UM1633
User manual

Omni2 stereo widening library software expansion for STM32Cube

Introduction

The Omni2 stereo widening library user manual describes the software interface and requirements for the integration of the module into a main program like the Audio STM32Cube expansion software and provides a rough understanding of the underlying algorithm.

The Omni2 stereo widening library implements the stereo widening effect. Multichannel audio virtualization (up to 7.1 input channel) is available in another package called Omni2 multichannel library.

The Omni2 stereo widening library is part of the X-CUBE-AUDIO firmware package.

Contents

1 Module overview 5

1.1 Algorithm function 5

1.2 Module configuration 6

1.3 Resource summary 7

2 Module interfaces 8

2.1 APIs 8

2.1.1 omni2_reset function 8

2.1.2 omni2_setParam function 8

2.1.3 omni2_getParam function 9

2.1.4 omni2_setConfig function 9

2.1.5 omni2_getConfig function 10

2.1.6 omni2_process function 10

2.2 External definitions and types 10

2.2.1 Input and output buffers 10

2.2.2 Returned error values 11

2.3 Static parameters structure 11

2.4 Dynamic parameters structure 12

3 Algorithm description 14

3.1 Processing steps 14

3.2 Data formats 14

3.3 Performance assessment 14

4 System requirements and hardware setup 16

4.1 Recommended setup for stereo widening effect 16

4.2 Recommendations for an optimal setup 17

4.2.1 Module integration example 18

4.2.2 Module integration summary 18

5 How to run and tune the application 20

6 Revision history 21

List of tables

Table 1. Sampling rates . . . 5
Table 2. Resources summary . . . 7
Table 3. omni2_reset . . . 8
Table 4. omni2_setParam . . . 9
Table 5. omni2_getParam . . . 9
Table 6. omni2_setConfig . . . 9
Table 7. omni2_getConfig . . . 10
Table 8. omni2_process . . . 10
Table 9. Input and output buffers . . . 11
Table 10. Returned error values . . . 11
Table 11. Static parameters structure . . . 12
Table 12. Dynamic parameters structure . . . 13
Table 13. Setup examples . . . 16
Table 14. Document revision history . . . 21

List of figures

Figure 1. Mono to stereo perception for mono inputs 5
Figure 2. Stereo widening perception for stereo inputs 6
Figure 3. Block diagram of the Omni2 module 14
Figure 4. Setup for stereo widening effect 16
Figure 5. Omnisurround placement in the audio chain 17
Figure 6. API call procedure 18

1 Module overview

1.1 Algorithm function

This module provides functions to handle mono to stereo expansion and stereo widening effect.

Table 1 describes the sampling rates and the input/output formats supported:

Table 1. Sampling rates

Library	Audio effect	Channel conversions	Supported sampling frequencies
OMNI2_SW_CMx_IAR.a OMNI2_SW_32b_CMx_IAR.a OMNI2_SW_CMx_GCC.a OMNI2_SW_32b_CMx_GCC.a OMNI2_SW_CMx_Keil.lib OMNI2_SW_32b_CMx_Keil.lib	Mono2Stereo	1.0 to 2.0	48 kHz
		1.0 to 3.0 (2.0 + center)	
	Stereo Widening	2.0 to 2.0	
		2.0 to 3.0 (2.0 + center)	

The *Figure 1* present the effect perception with only two physical loudspeakers:

Figure 1. Mono to stereo perception for mono inputs

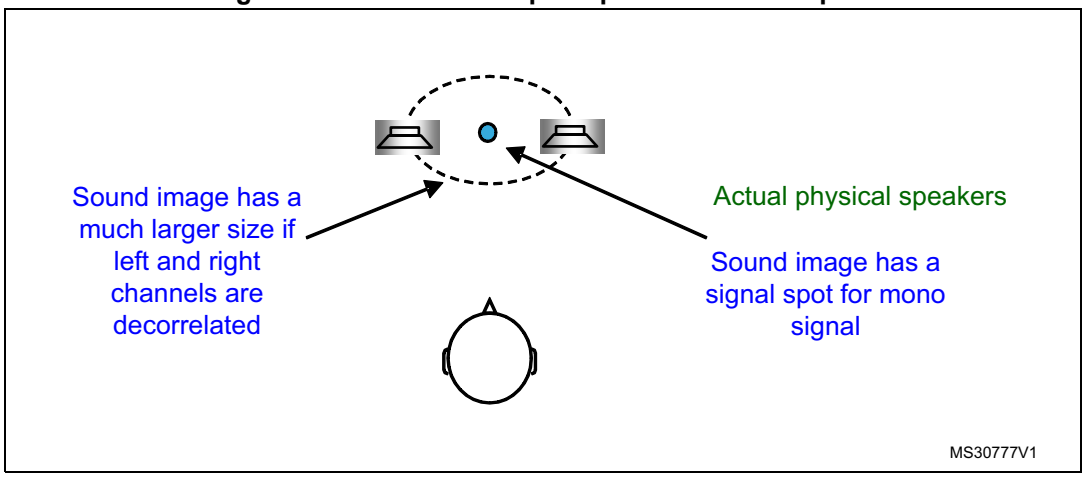
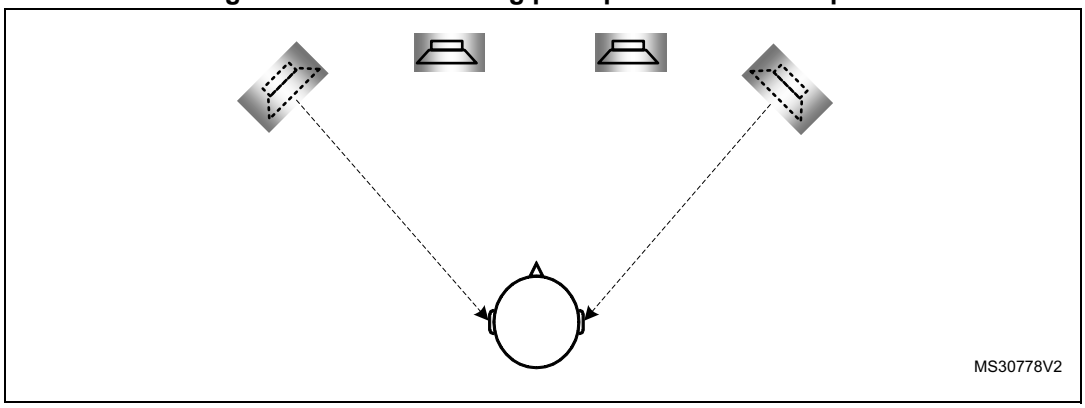


Figure 2. Stereo widening perception for stereo inputs



The listening angle corresponds to the angle between the listener and the physical speakers – please refer to *Section 4: System requirements and hardware setup* for more information. "_LS" refers to Largely Spaced speakers, that is about 30 degrees listening angle, "_CS" refers to Closely Spaced speakers, that is about 20 degrees listening angle and "_VCS" refers to Very Closely Spaced speakers, that is about 10 degrees listening angle.

1.2 Module configuration

The module supports mono and stereo interleaved 16-bit and 32-bit I/O data at a 48 kHz sampling frequency.

Several versions of the module are available depending on the I/O format, the Cortex Core and the used tool chain:

- OMNI2_SW_CM4_IAR.a / OMNI2_SW_CM4_GCC.a / OMNI2_SW_CM4_Keil.lib: for 16 bits input/output buffers and it runs on any STM32 microcontroller featuring a core with Cortex-M4 instruction set.
- OMNI2_SW_32b_CM4_IAR.a / OMNI2_SW_32b_CM4_GCC.a / OMNI2_SW_32b_CM4_Keil.lib: for 32 bits input/output buffers and it runs on any STM32 microcontroller featuring a core with Cortex-M4 instruction set.
- OMNI2_SW_CM7_IAR.a / OMNI2_SW_CM7_GCC.a / OMNI2_SW_CM7_Keil.lib: for 16 bits input/output buffers and it runs on any STM32 microcontroller featuring a core with Cortex-M7 instruction set.
- OMNI2_SW_32b_CM7_IAR.a / OMNI2_SW_32b_CM7_GCC.a / OMNI2_SW_32b_CM7_Keil.lib: for 32 bits input/output buffers and it runs on any STM32 microcontroller featuring a core with Cortex-M7 instruction set.

1.3 Resource summary

Table 2 contains Flash, stack, RAM and frequency requirements.

Those footprints are measured on board, using IAR Embedded Workbench for ARM v7.40 (IAR Embedded Workbench common components v7.2).

Table 2. Resources summary

-	Use case @ 48 kHz	Core	Flash code (.text)	Flash data (.rodata)	Stack	Persistent RAM	Scratch RAM	Frequency (MHz)
Stereo widening	1.0=>2.0	M4	4480 Bytes	1080 Bytes	310 Bytes	1996 Bytes	576 Bytes	4.8
		M7	**4518 Bytes**					**3.7**
	2.0=>2.0 largely spaced speakers	M4	4480 Bytes					13.6
		M7	**4518 Bytes**					**8.4**
	2.0=>2.0 closely spaced speakers	M4	4480 Bytes					17
		M7	**4518 Bytes**					**9.9**
	2.0=>2.0 very closely spaced speakers	M4	4480 Bytes					17
		M7	**4518 Bytes**					**10**

Note: Footprints on STM32F7 are measured on boards with stack and heap sections located in DTCM memory and with a 10ms framing.
Scratch RAM is the memory that can be shared with other process running on the same priority level. This memory is not used from one frame to another by Omni2 routines.

2 Module interfaces

Two files are needed to integrate the Omni2 module. OMNI2_SW_xxx_CMy_zzz.a /.lib library and the *omni2_glo.h* header file which contain all definitions and structures to be exported to the software integration framework.

Note: The audio_fw_glo.h file is a generic header file common to all audio modules; it must be included in the audio framework.

2.1 APIs

Six generic functions have a software interface to the main program:

- omni2_reset
- omni2_setParam
- omni2_getParam
- omni2_setConfig
- omni2_getConfig
- omni2_process

2.1.1 omni2_reset function

This procedure initializes the persistent memory of the Omni2 module and initializes static parameters with default values.

```
int32_t omni2_reset(void *persistent_mem_ptr, void *scratch_mem_ptr);
```

Table 3. omni2_reset

I/O	Name	Type	Description
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Input	scratch_mem_ptr	void *	Pointer to internal scratch memory
Returned value	–	int32_t	Error value

This routine must be called at least once at initialization time, when the real time processing has not started.

2.1.2 omni2_setParam function

This procedure writes module static parameters from the main framework to the module's internal memory. It can be called after the reset routine and before the start of the real time processing. It handles the static parameters, i.e. the parameters with values which cannot be changed during the module processing (frame by frame).

Note: Static parameters cannot be changed dynamically after module processing started, while dynamic parameters can be modified during processing (through omni2_setConfig() API described below).

```
int32_t omni2_setParam(omni2_static_param_t *input_static_param_ptr, void
*persistent_mem_ptr);
```

Table 4. omni2_setParam

I/O	Name	Type	Description
Input	input_static_param_ptr	omni2_static_param_t*	Pointer to static parameters structure
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Returned value	–	int32_t	Error value

2.1.3 omni2_getParam function

This procedure gets the module's static parameters from the module internal memory to the main framework. It can be called after the reset routine and before the start of the real time processing. It handles the static parameters, i.e. the parameter with values which cannot be changed during the module processing (frame by frame).

```
int32_t omni2_getParam(omni2_static_param_t *input_static_param_ptr, void
*persistent_mem_ptr;
```

Table 5. omni2_getParam

I/O	Name	Type	Description
Input	input_static_param_ptr	omni2_static_param_t *	Pointer to static parameters structure
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Returned value	–	int32_t	Error value

2.1.4 omni2_setConfig function

This procedure sets the module's dynamic parameters from the main framework to the module's internal memory. It can be called at any time during processing (after the reset and setParam routines).

```
int32_t omni2_setConfig(omni2_dynamic_param_t *input_dynamic_param_ptr,
void *persistent_mem_ptr);
```

Table 6. omni2_setConfig

I/O	Name	Type	Description
Input	input_dynamic_param_ptr	omni2_dynamic_param_t *	Pointer to dynamic parameters structure
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Returned value	–	int32_t	Error value

2.1.5 omni2_getConfig function

This procedure gets module's dynamic parameters from the internal persistent memory to the main framework. It can be called at any time during processing (after the reset and setParam routines).

```
int32_t omni2_getConfig(omni2_dynamic_param_t *input_dynamic_param_ptr,
void *persistent_mem_ptr);
```

Table 7. omni2_getConfig

I/O	Name	Type	Description
Input	input_dynamic_param_ptr	omni2_dynamic_param_t *	Pointer to dynamic parameters structure
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Returned value	–	int32_t	Error value

2.1.6 omni2_process function

This procedure is the module's main processing routine. It should be called at any time, to process each frame.

```
int32_t omni2_process(buffer_t *input_buffer, buffer_t *output_buffer, void
*persistent_mem_ptr);
```

Table 8. omni2_process

I/O	Name	Type	Description
Input	input_buffer	buffer_t *	Pointer to input buffer structure
Output	output_buffer	buffer_t *	Pointer to output buffer structure
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Returned value	–	int32_t	Error value

This process routine can run in place only in case of 2.0 to 2.0 processing, such as for the stereo widening effect.

2.2 External definitions and types

In order to facilitate the integration in the main frameworks, some types and definitions have been defined.

2.2.1 Input and output buffers

The Omni2 library is using extended I/O buffers which contain, in addition to the samples, some useful information on the stream such as the number of channels, the number of bytes per sample and the interleaving mode.

An I/O buffer structure type, as described below, must be followed and filled in by the main framework before each call to the processing routine:

```
typedef struct {
    int32_t    nb_channels;
    int32_t    nb_bytes_per_Sample;
    void       *data_ptr;
    int32_t    buffer_size;
    int32_t    mode;
} buffer_t;
```

Table 9. Input and output buffers

Name	Type	Description
nb_channels	int32_t	Number of channels in data: 1 for mono, 2 for stereo
nb_bytes_per_Sample	int32_t	Dynamic data in number of bytes (2 for 16-bit data, …)
data_ptr	void *	Pointer to data buffer (must be allocated by the main framework)
buffer_size	int32_t	Number of samples per channel in the data buffer
mode	int32_t	Buffer mode: 0 = not interleaved, 1 = interleaved

2.2.2 Returned error values

Possible returned error values are described below:

Table 10. Returned error values

Definition	Value	Description
OMNI2_ERROR_NONE	0	OK - No error detected
OMNI2_ERROR	-1	Could be a bad sampling frequency, or a bad dynamic memory allocation
OMNI2_ERROR_PARSE_COMMAND	-2	Internal error - covers bad internal settings
OMNI2_BAD_HW	-3	May happen if the library is not used with the right hardware

2.3 Static parameters structure

Some static parameters must be set before calling the processing routine.

```
struct omni2_static_param {
    int32_t   Omni2CentreOutput;
    int32_t   AudioMode;
    int32_t   SamplingFreq;
};
typedef struct omni2_static_param omni2_static_param_t;
```

Table 11. Static parameters structure

Name	Type	Description
Omni2CentreOutput	int32_t	0 to disable the center output (for 3.0), 1 to enable the center output (3.0)
AudioMode	int32_t	Can be one field of the OMNI2_AcMode_Supported_e enumeration described below; it is used to describe the input data format
SamplingFreq	int32_t	I/O sampling frequency in Hz

Here are described the possible audio modes, but only the ones in green are supported in this module:

```
enum OMNI2_AcMode_Supported_e
{
    AMODE20t = 0x0, /*  Stereo channels for dolby pro logic */
    AMODE10 = 0x1, /*  Mono channel (1.0) */
    AMODE20 = 0x2, /*  Stereo channels (2.0) */
    AMODE30 = 0x3, /*  Stereo + Center channel (3.0) */
    AMODE32 = 0x7, /*  Stereo + Center channel + Surround Channels (5.0) */
    AMODE34 = 0xB, /*  Stereo + Center channel + Surround Channels + Center
Surround Channels (7.0) */
    AMODE20t_LFE = 0x80, /*  Stereo channels for dolby pro logic + LFE
channel */
    AMODE20_LFE = 0x82, /*  Stereo + LFE channel (2.1) */
    AMODE30_LFE = 0x83, /*  Stereo + Center channel + LFE channel (3.1) */
    AMODE32_LFE = 0x87, /*  Stereo + Center channel + LFE channel +
Surround Channels (5.1) */
    AMODE34_LFE = 0x8B, /*  Stereo + Center channel + LFE channel +
Surround Channels + Center Surround Channels  (7.1) */
    AMODE_ID  = 0xFF /*  End of configurations */
};
```

2.4 Dynamic parameters structure

Three dynamic parameters can be used.

```
struct omni2_dynamic_param {
    int32_t Omni2Enable;
    int32_t Omni2Strength;
    int32_t Omni2ListeningAngle;
};
typedef struct omni2_dynamic_param omni2_dynamic_param_t;
```

Table 12. Dynamic parameters structure

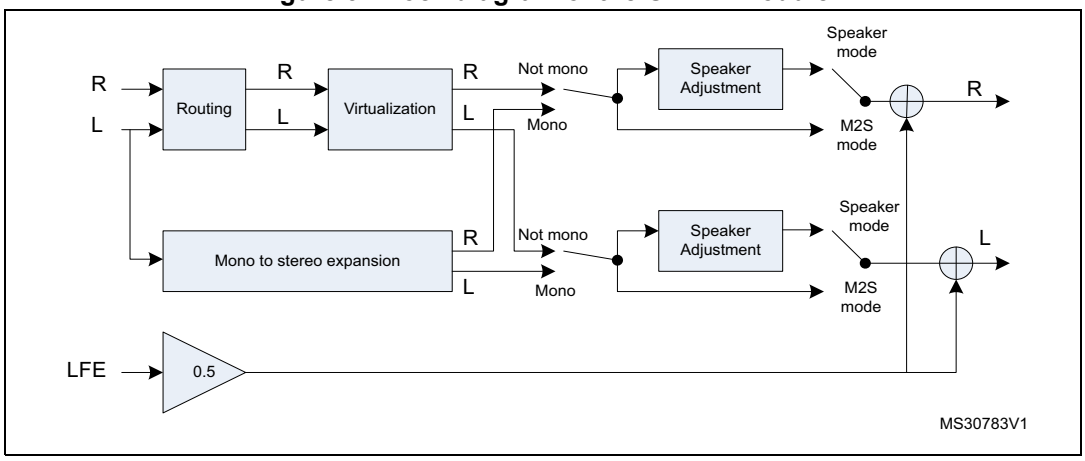
Name	Type	Description
Omni2Enable	int32_t	1 to enable the effect, 0 to disable the effect
Omni2Strength	int32_t	The value is from 0% (no widening perception) to 100% (maximum widening perception)
Omni2ListeningAngle	int32_t	Can be OMNI2_LISTENING_ANGLE_10 to have optimal effect with 10 degrees listening angle, OMNI2_LISTENING_ANGLE_20 to have optimal effect with 20 degrees listening angle and OMNI2_LISTENING_ANGLE_30 to have optimal effect with 30 degrees listening angle

3 Algorithm description

3.1 Processing steps

The block diagram of the Omni2 module is described in *Figure 3*.

Figure 3. Block diagram of the Omni2 module



Routing block: Carries out a premixing of the channels from mono-stereo interleaved channels so that it can be processed with a single virtualization structure.

Virtualization block: Applies the HRTF and Crosstalk cancellation function.

Speaker adjustment filter: Processes the audio signal after virtualization processing for speaker rendering and spectrum preservation.

Mono-to-stereo block: Carries out a mono-to-stereo expansion and bypasses the speaker rendering block.

Processing done in this mode increases the difference between left and right inputs (mainly in high frequencies) to increase the stereo sensation.

Omni2Strength parameter is used in this mode to modify direct and cross gains.

3.2 Data formats

The Omni2 module supports fixed point data in Q15 or Q31 format with a mono or a stereo interleaved pattern at 48 kHz sampling frequency.

3.3 Performance assessment

There is no objective measurement available for the Omni2 module; performances are based on a subjective assessment.

Below is a list of subjective indicators that could be used to evaluate the effect quality:

- **Balance between Left Front and Right Front:** capacity not to change energy on one front channel as compared to the other.
- **Stereo Widening:** ability to increase the audio perception angle to widen the stereo signal.
- **Center image stability:** ability to keep the center image at the center loudspeaker, or between the left and right front loudspeakers.
- **Sensitivity to sweet spot:** ability to feel a widening effect moving away from the sweet spot.
- **Spectrum preservation:** ability to keep the original spectrum perception, wherever the virtual sound comes from.

Note: For more information on the performance, refer to Section 4: System requirements and hardware setup and Section 5: How to run and tune the application.

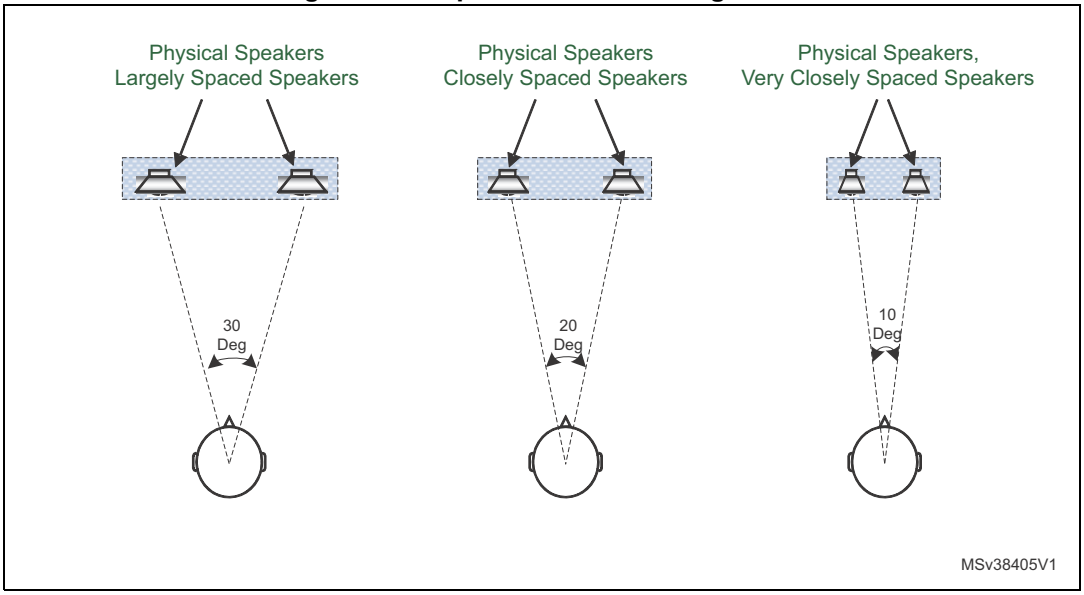
4 System requirements and hardware setup

Omni2 stereo widening libraries are built to run either on a Cortex M4 or on a Cortex M7 core, without FPU usage. They can be integrated and run on corresponding STM32F4/STM32L4 or STM32F7 family devices.

4.1 Recommended setup for stereo widening effect

The stereo widening effect is designed for a 10, 20 and 30 degrees typical listening angles.

Figure 4. Setup for stereo widening effect



Find in *Table 13* some setup examples and direct impacts on speaker distance to get a typical listening angle and an optimal stereo widening perception.

Table 13. Setup examples

Inter Speaker Distance	Speaker/Listener Distance	Corresponding Listening Angle	Recommended mode to use
0.3 m	2.5 m	7	OMNI2_LISTENING_ANGLE_10
0.3 m	1.8 m	10	OMNI2_LISTENING_ANGLE_10
0.3 m	1.2 m	14	OMNI2_LISTENING_ANGLE_10
0.3 m	0.6 m	28	OMNI2_LISTENING_ANGLE_30
0.4 m	2.5 m	9	OMNI2_LISTENING_ANGLE_10
0.4 m	1.8 m	13	OMNI2_LISTENING_ANGLE_10
0.4 m	1.2 m	19	OMNI2_LISTENING_ANGLE_20
0.4 m	0.6 m	37	OMNI2_LISTENING_ANGLE_30
0.6 m	2.5 m	14	OMNI2_LISTENING_ANGLE_10

Table 13. Setup examples (continued)

Inter Speaker Distance	Speaker/Listener Distance	Corresponding Listening Angle	Recommended mode to use
0.6 m	1.8 m	19	OMNI2_LISTENING_ANGLE_20
0.6 m	1.2 m	28	OMNI2_LISTENING_ANGLE_30
0.8 m	2.5 m	18	OMNI2_LISTENING_ANGLE_20
0.8 m	1.8 m	25	OMNI2_LISTENING_ANGLE_20 or OMNI2_LISTENING_ANGLE_30
0.8 m	1.2 m	37	OMNI2_LISTENING_ANGLE_30
1.0 m	2.5 m	23	OMNI2_LISTENING_ANGLE_20
1.0 m	1.8 m	31	OMNI2_LISTENING_ANGLE_30
1.2 m	2.5 m	27	OMNI2_LISTENING_ANGLE_30
1.2 m	1.8 m	37	OMNI2_LISTENING_ANGLE_30

It must be noted that the listener must be well centered between the two loudspeakers in order to benefit from the stereo widening effect because this effect is very sensitive to lateral sweet spot.

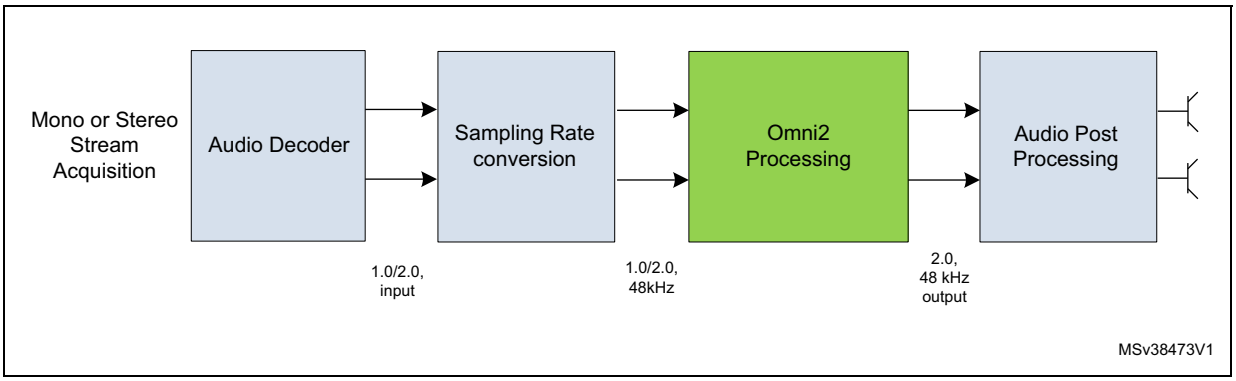
Listening angle below 20 degrees could correspond to a typical TV watching with down firing speakers, small sound bars or docking stations usage.

Positions of the virtual front channels should vary from ±15˚ to ±50˚.

4.2 Recommendations for an optimal setup

The library processing should be placed just after the sampling rate conversion in order to process the audio signal at 48 kHz. There is no need for this module to be close to the audio DAC, and some graphical equalizer and volume management modules can be placed after it without affecting the widening or virtualization perception.

Figure 5. Omnisurround placement in the audio chain

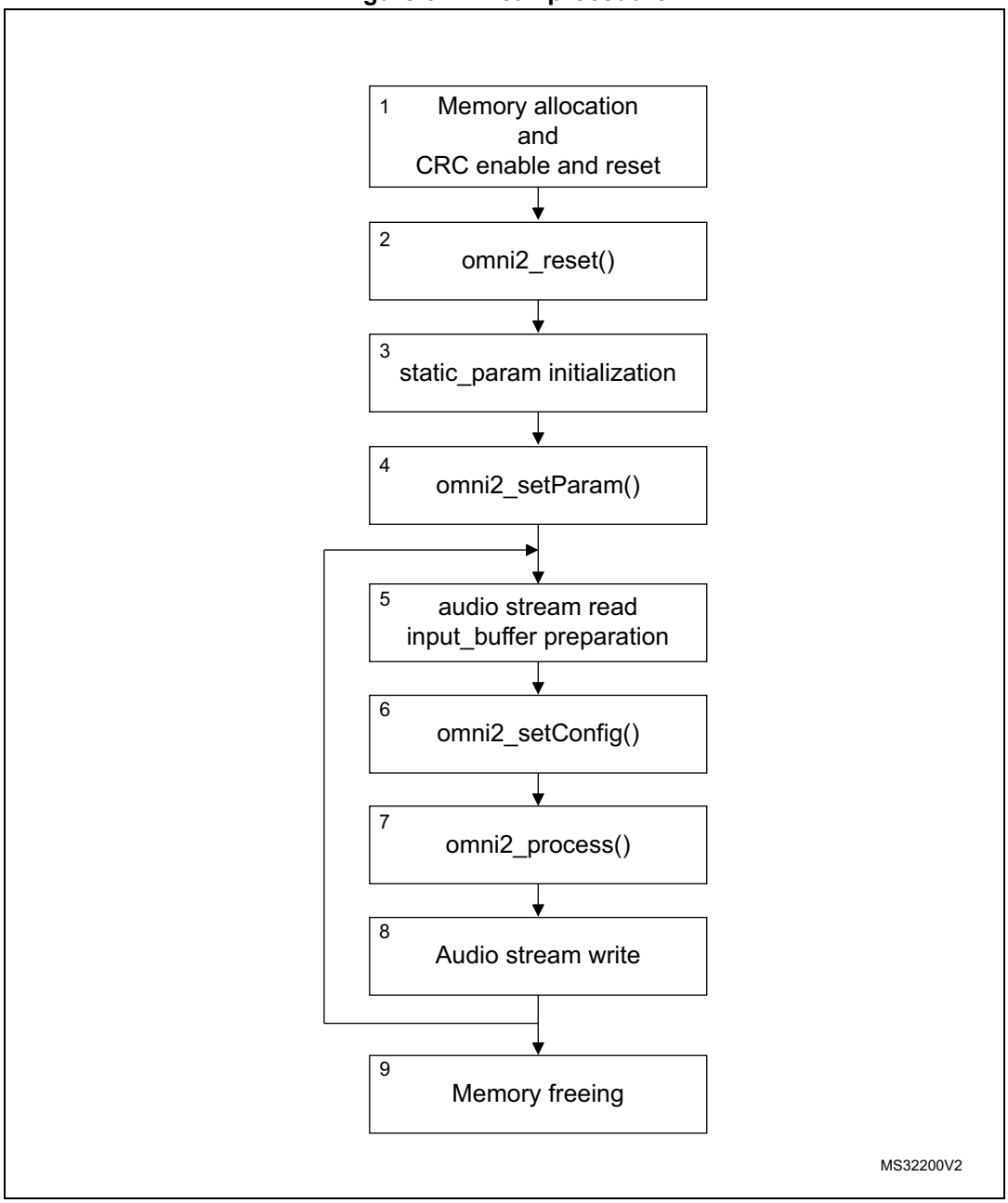


4.2.1 Module integration example

Cube expansion OMNI2_SW integration examples are provided on STM32746G-Discovery and STM32469I-Discovery boards. Please refer to provided integration code for more details.

4.2.2 Module integration summary

Figure 6. API call procedure



1. As explained above, the module's static and dynamic structures have to be allocated, as well as the input and output buffer, according to the structures defined in *Section 2.2.1: Input and output buffers*.

Furthermore, as Omni2 library runs on STM32 devices, CRC HW block must be enable and reset.

2. Once the memory has been allocated, the call to the omni2_reset() function initializes the internal variables.
3. The static configuration of module can now be set by initializing the static_param structure, once the input sampling frequency and the audio mode are known.
4. Call the omni2_setParam() routine to send the static parameters from the audio framework to the module.
5. The audio stream is read from the proper interface and the input_buffer structure has to be filled in according to the stream characteristics (number of channels, sample rate, interleaving and data pointer). The output buffer structure has to be set as well.
6. Get the dynamic parameters when they are updated and call the omni2_setConfig() routine to send the dynamic parameters from the audio framework to the module.
7. Call the main processing routine to apply the effect.
8. The output audio stream can now be written in the proper interface.
9. Once the processing loop is over, the allocated memory has to be freed.

5 How to run and tune the application

Once the module has been integrated into an audio framework to play stereo samples at 48kHz, user launches a player and the output file will be decoded and played with a stereo widening effect on loudspeakers without returning any error message.

The Omni2Enable, from the dynamic parameters of module, is used to enable and disable the effect.

The Omni2Strength field is used to change the virtual listening angle of front speakers. 0% means that front virtual speakers are close to physical speakers, while 100% leads to the widest virtual angle (typically 100 degrees).

Omni2ListeningAngle field, from the dynamic parameters of module, should be set depending on the End User configuration (largely, closely or very closely spaced speakers).

6 Revision history

Table 14. Document revision history

Date	Revision	Changes
11-Jun-2013	1	Initial release.
05-Jul-2013	2	Added 32-bit I/O data in *Section 1.2: Module configuration*. Updated *Table 2: Resources summary*. Added *Table 3: Frequency requirements (MHz)*. Added *lib_omni2_stereowidening_32b_m4.* library in *Section 2: Module interfaces*. Added *Q31* in *Section 3.2: Data formats*.
20-Aug-2013	3	Added “Omni2” to the document title. Updated the line introducing the “Multichannel audio virtualization” in the *Introduction*. Updated “Flash code stereo (.text)” and “Flash data (.rodata)“ values in *Table 2: Resources summary*. Updated the values on both Stereo lines of *Table 3: Frequency requirements (MHz)*. Replaced all “unset” speakers by “not set”.
28-Nov-2014	4	Classification changed from ST Restricted to public. Replaced the reference STSW-STM32APP by STM32-AUDIO100A.
10-Dec-2014	5	Updated *Section 5*.
22-Jul-2015	6	Updated: – *Section 1.1*, *Section 1.3*, *Section 4.1* – *Table 2*, *Table 12* – *Figure 4* – *Section 4.2.2* – *Section 5* Added *Table 13* removed: – section: “Omni2DownFiringSpekears set (closely spaced speakers mode)” – table: “Recommended angle (Omni2DownFiringSpekears not set)” – table: “Setup examples on the speaker distance (Omni2DownFiringSpekears not set)” – Section: “Recommended setup for stereo widening effect for closely spaced speakers (Omni2DownFiringSpekears set)” – table: “Recommended angle (Omni2DownFiringSpekears set)” – table: “Setup examples on the speaker distance (Omni2DownFiringSpekears set)”

Table 14. Document revision history (continued)

Date	Revision	Changes
25-Jan-2016	7	Updated: – *Table 1*, *Table 2*, *Table 12*, – *Section 1.1*, *Section 1.2*, *Section 2.4*, *Section 3.2*, *Section 4*, *Section 4.2*, *Section 4.2.1*, *Section 5* – *Figure 1*, *Figure 6* Added: – *Figure 5*
21-Mar-2017	8	Updated: – *Table 1: Sampling rates*, *Table 2: Resources summary*, *Table 3: omni2_reset*, *Table 4: omni2_setParam*, *Table 5: omni2_getParam*, *Table 6: omni2_setConfig*, *Table 7: omni2_getConfig*, *Table 8: omni2_process*, *Table 11: Static parameters structure*. – *Section 1.2: Module configuration*, *Section 1.3: Resource summary*, *Section 2: Module interfaces*, *Section 2.1.1: omni2_reset function*, *Section 2.1.2: omni2_setParam function*, , *Section 2.1.3: omni2_getParam function*, *Section 2.1.4: omni2_setConfig function*, *Section 2.1.5: omni2_getConfig function*, *Section 2.1.6: omni2_process function*, *Section 2.3: Static parameters structure*, *Section 4.2.1: Module integration example*, *Section 5: How to run and tune the application*.
08-Jan-2018	9	Replaced X-CUBE-AUDIO-F4, X-CUBE-AUDIO-F7 and X-CUBE-AUDIO-L4 with X-CUBE-AUDIO.