by means of function calls.
There are three types of functions:

- The first type of functions is called by the MMI for voice volume control: *ABB_DlVolume(), ABB_DlMute(), ABB_UlMute, ABB_SideTone()*.
- The second type of functions is used for calibration of uplink and downlink PGA gain: *ABB_CAL_UlVolume(), ABB_CAL_DlVolume().*
- The third type of functions is used to configure the Analog Base Band voice and audio path: *ABB_Audio_Config, ABB_Audio_Config_2, ABB_Audio_Control, ABB_Audio_On_Off, ABB_Audio_Volume, ABB_Audio_PLL, ABB_Audio_VBPop, ABB_Audio_Delay_Init.*

The audio functions concerning the voice uplink and downlink paths are implemented according to the Analogue Base Band specification.

Locosto TCS3.2 doesn't support this feature.

	NAUSICA	IOTA	SYREN
ABB_DlVolume	X	X	X
ABB_DlMute	X	X	X
ABB_UlMute	X	X	X
ABB_SideTone	X	X	X
ABB_CAL_UlVolume	X	X	X
ABB_CAL_DlVolume	X	X	X
ABB_Audio_Config	X	X	X
ABB_Audio_Config_2		X	X
ABB_Audio_Control			X
ABB_Audio_On_Off			X
ABB_Audio_Volume			X
ABB_Audio_PLL			X
ABB_Audio_VBPop			X
ABB_Audio_Delay_Init			X

Table 1: API functions available for an ABB

15.1 ABB_DlVolume

This function controls the voice downlink path. The function prototype is:

void ABB_DlVolume (UWORD8 volume_index)

The range of the input parameter **volume_index** is 0 to 255. The downlink path can be set into mute if the **volume-**index is equal to 0. The volume of the speaker is controlled by the input parameter ranging from 1

to 255, where 1 corresponds to the minimum volume (-24 dB) and 255 to the maximum volume (0 dB). The volume, set during the power on phase, is contained in the constant C_VBCR in the file l1_rf$(RF).h. This constant contains the value set to the Analog Base Band VBDR register at power on.
This function can be called by the MMI at anytime.

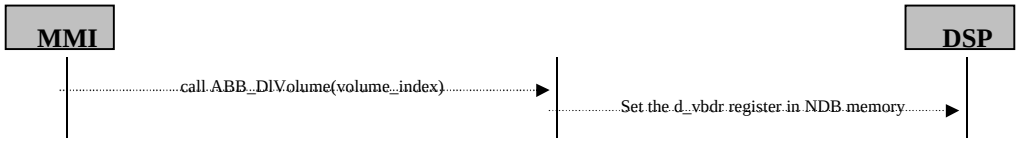


Figure 100: ABB_DlVolume process.

15.2 ABB_DlMute

By executing this function the speaker can be set to mute. The function prototype is:

void ABB_DlMute (BOOL mute)

When the input parameter **mute** is set to TRUE, the downlink path is set to mute. The volume applied to the speaker prior to switching to mute state is memorized, so that the same volume is output to the speaker after calling the function ABB_DlMute() with **mute**=FALSE.
This function can be called by the MMI at anytime.

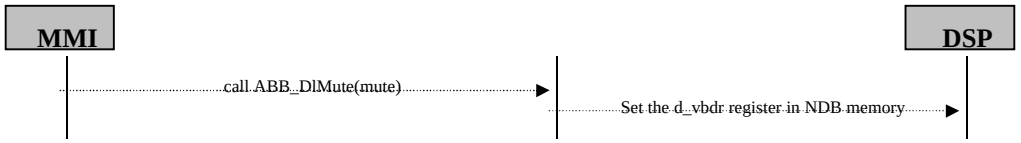


Figure 101: ABB_DlMute process.

15.3 ABB_SideTone

This function controls the audio side tone gain. The function prototype is:

void ABB_SideTone (UWORD8 volume_index)

The input parameter **volume_index** ranges from 0 to 255. When **volume_index** equals zero the side tone gain is set to mute. 1 corresponds to the minimum side tone gain (-23 dB) and 255 correspond to the maximum side tone gain (1 dB).
This function can be called by the MMI at anytime.

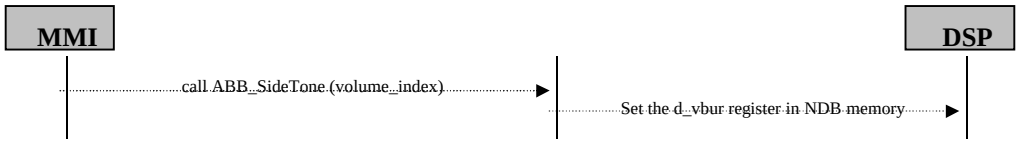


Figure 102: ABB_Sidetone process.

15.4 ABB_CAL_UlVolume

This function is provided to calibrate the voice uplink path. The function prototype is:

void ABB_CAL_UlVolume (UWORD8 pga_index)

The range of the input parameter **pga_index** is 0 to 24. The parameter determines the gain applied to the microphone, where 0 corresponds to the minimum volume (-12 dB) and 24 to the maximum volume (12 dB), see the table below for the detail of the dB value per **pga_index** value. Once the PGA gain has been tuned it is kept as a fixed constant in the C_VBUR file l1_rf$(RF).h that is used during the initialization of the Analogue Base Band registers. The constant is set to the value given Analog Base Band specification (register VBUR).

pga_index	VULPG 4	VULPG 3	VULPG 2	VULPG 1	VULPG 0	Relative Gain
0	1	0	0	0	0	-12 dB
1	1	0	1	1	1	-11 dB
2	1	1	0	0	0	-10 dB
3	1	1	0	0	1	-9 dB
4	1	1	0	1	0	-8 dB
5	1	1	0	1	1	-7 dB
6	0	0	0	0	0	-6 dB
7	0	0	0	0	1	-5 dB
8	0	0	0	1	0	-4 dB
9	0	0	0	1	1	-3 dB
10	0	0	1	0	0	-2 dB
11	0	0	1	0	1	-1 dB
12	0	0	1	1	0	0 dB
13	0	0	1	1	1	+1 dB
14	0	1	0	0	0	+2 dB
15	0	1	0	0	1	+3 dB
16	0	1	0	1	0	+4 dB
17	0	1	0	1	1	+5 dB
18	0	1	1	0	0	+6 dB
19	1	0	0	0	1	+7 dB
20	1	0	0	1	0	+8 dB
21	1	0	0	1	1	+9 dB
22	1	0	1	0	0	+10 dB
23	1	0	1	0	1	+11 dB
24	1	0	1	1	0	+12 dB

Figure 103: Uplink PGA gain per pga_index.

This function can be called by the MMI at anytime.

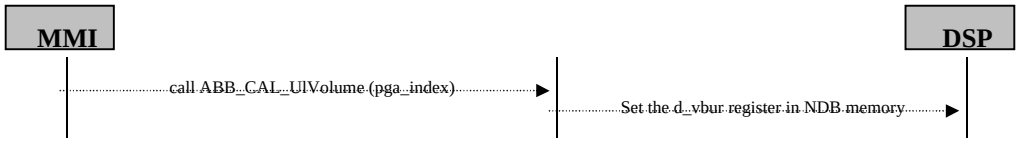


Figure 104: ABB_CAL_UlVolume process.

15.5 ABB_CAL_DlVolume

This function is provided to adjust the gain of the voice downlink path. The function prototype is:

void ABB_CAL_DlVolume (UWORD8 volume_index, UWORD8 pga_index)

The input parameter volume index is the same as specified for the function *ABB_DlVolume()*. The range of the input parameter **pga_index** is 0 to 12. The parameter determines the gain correction used in the amplifier stages in order to achieve the volume control gain (cf. Analogue Base Band specification, register VBDR) at the speaker output. Table below gives the downlink PGA gain. The parameter **pga_index** corresponds to the table index. The PGA gain determined during calibration phase is used for the initialization of the Analogue Base Band registers VBDR. This is achieved by setting the constant C_VBDR, file l1_rf$(RF).h to the required value.
Remark: This function may be used for volume control using same steps than the function ABB_DlVolume(). This is achieved by using different values of **pga_index** for a given volume which results in a combination of volume control gain setting and downlink PGA gain.

pga_index	VDLG3	VDLG2	VDLG1	VDLG0	Relative Gain
0	0	0	0	0	-6 dB
1	0	0	0	1	-5 dB
2	0	0	1	0	-4 dB
3	0	0	1	1	-3 dB
4	0	1	0	0	-2 dB
5	0	1	0	1	-1 dB
6	0	1	1	0	0 dB
7	0	1	1	1	+1 dB
8	1	0	0	0	+2 dB
9	1	0	0	1	+3 dB
10	1	0	1	0	+4 dB
11	1	0	1	1	+5 dB
12	1	1	0	0	+6 dB

Figure 105: Downlink PGA gain per pga_index.

This function can be called by the MMI at anytime.

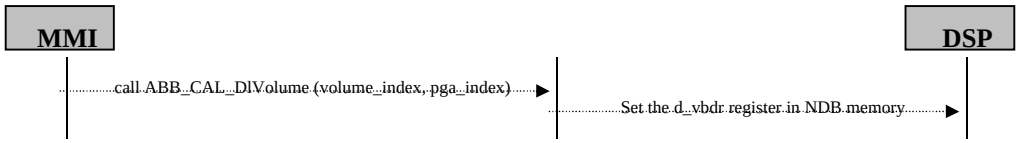


Figure 106: ABB_CAL_DlVolume process.

15.6 ABB_UlMute

By executing this function the microphone can be set to mute. The function prototype is:

void ABB_UlMute (BOOL mute)

When the input parameter **mute** is set to TRUE, the downlink path is set to mute. The volume applied to the speaker prior to switching to mute state is memorized, so that the same volume is output to the speaker after calling the function ABB_UlMute() with **mute**=FALSE.
This function can be called by the MMI at anytime.

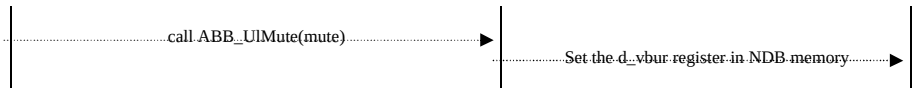


Figure 107: ABB_UlMute process.

15.7 ABB_Audio_Config

By executing this function the Analogue Base Band voice path can be set to a configuration. The function prototype is:

void ABB_Audio_Config (UWORD16 data)

The input parameter data corresponds to the new setting of the VBCR register of the Analog Base Band. The configuration determined during calibration phase is used for the initialization of the VBCR registers. This is achieved by setting the constant **C_VBCR**, file l1_rf$(RF).h to the required value.

This function can be called by the MMI at anytime.

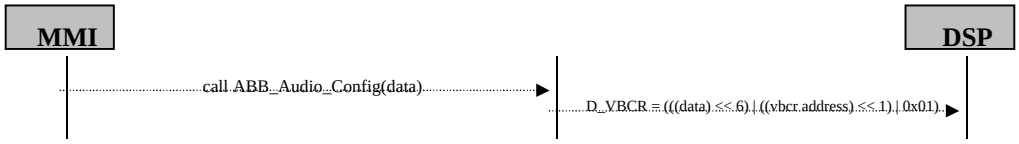


Figure 108: ABB_Audio_Config process.

15.8 ABB_Audio_Config_2

By executing this function the Analogue Base Band voice path 2 (VBCR2 register) can be set to a configuration. The function prototype is:

void ABB_Audio_Config_2 (UWORD16 data)

The input parameter data corresponds to the new setting of the VBCR2 register of the Analog Base Band. The configuration determined during calibration phase is used for the initialization of the VBCR2 registers. This is achieved by setting the constant **C_VBCR2**, file l1_rf$(RF).h to the required value.

This function can be called by the MMI at anytime.

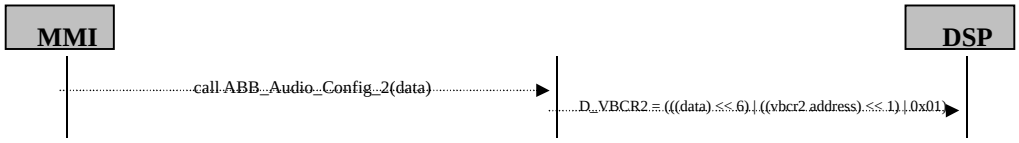


Figure 109: ABB_Audio_Config_2 process.

15.9 ABB_Audio_Control

By executing the function, the Analog Base Band audio stereo path is set to a configuration. The function prototype is:

void ABB_Audio_Control(UWORD16 data)

The input parameter data corresponds to the new setting of the VAUDCTRL register of the Analog Base Band.

The function can be called at anytime.

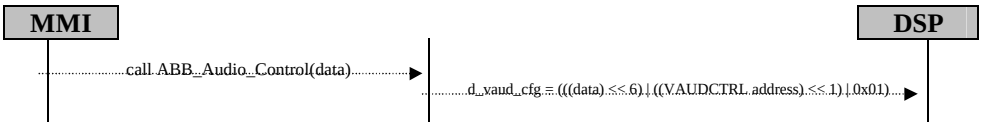


Figure 110: ABB_Audio_Control process.

15.10 ABB_Audio_On_Off

By executing the function, the Analog Base Band audio stereo output is set to a configuration. The function prototype is:

void ABB_Audio_On_Off(UWORD16 data)

The input parameter data corresponds to the new setting of the VAUOCTRL register of the Analog Base Band.

The function can be called at anytime.

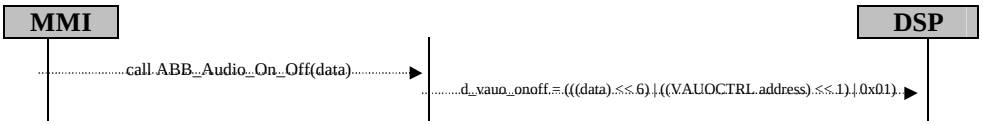


Figure 111: ABB_Audio_On_Off process.

15.11 ABB_Audio_Volume

By executing the function, the Analog Base Band audio stereo volume is set to a configuration. The function prototype is:

void ABB_Audio_Volume(UWORD16 data)

The input parameter data corresponds to the new setting of the VAUSCTRL register of the Analog Base Band.

The function can be called at anytime.

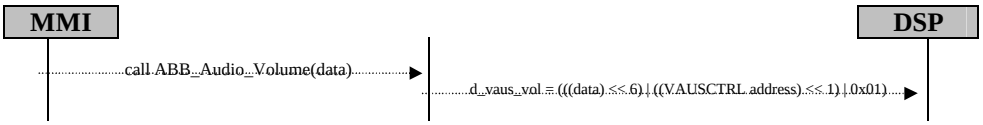


Figure 112: ABB_Audio_Volume process.

15.12 ABB_Audio_PLL

By executing the function, the Analog Base Band audio PLL is set to a configuration. The function prototype is:

void ABB_Audio_PLL(UWORD16 data)

The input parameter data corresponds to the new setting of the VAUDPLL register of the Analog Base Band.

The function can be called at anytime.

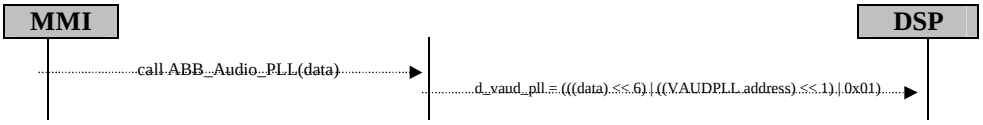


Figure 113: ABB_Audio_PLL process.

15.13 ABB_Audio_VBPop

By executing the function, the Analog Base Band Pop noise cancellation is set to a configuration. The function prototype is:

void ABB_Audio_VBPop(UWORD16 data)

The input parameter data corresponds to the new setting of the VBPOP register of the Analog Base Band.

The function can be called at anytime.

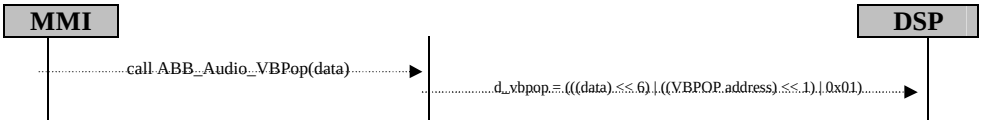


Figure 114: ABB_Audio_VBPop process.

15.14 ABB_Audio_Delay_Init

By executing the function, the Analog Base Band delay init is set to a configuration. The function prototype is:

void ABB_Audio_Delay_Init(UWORD8 delay)

The function can be called at anytime.

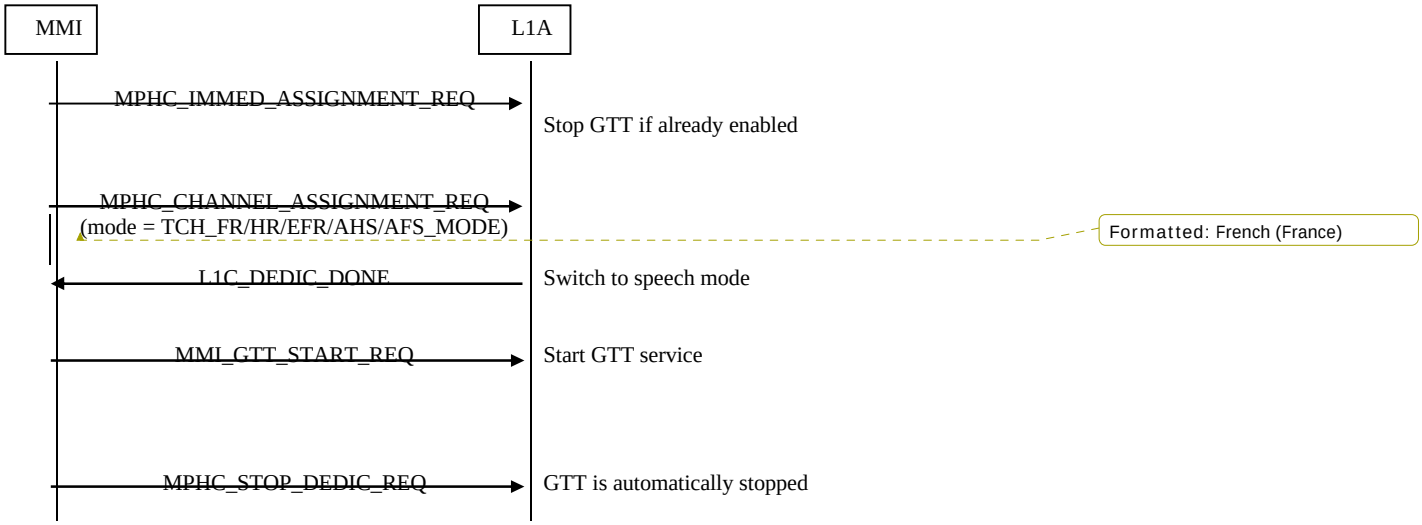
Figure 115: ABB_Audio_Delay_Init process.

16 Global Text Telephony

16.1 MMI rules

The MMI must respect following rules:

- When the MMI starts GTT (MMI_GTT_START_REQ message), it must wait the start confirmation (MMI_GTT_START_CON) before stopping the GTT
- The MMI isn't allowed to restart GTT before the reception of a stop confirmation (MMI_GTT_STOP_CON).
- The MMI can receive a stop confirmation (MMI_GTT_STOP_CON) in two cases:
  - The MMI requested to stop GTT (MMI_GTT_STOP_REQ) and the L1 confirms the stop confirmation message.
  - L1 automatically stops GTT because incompatibility with the MODEM mode of operation
- The MMI is only allowed to run GTT during GSM **dedicated speech** mode
- The MMI is allowed to run GTT task with all another audio tasks except **melody format E2** and **speech recognition**
- Because GTT cannot work at the same time as melody, GTT service must be disabled during immediate assignment part and idle parts. For security, the GTT service is automatically stopped by L1 at the end of dedicated mode or when an immediate assignment is performed. It is up to the upper layer to send the start message in order to enable the service after the channel assignment
- By default GTT is disabled at L1 initialization



16.2 MMI-L1 interface

Direction	Message name	Message Type
MMI->L1	MMI_GTT_START_REQ	Trigger
MMI<-L1	MMI_GTT_START_CON	Trigger
MMI->L1	MMI_GTT_STOP_REQ	Trigger
MMI<-L1	MMI_GTT_STOP_CON	Trigger

16.3 L1A-L1S message interface

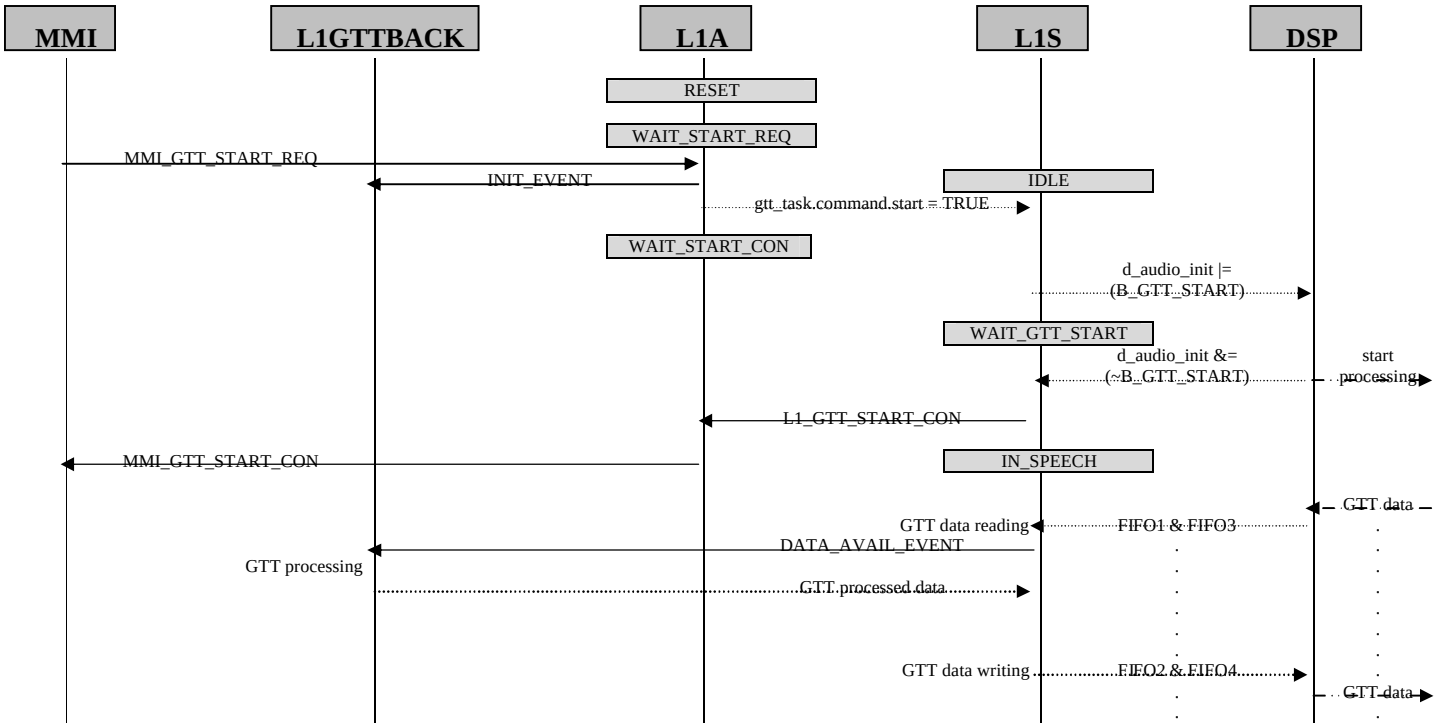
Direction	Message name	Message Type
L1S->L1A	L1_GTT_START_CON	Trigger
L1S->L1A	L1_GTT_STOP_CON	Trigger

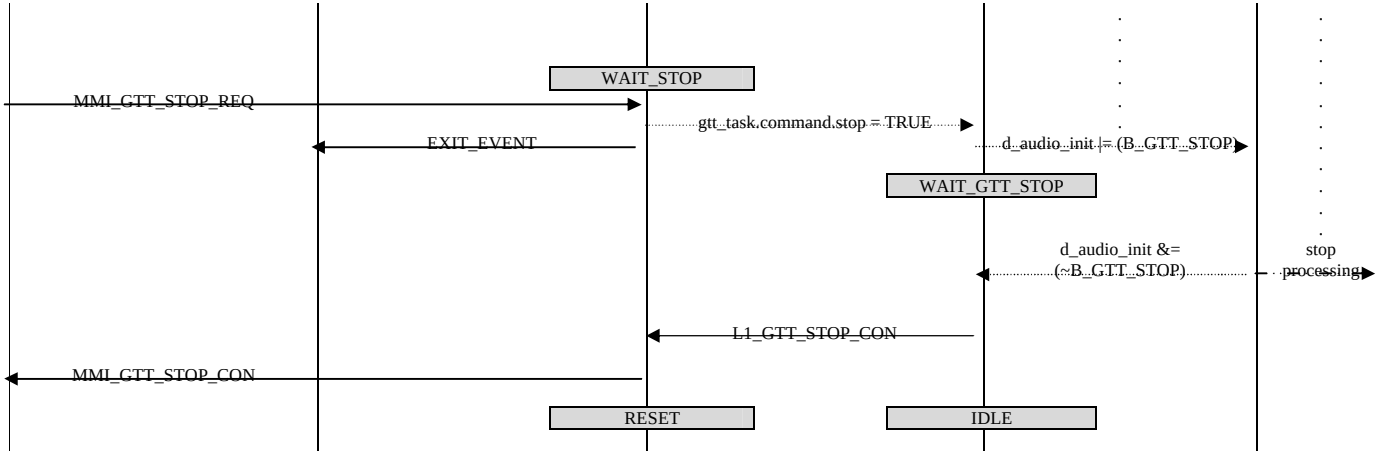
16.4 GTT background – L1 interface

This interface is performed using events instead of queues.

Direction	Event name
L1->GTT background	DATA_AVAIL_EVENT
L1->GTT background	INIT_EVENT
L1->GTT background	EXIT_EVENT

16.5 Process flow





During the IN_SPEECH state, L1S only performs GTT operation if speech is enabled in dedicated mode (half rate and full rate):

- L1S reads FIFO1 and FIFO3 every 4 frames (20 ms) if some data is available in FIFO1/3 from the DSP
- L1S sends an event to the GTT background task every 20 ms
- L1S writes to FIFO2 and FIFO4 every 26 frames (done on the idle frame fn%26=25 ó 125 ms) if some data is available for writing from the GTT background task

Note: Explanation of the data contained in MCU-DSP FIFOs

	DSP	MCU	MCU	DSP
Uplink	FIFO1 (TTY demodulator	à Baudot decoder)	FIFO2 (CTM transmitter	à CTM modulator)
Downlink	FIFO3 (CTM demodulator	à CTM receiver)	FIFO4 (CTM receiver	à TTY modulator)

17 Compilation flags

This chapter introduces the compilation flags dedicated to the audio features. All these flags are defined in the file *l1_confg.h* (folder *include*).

Flag name	Possible value	Role	Restriction
AUDIO_TASK	1	Enable all audio features	
	0	Disable all audio features	
KEYBEEP	1	Enable the keybeep features	The AUDIO_TASK must be equal to 1.
	0	Disable the keybeep features	None.
TONE	1	Enable the tone features	The AUDIO_TASK must be equal to 1.
	0	Disable the tone features	None.
MELODY_E1	1	Enable the melody format E1 features	The AUDIO_TASK must be equal to 1.
	0	Disable the melody format E1 features	None.
VOICE_MEMO	1	Enable the voice memorization features	The AUDIO_TASK must be equal to 1.
	0	Disable the voice memorization features	None.
L1_VOICE_MEMO_AMR	1	Enable the voice memorization AMR features	The AUDIO_TASK must be equal to 1.
	0	Disable the voice memorization AMR features	None
SPEECH_RECO	1	Enable the speech recognition features	The AUDIO_TASK must be equal to 1.
	0	Disable the speech recognition features	None.
FIR	1	Enable the FIR features	The AUDIO_TASK must be equal to 1.
	0	Disable the FIR features	None.
AEC	1	Enable the AEC features	The AUDIO_TASK must be equal to 1.
	0	Disable the AEC features	None.
AUDIO_MODE	1	Enable the Audio mode feature.	The AUDIO_TASK must be equal to 1 and the DSP code must be the v3300.
	0	Disable the Audio mode feature.	None.
MELODY_E2	1	Enable the Melody format E2 feature.	The AUDIO_TASK must be equal to 1 and the DSP code must be the v3300.
	0	Disable the Melody format E2 feature.	None.
AUDIO_L1_STANDALONE	1	Enable the MMI emulator for the L1 stand-alone.	The AUDIO_TASK must be equal to 1
	0	Disable the MMI emulator for the L1 stand-alone.	None.
AUDIO_SIMULATION	1	Enable the PC audio simulator.	The AUDIO_TASK must be equal to 1.
	0	Disable the PC audio simulator.	None.

Figure 116: Compilation flags tables.

18 Annexes

18.1 Audio task with a DSP code

The 2 drawings below and documents [3] show the compatibility of all audio tasks depending on the DSP code:

- with GSM or GPRS modes
- with another audio task.

It reflects L1 MCU and DSP limitations.

DSP Code 32

	GSM						GPRS		GSM->GPRS	GPRS->GSM
	Idle	SMS	Dedicated				Idle	Transfert		
			Speech	FACCH	TCH/Data	IDS				
Keybeep										
Tones										
Melody										
VM play										
VM rec										
SR enroll										
SR update										
SR reco										
FIR										
AEC										

CAPTION:

can run in this mode.
can't run in this mode.

With DSP code 17/18/32, the Voice Memo Play feature does not work with IDS module at 9600bps in non-transparent mode. GSM-DSP BUG00670 describes this problem.

DSP Code 32

	Keybeep	Tones	Melody E1	VM play	VM rec	SR enroll	SR update	SR reco	FIR	AEC
Keybeep										
Tones										
Melody E1										
VM play										
VM rec										
SR enroll										
SR update										
SR reco										
FIR										
AEC										

CAPTION:

can run with this task
can't run with this task.

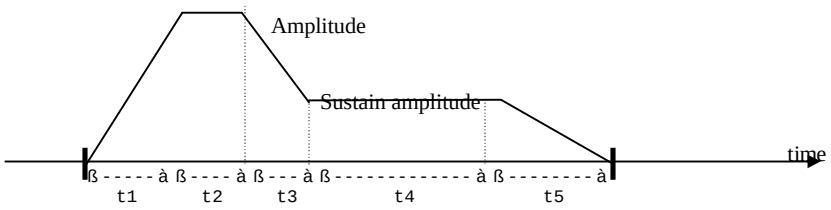
18.2 Melody format E1

The software module named “MELODY” performs the melody generation used in a mobile phone during the ringing of an incoming call. The melody can be generated by the mean of adding eight sine waves.

The sine waves synthesis consists in loading the DSP/MCU interface with data at a maximum of 20 ms rate. Each sine wave is described by a frequency and an energy envelope described with several segments. These descriptions are associated in the melody file with a time stamp indicating the starting time of the sine wave. The first segment is the attack, the second is decay, the third is the sustain, the last is the release.

The melody format E1 gives the same features and solves some problems found in the previous format E0. Because of the new DSP algorithms used the CPU load is now much smaller and is compatible with the simultaneous run of a vocoder (which is not the case with E0). The melody generator roadmap will be based on the E1 interface, the E0 format will not be supported later and a converter E0->E1 will be provided. The E1 file format includes some unused bit fields which must be loaded with null values, this spare area is reserved for extensions using programmable wave source instead of sine wave.

Melodies sine waves generation



The melody format features a maximum of 8 simultaneous tones. The tones have a five segments energy pattern.
The format consists in:

```
typedef struct {
   unsigned int ml_time_offset: 8;    // 20 ms duration unit until next data load

   unsigned int ml_ocscil_01  : 1;    // conditional activation of
   unsigned int ml_ocscil_02  : 1;    //  the oscillators
   unsigned int ml_ocscil_03  : 1;    //
   unsigned int ml_ocscil_04  : 1;    //
   unsigned int ml_ocscil_05  : 1;    //
   unsigned int ml_ocscil_06  : 1;    //
   unsigned int ml_ocscil_07  : 1;    //
   unsigned int ml_ocscil_08  : 1;    //

) T_ML_time_descriptor;
```

Followed by all the necessary number of `T_ML_osc_descriptor` descriptors as defined in the bit fields "`ml_ocscil_0(i)`".

```
typedef struct {
   unsigned int ml_synchro : 1;       // LSB is a synchronization flag
   unsigned int ml_square1 : 1;       // soft squaring of the waveforms
   unsigned int ml_square2 : 1;       // sharp squaring of the waveforms
   unsigned int ml_directF : 1;       // flag for precision frequency generation
   unsigned int ml_amp     : 6;       // amplitude of the sine wave
   unsigned int ml_freq    : 6;       // frequency of the sine wave

   unsigned int ml_unused1 : 4;       // reserved, must be 0000
   unsigned int ml_load1   : 1;       // load first field
   unsigned int ml_load2   : 1;       // load second field
   unsigned int ml_length  : 10;      // total time in 20ms unit (0 .. 20s)

   unsigned int ml_unused2 : 8;       // reserved, must be 00000000
   unsigned int ml_tremT0  : 3;       // tremolo period
   unsigned int ml_tremFreq: 5;       // tremolo frequency sweep amplitude

   unsigned int ml_sustain : 4;       // sustain gain relative to ml_amp
   unsigned int ml_t5      : 3;       // release time (20ms .. 2.56s)
   unsigned int ml_t3      : 3;       // decay time (20ms .. 2.56s)
   unsigned int ml_t2      : 3;       // attack time (0 .. 2.54s)
   unsigned int ml_t1      : 3;       // attack time (20ms .. 2.56s)
) T_ML_osc_descriptor;
```

Formatted: English (U.S.)

ml_synchro is loaded with value “1” when a new data field is to be taken into account by the DSP, this bit is reseted by the DSP.

ml_directF is used to generate pure sine waves without energy pattern computations. When ml_directF is set the 14 bits of [ml_square1/2, ml_amp, ml_freq] are used for the direct coding of a sine wave frequency and the 10bits of ml_t4 are used to define the amplitude:

freq = 8192 * cos (2 * pi * (sine wave frequency in Hertz) / 8000)
amp = A/32 * sin ((2 * pi * (sine wave frequency in Hertz) / 8000) where A is the desired for amplitude (ml_amp range is [0 .. 1023]), A range is [0..32767].

For example a 150 Hz full amplitude sine wave gives freq=0x1FC7, amp=0x078, the coding of the tone descriptor is: `7F1F 1E00`.

ml_square1 is a flag used to generate soft square waveforms: the original sine wave is clipped after a 2 bits left shift on each samples.
ml_square2 is a flag used to generate hard square waveforms: the original sine wave is clipped after a 5 bits left shift on each samples.

ml_amp is the amplitude in linear format: the ml_amp range is [0 .. 63] the corresponding 16 bits absolute amplitude range is [511..32767].

ml_freq is the pitch in semitones unit: the ml_freq range is [15 .. 55] corresponding to the notes [E4 (330Hz) .. G#7(3322Hz)].

ml_load1 is used determine if the 16bits fields [ml_tremT0..] is present in the file descriptor.
ml_load2 is used determine if the 16bits fields [ml_sustain..] is present in the file descriptor.

ml_length is the total length of the tone in linear 20ms unit. The ml_length range is [2..1023] corresponding to [40ms .. 20460ms].

ml_tremT0 is the tremolo period in 20ms logarithmic unit, ml_tremT0 range is [0..7]. Value 0 corresponds to 20ms, 1=> 40ms, 2 => 80ms, ..., 7 => 2560ms.

ml_tremFreq is the tremolo pitch increment and decrement to the field ml_freq in semitones unit. The addition ml_freq+ml_tremFreq is clipped in order to fit within the ml_freq range.

ml_sustain : amplitude during sustain period (ml_sustain range is [0..15])
ml_sustain = ((D / A) * 16 –1) where D is the desired output value and A the amplitude described with ml_amp.
for example with ml_amp = 10000 and a desired output of 5000 the
value is ml_sustain= 7

ml_t1 is the attack time periods in logarithmic 20ms period. The ml_t1 range is [0..7] corresponding to 20ms, 40ms, 80ms,..., 2560ms.

ml_t2 is the extension of attack time periods in logarithmic 20ms period. The ml_t2 range is [0..7] corresponding to 0ms, 20ms, 60ms, 140ms ..., 2540ms.

ml_t3 is the decay time in logarithmic 20ms period. The range is [0..7] corresponding to 20ms, 40ms, 80ms, ..., 2560ms.

ml_t5 is the release time logarithmic 20ms period. The range is [0..7] corresponding to 20ms, 40ms, 80ms, ..., 2560ms.
The file is ended with a time descriptor nulled.

Development tools

See [2].

18.3 Key beep/tones generation

Value	Frequency (Hz)
40	1800,24
41	1795,20
42	1790,16
43	1785,12
44	1780,07
45	1775,02
46	1769,97
47	1764,91
48	1759,85
49	1754,78
50	1749,71
51	1744,64
52	1739,56
53	1734,48
54	1729,39
55	1724,30
56	1719,21
57	1714,11
58	1709,00
59	1703,90
60	1698,78
61	1693,66
62	1688,54
63	1683,41
64	1678,28
65	1673,14
66	1667,99
67	1662,84
68	1657,69
69	1652,52
70	1647,36
71	1642,18
72	1637,00
73	1631,82
74	1626,63
75	1621,43
76	1616,22
77	1611,01
78	1605,79
79	1600,57
80	1595,33
81	1590,09
82	1584,85
83	1579,59
84	1574,33
85	1569,06
86	1563,79
87	1558,50
88	1553,21
89	1547,91
90	1542,60
91	1537,28
92	1531,96
93	1526,63
94	1521,28
95	1515,93
96	1510,57
97	1505,20
98	1499,82
99	1494,43
100	1489,04
101	1483,63
102	1478,21
103	1472,78
104	1467,34
105	1461,90
106	1456,44
107	1450,97
108	1445,49
109	1440,00
110	1434,49
111	1428,98
112	1423,46
113	1417,92
114	1412,37
115	1406,81
116	1401,24
117	1395,65
118	1390,05
119	1384,44
120	1378,82
121	1373,18
122	1367,53
123	1361,87
124	1356,19
125	1350,50
126	1344,79
127	1339,07
128	1333,33
129	1327,58
130	1321,82
131	1316,04
132	1310,24
133	1304,43
134	1298,60
135	1292,75
136	1286,89
137	1281,01
138	1275,11
139	1269,20
140	1263,27
141	1257,32
142	1251,35
143	1245,36
144	1239,36
145	1233,33
146	1227,29
147	1221,22
148	1215,14
149	1209,03
150	1202,91
151	1196,76
152	1190,59
153	1184,40
154	1178,18
155	1171,94
156	1165,68
157	1159,40

158	1153,09
159	1146,75
160	1140,40
161	1134,01
162	1127,60
163	1121,16
164	1114,70
165	1108,21
166	1101,69
167	1095,14
168	1088,57
169	1081,96
170	1075,32
171	1068,66
172	1061,96
173	1055,23
174	1048,46
175	1041,66
176	1034,83
177	1027,97
178	1021,06
179	1014,13
180	1007,15
181	1000,14
182	993,08
183	985,99
184	978,86
185	971,68
186	964,47
187	957,21
188	949,90
189	942,55
190	935,15
191	927,71
192	920,21
193	912,67
194	905,07
195	897,42
196	889,72
197	881,96
198	874,14
199	866,27
200	858,33

201	850,33
202	842,27
203	834,14
204	825,94
205	817,67
206	809,33
207	800,92
208	792,42
209	783,85
210	775,20
211	766,46
212	757,63
213	748,71
214	739,70
215	730,59
216	721,37
217	712,05
218	702,62
219	693,08
220	683,41
221	673,62
222	663,70
223	653,64
224	643,44
225	633,10
226	622,59
227	611,92
228	601,07
229	590,04
230	578,81
231	567,38
232	555,73
233	543,85
234	531,71
235	519,31
236	506,63
237	493,63
238	480,31
239	466,62
240	452,54
241	438,02
242	423,03
243	407,50

244	391,39
245	374,60
246	357,05
247	338,62
248	319,14
249	298,43
250	276,21
251	252,06
252	225,37
253	195,12
254	159,26
255	112,58

Value	Amplitude (dB)
1	-48,16
2	-42,14
3	-38,62
4	-36,12
5	-34,18
6	-32,60
7	-31,26
8	-30,10
9	-29,08
10	-28,16
11	-27,33
12	-26,58
13	-25,88
14	-25,24
15	-24,64
16	-24,08
17	-23,55
18	-23,05
19	-22,58
20	-22,14
21	-21,72
22	-21,31
23	-20,93
24	-20,56
25	-20,20
26	-19,86
27	-19,53
28	-19,22
29	-18,91
30	-18,62
31	-18,33
32	-18,06
33	-17,79
34	-17,53
35	-17,28
36	-17,03
37	-16,80
38	-16,56
39	-16,34
40	-16,12
41	-15,90
42	-15,70
43	-15,49
44	-15,29
45	-15,10
46	-14,90
47	-14,72
48	-14,54
49	-14,36
50	-14,18
51	-14,01
52	-13,84
53	-13,67
54	-13,51
55	-13,35
56	-13,20
57	-13,04
58	-12,89
59	-12,74
60	-12,60
61	-12,45
62	-12,31
63	-12,17
64	-12,04
65	-11,90
66	-11,77
67	-11,64
68	-11,51
69	-11,38
70	-11,26
71	-11,13
72	-11,01
73	-10,89
74	-10,78
75	-10,66
76	-10,54
77	-10,43
78	-10,32
79	-10,21
80	-10,10
81	-9,99
82	-9,88
83	-9,78
84	-9,67
85	-9,57
86	-9,47
87	-9,37
88	-9,27
89	-9,17
90	-9,08
91	-8,98
92	-8,88
93	-8,79
94	-8,70
95	-8,61
96	-8,51
97	-8,42
98	-8,34
99	-8,25
100	-8,16
101	-8,07
102	-7,99
103	-7,90
104	-7,82
105	-7,74
106	-7,65
107	-7,57
108	-7,49
109	-7,41
110	-7,33
111	-7,25
112	-7,18
113	-7,10
114	-7,02
115	-6,95
116	-6,87
117	-6,80
118	-6,72
119	-6,65
120	-6,58
121	-6,50
122	-6,43
123	-6,36
124	-6,29
125	-6,22
126	-6,15
127	-6,08

128	-6,02
129	-5,95
130	-5,88
131	-5,81
132	-5,75
133	-5,68
134	-5,62
135	-5,55
136	-5,49
137	-5,43
138	-5,36
139	-5,30
140	-5,24
141	-5,18
142	-5,11
143	-5,05
144	-4,99
145	-4,93
146	-4,87
147	-4,81
148	-4,75
149	-4,70
150	-4,64
151	-4,58
152	-4,52
153	-4,47
154	-4,41
155	-4,35
156	-4,30
157	-4,24
158	-4,19
159	-4,13
160	-4,08
161	-4,02
162	-3,97
163	-3,92
164	-3,86
165	-3,81
166	-3,76
167	-3,71
168	-3,65
169	-3,60
170	-3,55
171	-3,50
172	-3,45
173	-3,40
174	-3,35
175	-3,30
176	-3,25
177	-3,20
178	-3,15
179	-3,10
180	-3,05
181	-3,01
182	-2,96
183	-2,91
184	-2,86
185	-2,82
186	-2,77
187	-2,72
188	-2,68
189	-2,63
190	-2,58
191	-2,54
192	-2,49
193	-2,45
194	-2,40
195	-2,36
196	-2,31
197	-2,27
198	-2,23
199	-2,18
200	-2,14
201	-2,10
202	-2,05
203	-2,01
204	-1,97
205	-1,92
206	-1,88
207	-1,84
208	-1,80
209	-1,76
210	-1,72
211	-1,67
212	-1,63
213	-1,59
214	-1,55
215	-1,51
216	-1,47
217	-1,43
218	-1,39
219	-1,35
220	-1,31
221	-1,27
222	-1,23
223	-1,19
224	-1,16
225	-1,12
226	-1,08
227	-1,04
228	-1,00
229	-0,96
230	-0,93
231	-0,89
232	-0,85
233	-0,81
234	-0,78
235	-0,74
236	-0,70
237	-0,67
238	-0,63
239	-0,59
240	-0,56
241	-0,52
242	-0,48
243	-0,45
244	-0,41
245	-0,38
246	-0,34
247	-0,31
248	-0,27
249	-0,24
250	-0,20
251	-0,17
252	-0,13
253	-0,10
254	-0,06
255	-0,03

18.4 Audio message identifier

The following list sums up the audio message identifier useful for the upper layers what want to use the L1 audio features.

```
#if (AUDIO_TASK == 1)
  #define P_AUDIO ( 0x18 )

  // Messages MMI <-> L1A
  #if (KEYBEEP)
    #define MMI_KEYBEEP_START_REQ            ( ( P_AUDIO << 8 ) |   0 )
    #define MMI_KEYBEEP_STOP_REQ             ( ( P_AUDIO << 8 ) |   1 )
    #define MMI_KEYBEEP_START_CON            ( ( P_AUDIO << 8 ) |   2 )
    #define MMI_KEYBEEP_STOP_CON             ( ( P_AUDIO << 8 ) |   3 )
  #endif
  #if (TONE)
    #define MMI_TONE_START_REQ               ( ( P_AUDIO << 8 ) |   4 )
    #define MMI_TONE_STOP_REQ                ( ( P_AUDIO << 8 ) |   5 )
    #define MMI_TONE_START_CON               ( ( P_AUDIO << 8 ) |   6 )
    #define MMI_TONE_STOP_CON                ( ( P_AUDIO << 8 ) |   7 )
  #endif
  #if (MELODY_E1)
    #define MMI_MELODY0_START_REQ            ( ( P_AUDIO << 8 ) |   8 )
    #define MMI_MELODY0_STOP_REQ             ( ( P_AUDIO << 8 ) |   9 )
    #define MMI_MELODY0_START_CON            ( ( P_AUDIO << 8 ) |  10 )
    #define MMI_MELODY0_STOP_CON             ( ( P_AUDIO << 8 ) |  11 )

    #define MMI_MELODY1_START_REQ            ( ( P_AUDIO << 8 ) |  12 )
    #define MMI_MELODY1_STOP_REQ             ( ( P_AUDIO << 8 ) |  13 )
    #define MMI_MELODY1_START_CON            ( ( P_AUDIO << 8 ) |  14 )
    #define MMI_MELODY1_STOP_CON             ( ( P_AUDIO << 8 ) |  15 )
  #endif
  #if (VOICE_MEMO)
    #define MMI_VM_PLAY_START_REQ            ( ( P_AUDIO << 8 ) |  16 )
    #define MMI_VM_PLAY_STOP_REQ             ( ( P_AUDIO << 8 ) |  17 )
    #define MMI_VM_PLAY_START_CON            ( ( P_AUDIO << 8 ) |  18 )
    #define MMI_VM_PLAY_STOP_CON             ( ( P_AUDIO << 8 ) |  19 )

    #define MMI_VM_RECORD_START_REQ          ( ( P_AUDIO << 8 ) |  20 )
    #define MMI_VM_RECORD_STOP_REQ           ( ( P_AUDIO << 8 ) |  21 )
    #define MMI_VM_RECORD_START_CON          ( ( P_AUDIO << 8 ) |  22 )
    #define MMI_VM_RECORD_STOP_CON           ( ( P_AUDIO << 8 ) |  23 )
  #endif
  #if (SPEECH_RECO)
    #define MMI_SR_ENROLL_START_REQ          ( ( P_AUDIO << 8 ) |  24 )
    #define MMI_SR_ENROLL_STOP_REQ           ( ( P_AUDIO << 8 ) |  25 )
    #define MMI_SR_ENROLL_START_CON          ( ( P_AUDIO << 8 ) |  26 )
    #define MMI_SR_ENROLL_STOP_CON           ( ( P_AUDIO << 8 ) |  27 )

    #define MMI_SR_UPDATE_START_REQ          ( ( P_AUDIO << 8 ) |  28 )
    #define MMI_SR_UPDATE_STOP_REQ           ( ( P_AUDIO << 8 ) |  29 )
    #define MMI_SR_UPDATE_START_CON          ( ( P_AUDIO << 8 ) |  30 )
    #define MMI_SR_UPDATE_STOP_CON           ( ( P_AUDIO << 8 ) |  31 )

    #define MMI_SR_RECO_START_REQ            ( ( P_AUDIO << 8 ) |  32 )
    #define MMI_SR_RECO_STOP_REQ             ( ( P_AUDIO << 8 ) |  33 )
    #define MMI_SR_RECO_START_CON            ( ( P_AUDIO << 8 ) |  34 )
    #define MMI_SR_RECO_STOP_CON             ( ( P_AUDIO << 8 ) |  35 )

    #define MMI_SR_UPDATE_CHECK_START_REQ    ( ( P_AUDIO << 8 ) |  36 )
    #define MMI_SR_UPDATE_CHECK_STOP_REQ     ( ( P_AUDIO << 8 ) |  37 )
    #define MMI_SR_UPDATE_CHECK_START_CON    ( ( P_AUDIO << 8 ) |  38 )
```

```
  #define MMI_SR_UPDATE_CHECK_STOP_CON      ( ( P_AUDIO << 8 ) |  39 )
#endif
#if (AEC)
  #define MMI_AEC_REQ                       ( ( P_AUDIO << 8 ) |  40 )
  #define MMI_AEC_CON                       ( ( P_AUDIO << 8 ) |  41 )
#endif
#if (FIR)
  #define MMI_AUDIO_FIR_REQ                 ( ( P_AUDIO << 8 ) |  42 )
  #define MMI_AUDIO_FIR_CON                 ( ( P_AUDIO << 8 ) |  43 )
#endif
#if (AUDIO_MODE)
  #define MMI_AUDIO_MODE_REQ                ( ( P_AUDIO << 8 ) |  44 )
  #define MMI_AUDIO_MODE_CON                ( ( P_AUDIO << 8 ) |  45 )
#endif
#if (MELODY_E2)
  #define MMI_MELODY0_E2_START_REQ          ( ( P_AUDIO << 8 ) |  46 )
  #define MMI_MELODY0_E2_STOP_REQ           ( ( P_AUDIO << 8 ) |  47 )
  #define MMI_MELODY0_E2_START_CON          ( ( P_AUDIO << 8 ) |  48 )
  #define MMI_MELODY0_E2_STOP_CON           ( ( P_AUDIO << 8 ) |  49 )

  #define MMI_MELODY1_E2_START_REQ          ( ( P_AUDIO << 8 ) |  50 )
  #define MMI_MELODY1_E2_STOP_REQ           ( ( P_AUDIO << 8 ) |  51 )
  #define MMI_MELODY1_E2_START_CON          ( ( P_AUDIO << 8 ) |  52 )
  #define MMI_MELODY1_E2_STOP_CON           ( ( P_AUDIO << 8 ) |  53 )
#endif
#if (L1_VOICE_MEMO_AMR)
  #define MMI_VM_AMR_PLAY_START_REQ         ( ( P_AUDIO << 8 ) |  54 )
  #define MMI_VM_AMR_PLAY_STOP_REQ          ( ( P_AUDIO << 8 ) |  55 )
  #define MMI_VM_AMR_PLAY_START_CON         ( ( P_AUDIO << 8 ) |  56 )
  #define MMI_VM_AMR_PLAY_STOP_CON          ( ( P_AUDIO << 8 ) |  57 )

  #define MMI_VM_AMR_RECORD_START_REQ       ( ( P_AUDIO << 8 ) |  58 )
  #define MMI_VM_AMR_RECORD_STOP_REQ        ( ( P_AUDIO << 8 ) |  59 )
  #define MMI_VM_AMR_RECORD_START_CON       ( ( P_AUDIO << 8 ) |  60 )
  #define MMI_VM_AMR_RECORD_STOP_CON        ( ( P_AUDIO << 8 ) |  61 )
#endif
#if (L1_CPORT == 1)
  #define MMI_CPORT_CONFIGURE_REQ           ( ( P_AUDIO << 8 ) |   62 )
  #define MMI_CPORT_CONFIGURE_CON           ( ( P_AUDIO << 8 ) |   63 )
#endif


// Messages L1S -> L1A
#if (KEYBEEP)
  #define L1_KEYBEEP_START_CON              ( ( P_AUDIO << 8 ) |  64 )
  #define L1_KEYBEEP_STOP_CON               ( ( P_AUDIO << 8 ) |  65 )
#endif
#if (TONE)
  #define L1_TONE_START_CON                 ( ( P_AUDIO << 8 ) |  66 )
  #define L1_TONE_STOP_CON                  ( ( P_AUDIO << 8 ) |  67 )
#endif
#if (MELODY_E1)
  #define L1_MELODY0_START_CON              ( ( P_AUDIO << 8 ) |  68 )
  #define L1_MELODY0_STOP_CON               ( ( P_AUDIO << 8 ) |  69 )

  #define L1_MELODY1_START_CON              ( ( P_AUDIO << 8 ) |  70 )
  #define L1_MELODY1_STOP_CON               ( ( P_AUDIO << 8 ) |  71 )
#endif
#if (VOICE_MEMO)
  #define L1_VM_PLAY_START_CON              ( ( P_AUDIO << 8 ) |  72 )
  #define L1_VM_PLAY_STOP_CON               ( ( P_AUDIO << 8 ) |  73 )

  #define L1_VM_RECORD_START_CON            ( ( P_AUDIO << 8 ) |  74 )
```

```
    #define L1_VM_RECORD_STOP_CON            ( ( P_AUDIO << 8 ) |  75 )
  #endif
  #if (SPEECH_RECO)
    #define L1_SR_ENROLL_START_CON           ( ( P_AUDIO << 8 ) |  76 )
    #define L1_SR_ENROLL_STOP_CON            ( ( P_AUDIO << 8 ) |  77 )

    #define L1_SR_UPDATE_START_CON           ( ( P_AUDIO << 8 ) |  78 )
    #define L1_SR_UPDATE_STOP_CON            ( ( P_AUDIO << 8 ) |  79 )

    #define L1_SR_RECO_START_CON             ( ( P_AUDIO << 8 ) |  80 )
    #define L1_SR_RECO_STOP_CON              ( ( P_AUDIO << 8 ) |  81 )
    #define L1_SR_RECO_STOP_IND              ( ( P_AUDIO << 8 ) |  82 )
    #define L1_SR_PROCESSING_STOP_CON        ( ( P_AUDIO << 8 ) |  83 )
  #endif
  #if (AEC)
    #define L1_AEC_CON                       ( ( P_AUDIO << 8 ) |  84 )
    #define L1_AEC_IND                       ( ( P_AUDIO << 8 ) |  85 )
  #endif
  #if (FIR)
    #define L1_AUDIO_FIR_CON                 ( ( P_AUDIO << 8 ) |  86 )
  #endif
  #if (AUDIO_MODE)
    #define L1_AUDIO_MODE_CON                ( ( P_AUDIO << 8 ) |  87 )
  #endif
  #if (L1_VOICE_MEMO_AMR)
    #define L1_VM_AMR_PLAY_START_CON         ( ( P_AUDIO << 8 ) |  88 )
    #define L1_VM_AMR_PLAY_STOP_CON          ( ( P_AUDIO << 8 ) |  89 )

    #define L1_VM_AMR_RECORD_START_CON       ( ( P_AUDIO << 8 ) |  90 )
    #define L1_VM_AMR_RECORD_STOP_CON        ( ( P_AUDIO << 8 ) |  91 )
  #endif
  #if (L1_CPORT == 1)
    #define L1_CPORT_CONFIGURE_CON           ( ( P_AUDIO << 8 ) |  92 )
  #endif

  // Messages L1A <-> Audio background
  #if (SPEECH_RECO)
    #define L1_SRBACK_SAVE_DATA_REQ          ( ( P_AUDIO << 8 ) |  93 )
    #define L1_SRBACK_SAVE_DATA_CON          ( ( P_AUDIO << 8 ) |  94 )
    #define L1_SRBACK_LOAD_MODEL_REQ         ( ( P_AUDIO << 8 ) |  95 )
    #define L1_SRBACK_LOAD_MODEL_CON         ( ( P_AUDIO << 8 ) |  96 )
    #define L1_SRBACK_TEMP_SAVE_DATA_REQ     ( ( P_AUDIO << 8 ) |  97 )
    #define L1_SRBACK_TEMP_SAVE_DATA_CON     ( ( P_AUDIO << 8 ) |  98 )
  #endif
  #if (MELODY_E2)
    #define L1_BACK_MELODY_E2_LOAD_INSTRUMENT_REQ   ( ( P_AUDIO << 8 ) |  99 )
    #define L1_BACK_MELODY_E2_LOAD_INSTRUMENT_CON   ( ( P_AUDIO << 8 ) |  100 )
    #define L1_BACK_MELODY_E2_UNLOAD_INSTRUMENT_REQ ( ( P_AUDIO << 8 ) |  101 )
    #define L1_BACK_MELODY_E2_UNLOAD_INSTRUMENT_CON ( ( P_AUDIO << 8 ) |  102 )

    #define L1_MELODY0_E2_STOP_CON                  ( ( P_AUDIO << 8 ) |  103 )
    #define L1_MELODY1_E2_STOP_CON                  ( ( P_AUDIO << 8 ) |  104 )
  #endif

  #if (OP_RIV_AUDIO == 1)
    #if (L1_AUDIO_DRIVER == 1)
      #define L1_AUDIO_DRIVER_IND                ( ( P_AUDIO << 8 ) |  105 )
    #endif
  #endif
#endif // AUDIO_TASK == 1
```