UM1641
User manual

Sampling rate conversion SRC236 library software expansion for STM32Cube

Introduction

The sampling rate conversion SRC236 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 SRC236 library is used to convert the sampling frequency from any rate with a ratio of 2, 3, 6, 3/2, 1/2, 1/3, 1/6 or 2/3.

The SRC236 library is part of the X-CUBE-AUDIO firmware package.

Contents

List of tables

List of figures

1 Module overview

1.1 Algorithm function

The SRC236 module provides functions to handle the rate conversions with a ratio of 2, 3, 6, 1/2, 1/3, 1/6, 3/2 and 2/3, according to the input sampling rate.

Table 1 shows the supported sampling rate conversions with the most commonly used sampling frequencies.

Note: Conversions from 44.1 kHz to 48 kHz are handled by another module named SRC441.

Table 1. Examples of supported rate conversions

Inputs in kHz	Output in kHz				
	8	**16**	**32**	**48**	**96**
8	–	Y	N	Y	N
16	Y	–	Y	Y	Y
32	N	Y	–	Y	Y
48	Y	Y	Y	–	Y
96	N	Y	Y	Y	–

1.2 Module configuration

The SRC236 module supports mono and stereo interleaved 16-bit or 32-bit I/O data, with a maximum input frame size of 240 stereo samples (corresponding to 5 ms scheduling at a 48 kHz sampling frequency).

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

- SRC236_CM4_IAR.a / SRC236_CM4_GCC.a / SRC236_CM4_Keil.lib: Standard configuration for low-MIPS and good quality requirements with 16 bits input/output buffers, it runs on any STM32 microcontroller featuring a core with Cortex-M4 instruction set.
- SRC236HQ_CM4_IAR.a / SRC236HQ_CM4_GCC.a / SRC236HQ_CM4_Keil.lib: Reserved for high quality needs (consumes more MIPS and memory as well) with 16 bits input/output buffers, it runs on any STM32 microcontroller featuring a core with Cortex-M4 instruction set.
- SRC236_32b_CM4_IAR.a / SRC236_32b_CM4_GCC.a / SRC236_32b_CM4_Keil.lib: Standard configuration for low-MIPS and good quality requirements, with 32 bits input/output buffers, it runs on any STM32 microcontroller featuring a core with Cortex-M4 instruction set.
- SRC236HQ_32b_CM4_IAR.a / SRC236HQ_32b_CM4_GCC.a / SRC236HQ_32b_CM4_Keil.lib: Reserved for high quality needs (consumes more MIPS and memory as well), with 32 bits input/output buffers, it runs on any STM32 microcontroller featuring a core with Cortex-M4 instruction set.
- SRC236_CM7_IAR.a / SRC236_CM7_GCC.a / SRC236_CM7_Keil.lib: Standard configuration for low-MIPS and good quality requirements with 16 bits input/output buffers, it runs on any STM32 microcontroller featuring a core with Cortex-M7 instruction set.
- SRC236HQ_CM7_IAR.a / SRC236HQ_CM7_GCC.a / SRC236HQ_CM7_Keil.lib: Reserved for high quality needs (consumes more MIPS and memory as well) with 16 bits input/output buffers, it runs on any STM32 microcontroller featuring a core with Cortex-M7 instruction set.
- SRC236_32b_CM7_IAR.a / SRC236_32b_CM7_GCC.a / SRC236_32b_CM7_Keil.lib: Standard configuration for low-MIPS and good quality requirements with 32 bits input/output buffers, it runs on any STM32 microcontroller featuring a core with Cortex-M7 instruction set.
- SRC236HQ_32b_CM7_IAR.a / SRC236HQ_32b_CM7_GCC.a / SRC236HQ_32b_CM7_Keil.lib: Reserved for high quality needs (consumes more MIPS and memory as well) with 32 bits input/output buffers, it runs on any STM32 microcontroller featuring a core with Cortex-M7 instruction set.

1.3 Resource summary

Table 2 contains the module requirements for the Flash, stack and RAM memories, all of them being independent of the SRC ratio. Only the required core frequency (MHz) is a variable and depends on the maximum sampling frequency between the input and the output.

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

Table 2. Resource summary

	Use Case with 5ms stereo interleaved buffers	Core	Flash code (.text)	Flash data (.rodata)	Stack	Persistent RAM	Scratch RAM	Frequency (MHz)
Standard	ratio 2: 8 -> 16 kHz	M4	1868 Bytes	40 Bytes	70 Bytes	492 Bytes	1924 Bytes	3.9
		M7	1816 Bytes					2
	ratio 2: 48 -> 96 kHz	M4	1868 Bytes					22.2
		M7	1816 Bytes					11.4
	ratio 3: 16 -> 48 kHz	M4	1868 Bytes					10.4
		M7	1816 Bytes					5.6
	ratio 6: 8 -> 48kHz	M4	1868 Bytes					10.2
		M7	1816 Bytes					5.5
	ratio 1/2: 16 -> 8 kHz	M4	1868 Bytes					3.6
		M7	1816 Bytes					1.7
	ratio 1/3: 48 -> 16kHz	M4	1868 Bytes					10
		M7	1816 Bytes					4.6
	ratio 1/6: 48 ->8 kHz	M4	1868 Bytes					9.6
		M7	1816 Bytes					4.4
	ratio 3/2: 32 -> 48 kHz	M4	1868 Bytes					10.4
		M7	1816 Bytes					5.6
	ratio 2/3: 24 -> 16 kHz	M4	1868 Bytes					5.4
		M7	1816 Bytes					2.7

Table 2. Resource summary (continued)

	Use Case with 5ms stereo interleaved buffers	Core	Flash code (.text)	Flash data (.rodata)	Stack	Persistent RAM	Scratch RAM	Frequency (MHz)
High quality	ratio 2: 8 -> 16 kHz	M4	2442 Bytes	40 Bytes	70 Bytes	972 Bytes	2844 Bytes	6.1
		M7	2294 Bytes					3.4
	ratio 2: 48 -> 96 kHz	M4	2442 Bytes					34.8
		M7	2294 Bytes					19.1
	ratio 3: 16 -> 48 kHz	M4	2442 Bytes					17.3
		M7	2294 Bytes					9.6
	ratio 6: 8 -> 48kHz	M4	2442 Bytes					17.1
		M7	2294 Bytes					9.4
	ratio 1/2: 16 -> 8 kHz	M4	2442 Bytes					5.8
		M7	2294 Bytes					3.1
	ratio 1/3: 48 -> 16kHz	M4	2442 Bytes					16.4
		M7	2294 Bytes					8.6
	ratio 1/6: 48 ->8 kHz	M4	2442 Bytes					16.1
		M7	2294 Bytes					8.3
	ratio 3/2: 32 -> 48 kHz	M4	2442 Bytes					17.3
		M7	2294 Bytes					9.5
	ratio 2/3: 24 -> 16 kHz	M4	2442 Bytes					8.6
		M7	2294 Bytes					4.6

Table 2. Resource summary (continued)

	Use Case with 5ms stereo interleaved buffers	Core	Flash code (.text)	Flash data (.rodata)	Stack	Persistent RAM	Scratch RAM	Frequency (MHz)
Standard 32-bit I/O	ratio 2: 8 -> 16 kHz	M4	1700 Bytes	40 Bytes	70 Bytes	972 Bytes	3364 Bytes	4.4
		M7	1774 Bytes					2.2
	ratio 2: 48 -> 96 kHz	M4	1700 Bytes					25.3
		M7	1774 Bytes					12.1
	ratio 3: 16 -> 48 kHz	M4	1700 Bytes					12.5
		M7	1774 Bytes					6
	ratio 6: 8 -> 48kHz	M4	1700 Bytes					12.3
		M7	1774 Bytes					5.9
	ratio 1/2: 16 -> 8 kHz	M4	1700 Bytes					4.2
		M7	1774 Bytes					1.9
	ratio 1/3: 48 -> 16kHz	M4	1700 Bytes					11.7
		M7	1774 Bytes					5.1
	ratio 1/6: 48 ->8 kHz	M4	1700 Bytes					11.3
		M7	1774 Bytes					5
	ratio 3/2: 32 -> 48 kHz	M4	1700 Bytes					12.7
		M7	1774 Bytes					6
	ratio 2/3: 24 -> 16 kHz	M4	1700 Bytes					6.2
		M7	1774 Bytes					2.9

Table 2. Resource summary (continued)

	Use Case with 5ms stereo interleaved buffers	Core	Flash code (.text)	Flash data (.rodata)	Stack	Persistent RAM	Scratch RAM	Frequency (MHz)
High quality 32-bit I/O	ratio 2: 8 -> 16 kHz	M4	2198 Bytes	40 Bytes	70 Bytes	1932 Bytes	4804 Bytes	7.1
		M7	2254 Bytes					3.7
	ratio 2: 48 -> 96 kHz	M4	2198 Bytes					40.7
		M7	2254 Bytes					20.7
	ratio 3: 16 -> 48 kHz	M4	2198 Bytes					20.4
		M7	2254 Bytes					10.4
	ratio 6: 8 -> 48kHz	M4	2198 Bytes					20.2
		M7	2254 Bytes					10.3
	ratio 1/2: 16 -> 8 kHz	M4	2198 Bytes					7
		M7	2254 Bytes					3.5
	ratio 1/3: 48 -> 16kHz	M4	2198 Bytes					19.6
		M7	2254 Bytes					9.6
	ratio 1/6: 48 ->8 kHz	M4	2198 Bytes					19.5
		M7	2254 Bytes					9.4
	ratio 3/2: 32 -> 48 kHz	M4	2198 Bytes					20.5
		M7	2254 Bytes					10.4
	ratio 2/3: 24 -> 16 kHz	M4	2198 Bytes					10.2
		M7	2254 Bytes					5.1

Note: Footprints on STM32F7 are measured on boards with stack and heap sections located in DTCM memory.

Note: 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 SRC236 routines.

2 Module interfaces

Two files are needed to integrate the SRC236 module: SRC236_xxx_CMy_zzz.a/.lib and *src236_glo.h* header file which contains all definitions and structures to be exported to the 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 API

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

- src236_reset
- src236_setParam
- src236_getParam
- src236_setConfig
- src236_getConfig
- src236_process

2.1.1 src236_reset function

This procedure initializes the persistent memory of the SRC236 module and initializes the static and dynamic parameters with default values.

API description:

```
int32_t src236_reset(void *persistent_mem_ptr, void *scratch_mem_ptr);
```

Table 3. src236_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

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

2.1.2 src236_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 parameter with values which cannot be changed during the module processing (frame after frame).

API description:

```
int32_t src236_setParam(src236_static_param_t *input_static_param_ptr, void
*persistent_mem_ptr);
```

Table 4. src236_setParam

I/O	Name	Type	Description
Input	input_static_param_ptr	src236_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 src236_getParam function

This procedure gets the module 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, that is the parameters with values which cannot be changed during the module processing (frame after frame).

API description:

```
int32_t src236_getParam(src236_static_param_t *input_static_param_ptr, void
*persistent_mem_ptr);
```

Table 5. src236_getParam

I/O	Name	Type	Description
Input	input_static_param_ptr	src236_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 src236_setConfig function

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

API description:

```
int32_t src236_setConfig(src236_dynamic_param_t *input_dynamic_param_ptr,
void *persistent_mem_ptr);
```

Table 6. src236_setConfig

I/O	Name	Type	Description
Input	input_dynamic_param_ptr	src236_dynamic_param_t *	Pointer to dynamic parameters structure
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Returned value	–	int32_t	Error value

Note: As there is currently no dynamic parameter, there is no need to call this routine in this module version.

2.1.5 src236_getConfig function

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

API description:

```
int32_t src236_getConfig(src236_dynamic_param_t *input_dynamic_param_ptr,
void *persistent_mem_ptr);
```

Table 7. src236_getConfig

I/O	Name	Type	Description
Input	input_dynamic_param_ptr	src236_dynamic_param_t *	Pointer to dynamic parameters structure
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Returned value	–	int32_t	Error value

Note: As there is currently no dynamic parameter, there is no need to call this routine in this module version.

2.1.6 src236_process function

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

API description:

```
int32_t src236_process(buffer_t *input_buffer, buffer_t *output_buffer,
void *persistent_mem_ptr);
```

Table 8. src236_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

Note: This process routine cannot run in place; the Input_buffer data is modified during processing, thus it cannot be used as is after any call to the src236_process() routine.

2.2 External definitions and types

For genericity reasons and to facilitate the integration in main frameworks, some types and definitions are adopted.

2.2.1 Input and output buffers

The SRC236 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 respected each time before calling processing routine; else, errors will be returned:

```
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, etc.
nb_bytes_per_Sample	int32_t	Dynamic of data in number of bytes (16-bit = 2, 24-bit = 3, 32-bit = 4)
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	In case of stereo stream, left and right channels can be interleaved. 0 = not interleaved, 1 = interleaved.

2.2.2 Returned error values

Table 10 describes possible returned error values.

Table 10. Returned error values

Definition	Value	Description
SRC236_ERROR_NONE	0	No error
SRC236_UNSUPPORTED_MODE	-1	Could be: unsupported SRC ratio used or unsupported input buffer format
SRC236_WRONG_NBBYTES_PER_SAMPLES	-2	Input data format is not correct
SRC236_UNSUPPORTED_NB_CHANNELS	-3	Input data is neither mono nor stereo
SRC236_UNSUPPORTED_INPLACE_PROCESSING	-4	Input and output buffer must be different
SRC236_INPUT_BUFFERS_TOO_BIG	-5	Input buffer size is bigger than allowed. The size is fixed to 240 stereo samples (5 ms stereo signal at 48 kHz sampling frequency).
SRC236_BAD_HW	-6	May happen if the library is not used with the right hardware.

2.3 Static parameters structure

There is one static parameter to be set before calling the process routine.

```
struct src236_static_param {
    int32_t   src_mode;
};
typedef struct src236_static_param src236_static_param_t;
```

Table 11. Static parameters structure

Name	Type	Description		
src_mode	int32_t	This corresponds to the SRC ratio to be applied, with the supported ratio as defined below		
		#define SRC_RATIO_2	0	// Means FS_Out = FS_In * 2
		#define SRC_RATIO_3	1	// Means FS_Out = FS_In * 3
		#define SRC_RATIO_6	2	// Means FS_Out = FS_In * 6
		#define SRC_RATIO_1_2	3	// Means FS_Out = FS_In / 2
		#define SRC_RATIO_1_3	4	// Means FS_Out = FS_In / 3
		#define SRC_RATIO_1_6	5	// Means FS_Out = FS_In / 6
		#define SRC_RATIO_3_2	6	// Means FS_Out = FS_In * 3/2
		#define SRC_RATIO_2_3	7	// Means FS_Out = FS_In * 2/3

2.4 Dynamic parameters structure

There is no dynamic parameter to be used. For compatibility with other structures, the dynamic parameter structure contains a dummy field.

```
struct src236_dynamic_param {
    int32_t   empty;
};
typedef struct src236_dynamic_param src236_dynamic_param_t;
```

Table 12. Dynamic parameters structure

Name	Type	Description
empty	int32_t	Dummy field, just required to have a non-empty structure

3 Algorithm description

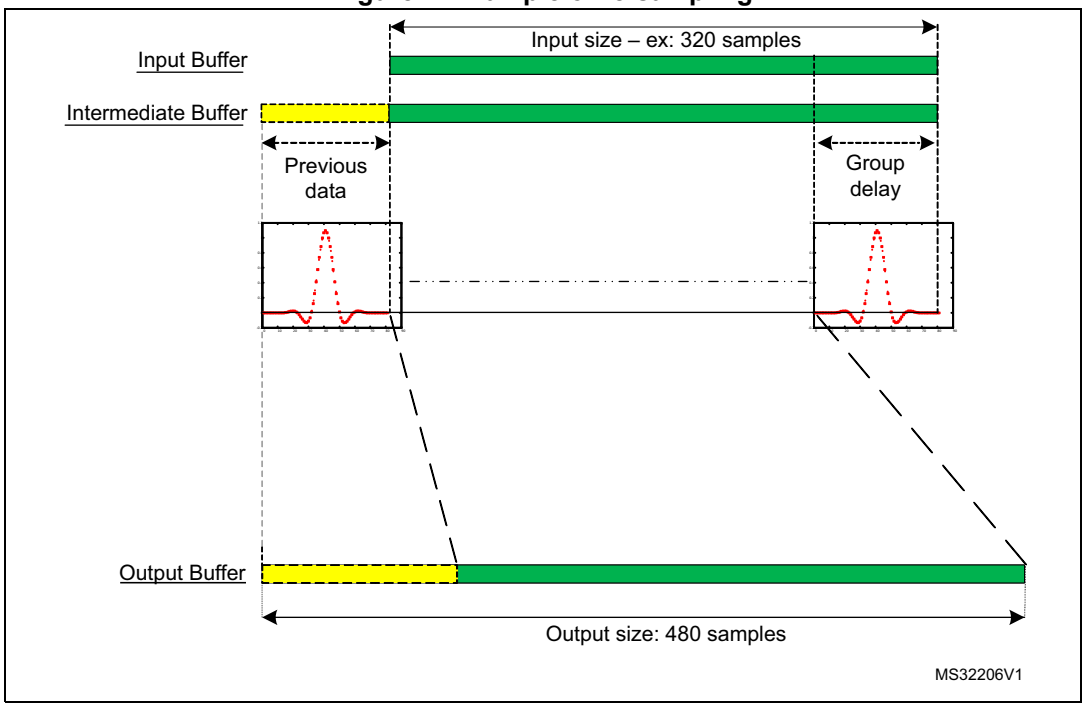
3.1 Processing steps

The SRC236 module is a re-sampler based on a one-stage polyphase filter. This implementation has been MIPS-optimized for Cortex M4 and M7 cores using a polyphase filter and SIMD instructions set.

The Flash size usage is optimized by a unique low pass filter that covers all supported rates and bit-fields that standardizes the processing, whatever the selected ratio.

Figure 1 is an example of re-sampling from 32 kHz to 48 kHz (`src236_mode = SRC_RATIO_3_2`) with a scheduling of 10 ms.

Figure 1. Example of re-sampling



3.2 Data formats

The module supports fixed point data, in Q15 or Q31 format with a mono or a stereo interleaved pattern.

The input buffer size depends in fact on the output buffer size and ratio; thus, the output buffer size divided by the SRC ratio must be an integer value. It is also recommended not to run processing below 2 ms scheduling to optimize the MIPS; here, the baseline is a 5 ms scheduling task.

3.3 Performance measurements

The performance is measured using fixed point Matlab model, which is representative of the implemented fixed point code.

3.3.1 SINAD measurements

Quality measurement is done on 16-bit input signals with 16-bit I/O library versions, and on 32-bit input signal (derived from a 24-bit input signal) with 32-bit I/O library versions.

THDN (Total Harmonic Distortion + Noise) corresponds to the reverse of SINAD (Signal to Noise And Distortion ratio) in case of a pure frequency tone input. The measurements below estimate that the SRC quality follows the AES 17-1998 (r2004) recommendations, by injecting a sine-wave, filtering the output with a standard notch filter (quality factor Q = 5), and the following computing ratio :

$$SINAD = \frac{1}{THDN} = \frac{(\text{Power of input})}{(\text{Power of notched filtered output})}$$

Table 13 summarizes SINAD values in dB depending on the re-sampling ratio and input tone frequency.

Table 13. SINAD values

	Tests	Amp (dBFS)	-0.1	-1	-1	-1	-1	-1	-1	-1	FS x 0.45
		Freq (kHZ)	997	40	160	640	1280	2560	5120	10240	
Standard	Ratio 2	8 to 16	94.4	93	94.8	94.3	94.6	94.2	N/A	N/A	46.4
	Ratio 3	16 to 48	94.4	93.4	94.4	94.4	93.9	94.9	94.8	N/A	46.7
	Ratio 6	8 to 48	94.8	93.7	95.8	94.8	94.9	94.6	N/A	N/A	46.9
	Ratio 3/2	32 to 48	94.6	93.5	93	95.8	94.4	93.9	95	94.8	46.2
	Ratio 1/2	16 to 8	96.8	94.9	96.1	98.1	95.8	97.3	N/A	N/A	101.9
	Ratio 1/3	48 to 16	97	96	96	97.2	97.1	95.6	95.7	N/A	99.3
	Ratio 1/6	48 to 8	97.5	97	97.7	97.5	97	97.8	N/A	N/A	101.9
	Ratio 2/3	24 to 16	95.4	94.2	94.5	94	94.7	96.6	95.7	N/A	99.3

Table 13. SINAD values (continued)

	Tests	Amp (dBFS)	-0.1	-1	-1	-1	-1	-1	-1	-1	FS x 0.45
		Freq (kHZ)	997	40	160	640	1280	2560	5120	10240	
High quality	Ratio 2	8 to 16	94.4	93	94.8	94.3	94.6	94.2	N/A	N/A	46.4
	Ratio 3	16 to 48	94.4	93.4	94.4	94.4	93.9	94.9	94.8	N/A	46.7
	Ratio 6	8 to 48	94.8	93.7	95.8	94.8	94.9	94.6	N/A	N/A	46.9
	Ratio 3/2	32 to 48	94.6	93.5	93	95.8	94.4	93.9	95	94.8	46.2
	Ratio 1/2	16 to 8	96.8	94.9	96.1	98.1	95.8	97.3	N/A	N/A	101.9
	Ratio 1/3	48 to 16	97	96	96	97.2	97.1	95.6	95.7	N/A	99.3
	Ratio 1/6	48 to 8	97.5	97	97.7	97.5	97	97.8	N/A	N/A	101.9
	Ratio 2/3	24 to 16	95.4	94.2	94.5	94	94.7	96.6	95.7	N/A	99.3
Standard 32-bit I/O	Ratio 2	8 to 16	100.6	99.8	100.6	100.3	101	99.9	N/A	N/A	96
	Ratio 3	16 to 48	100.8	100.4	100.6	100.5	100.8	101.1	100.5	N/A	111.2
	Ratio 6	8 to 48	103	102.8	103	102.9	103.4	102	N/A	N/A	97.4
	Ratio 3/2	32 to 48	100.8	100.4	100.4	100.9	100.5	100.8	101.1	100.5	46.2
	Ratio 1/2	16 to 8	144.8	144.2	146.1	145.8	143.9	146.8	N/A	N/A	109.5
	Ratio 1/3	48 to 16	145.2	144.2	144.2	145	148.1	145.1	145.8	N/A	93.1
	Ratio 1/6	48 to 8	145.8	146	147.8	145.9	146.3	144.3	N/A	N/A	111.2
	Ratio 2/3	24 to 16	103.6	103	103.4	103.3	104	103.5	104.3	N/A	96.7
High quality 32-bit I/O	Ratio 2	8 to 16	117.5	119.3	119.3	119.6	123.9	120.1	N/A	N/A	111.2
	Ratio 3	16 to 48	122.5	120.9	122.1	122.5	120.9	123.4	122.4	N/A	118.5
	Ratio 6	8 to 48	120	120.8	121	120.3	121.8	122.5	N/A	N/A	118.8
	Ratio 3/2	32 to 48	121.3	120.7	121.2	121.3	122.5	120.9	123.4	122.4	118.1
	Ratio 1/2	16 to 8	144.7	144.1	145.4	143.6	145.7	143.7	N/A	N/A	146
	Ratio 1/3	48 to 16	145.1	143.8	144.8	146.9	143.9	143.6	143.3	N/A	146.5
	Ratio 1/6	48 to 8	145.8	144.5	145	146.7	143.9	145.7	N/A	N/A	153.4
	Ratio 2/3	24 to 16	127	131.1	127.4	127.6	129.3	126.7	129.9	N/A	125.2

Note: No windowing is applied in measurements above. A-Law usage shows a gain of about 2-3 dB between 600 Hz and 8 kHz and a loss of several dBs outside this range due to the A-Law shape that brings the signal to analyze closer to the noise floor.

3.3.2 Aliasing measurements

Alias analysis is only used for down-samplers to measure aliasing attenuation.

The measurement method is similar to the SINAD measurement, but with input frequency tones above the output sampling frequency.

The tables below summarize the Alias attenuation values in dB depending on the re-sampling ratio and the input tone frequency.

Table 14. Alias attenuation values (dBFS) - Ratio 1/2: 16 to 8 kHZ

Tests	Amp (dBFS)	-1	-1	-1	-1	-1	-1	-1	-1
	Freq (Hz)	4080	4200	4400	4840	5324	5856	6442	7086
Standard	Ratio 1/2: 16 to 8	24.6	31.4	46	100.2	99.3	98.2	99.6	99.2
High quality		29.3	46.5	Inf	Inf	108.5	Inf	112.3	110.2
Standard 32-bit I/O		24.6	31.4	46	101.4	100.8	100.5	100.8	100.8
High quality 32-bit I/O		29.3	46.5	118.8	122	119.8	116.8	117.1	121.9

Table 15. Alias attenuation values (dBFS) - Ratio 1/3: 48 to 16 kHZ

Tests	Amp (dBFS)	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	Freq (Hz)	8160	8400	8800	9680	10648	11713	12884	14172	15590	17149	18864
Standard	Ratio 1/3: 48 to 16	24.6	31.4	46	Inf	108.5	108.6	108.1	108.1	108.3	108.2	113.2
High quality		29.3	46.5	Inf	Inf	Inf	118.3	119.3	114.5	115.3	120.5	Inf
Standard 32-bit I/O		24.6	31.4	46	106.2	104.7	104.2	105.2	105.2	103.9	103.7	104.2
High quality 32-bit I/O		29.3	46.5	118	120.1	122.3	119.7	120.5	127.3	122.9	123.9	130.5

Table 16. Alias attenuation values (dBFS) - Ratio 1/6: 48 to 8 kHZ

Tests	Amp (dBFS)	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
	Freq (Hz)	4080	4200	4400	4840	5324	5856	6442	7086	7795	8574	9432
Standard	Ratio 1/6: 48 to 8 kHz	24.6	31.4	46	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
High quality		29.3	46.5	Inf	Inf	Inf	Inf		"Inf	Inf	Inf	Inf
Standard 32-bit I/O		24.6	31.4	46	112.6	111.3	108.9	111.2	110.5	110.1	110.1	109.4
High quality 32-bit I/O		29.3	46.5	118	119.6	124.9	121	121.8	131.5	122.9	123.3	136.6

Table 17. Alias attenuation values (dBFS) - Ratio 2/3: 24 to 16 kHZ

Tests	Amp (dBFS)	-1	-1	-1	-1	-1
	Freq (Hz)	8160	8400	8800	9680	10648
Standard	Ratio 2/3: 24 to 16 kHz	24.6	31.4	46	97.8	97.7
High quality		29.3	46.5	Inf	106.3	104.5
Standard 32-bit I/O		24.6	31.4	46	102.5	101.9
High quality 32-bit I/O		29.3	46.5	116.7	117.6	120.5

Note: "Inf" means that the aliasing attenuation is so strong (> 115 dB) that the measurement cannot be done.

3.3.3 Frequency response measurements

The frequency response analysis gives information on in-band ripple, frequency cut at -1 and -3 dB and filter group delay (the filters used have a linear phase).

Table 18 summarizes the data with a standard version.

Table 18. Frequency response analysis (dB) with a standard version

Tests	Freq (kHZ)	Max. ripple (dB)	Min. ripple (dB)	Frequency cut at -1 dB	Frequency cut at -3 dB	Filter group delay (ms)
Ratio 2	8 to 16 kHZ	0.17	-0.07	3173	3362	1.25
Ratio 3	16 to 48 kHZ			6346	6724	0.62
Ratio 6	8 to 48 kHZ			3173	3362	1.25
Ratio 3/2	32 to 48 kHZ			12692	13449	0.31
Ratio 1/2	16 to 8 kHZ			3173	3362	0.62
Ratio 1/3	48 to 16 kHZ			6346	6724	0.21
Ratio 1/6	48 to 8 kHZ			3173	3362	0.21
Ratio 2/3	24 to 16 kHZ			6346	6724	0.42

Table 19 summarizes the data with a high quality version.

Table 19. Frequency response analysis (dB) with a high quality version

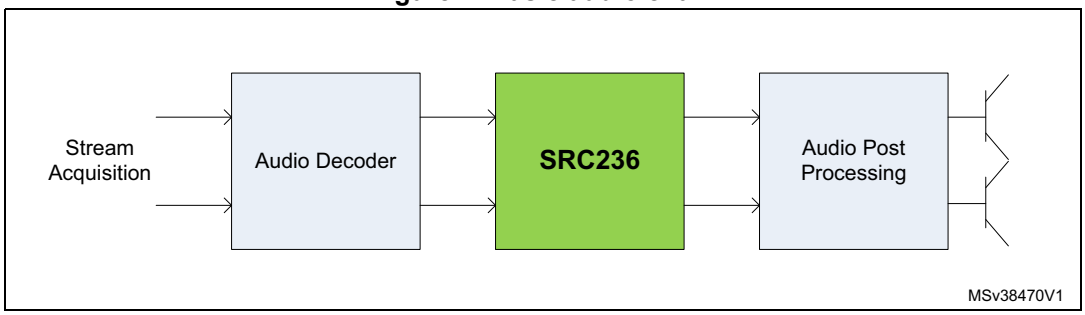
Tests	Freq (kHZ)	Max. ripple (dB)	Min. ripple (dB)	Frequency cut at -1 dB	Frequency cut at -3 dB	Filter group delay (ms)
Ratio 2	8 to 16 kHZ	0.18	-0.08	3591	3683	2.5
Ratio 3	16 to 48 kHZ			7181	7366	1.3
Ratio 6	8 to 48 kHZ			3591	3683	2.5
Ratio 3/2	32 to 48 kHZ			14363	14732	0.62
Ratio 1/2	16 to 8 kHZ			3591	3683	1.3
Ratio 1/3	48 to 16 kHZ			7181	7366	0.42
Ratio 1/6	48 to 8 kHZ			3591	3683	0.42
Ratio 2/3	24 to 16 kHZ			7181	7366	0.83

4 System requirements and hardware setup

SRC236 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. There is no other hardware dependency.

4.1 Recommendations for optimal setup

Figure 2. Basic audio chain



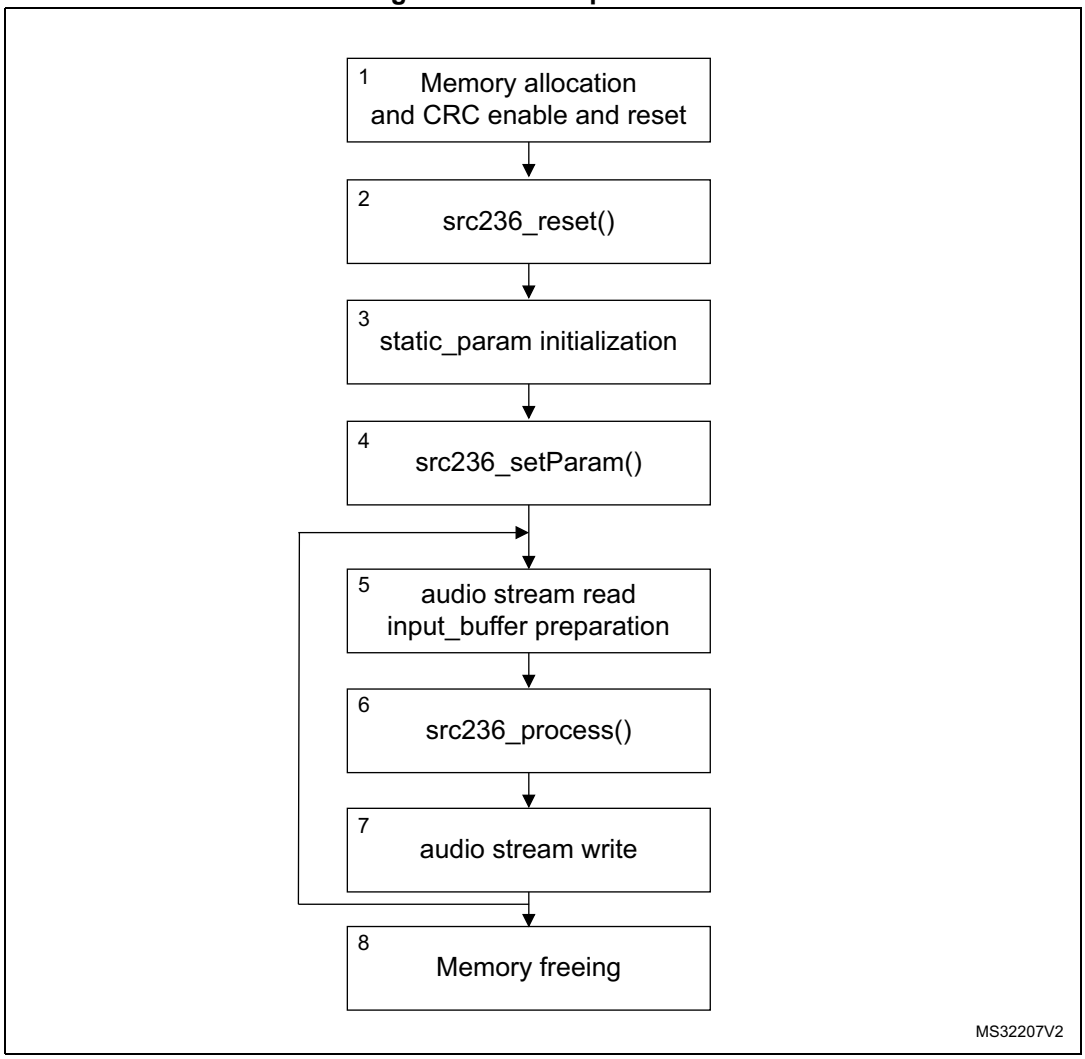
The sampling rate conversion algorithm should be placed quite early in the audio chain, for instance just after the audio decoder in order to get all the audio streams at the same sampling frequency (ideally, the highest value, 48 kHz, should be used). If needed, streams can be mixed now, or a post-processing can be applied. Samples are then played on the audio DAC.

4.1.1 Module integration example

Cube expansion SRC236 integration examples are provided on STM32746G-Discovery and STM32469I-Discovery boards. Please refer to provided integration code for more details.

4.1.2 Module integration summary

Figure 3. API call procedure



1. As explained above, the SRC236 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*. Also, SRC236 library must run on STM32

devices so CRC HW block must be enable and reset.

2. Once the memory is allocated, the call to the src236_reset() function initializes the internal variables.
3. The SRC236 configuration for the desired SRC ratio can now be set by initializing the static_param structure, once the input sampling frequency is known.
4. Call the src236_setParam() routine to configure the SRC with the right ratio.
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. Call the processing main routine to re-sample the stream in the output buffer.
7. The output audio stream can now be written in the proper interface.
8. Once the processing loop is over, the allocated memory has to be freed.

Note: Since there is no dynamic parameter, src236_setConfig() and src236_getConfig() are not used.

5 How to tune and run the application

The only parameter which is static represents the SRC ratio, so there is no tuning available for the SRC236 module.

The only available choice is to link SRC236_xxx_CMy_zzz.a/.lib library version to the *src236_glo.h* header file and set the static src_mode parameter.

Once the module is integrated into the audio framework to play samples for instance at 48 kHz, launch the Audio player with an 8, 16, 24, 32, 48 or 96 kHz input sampling frequency file. The output file will be decoded and played at 48 kHz without returning any error message.

6 Revision history

Table 20. Document revision history

Date	Revision	Changes
17-June-2013	1	Initial release.
26-Nov-2014	2	Updated RPN reference on cover page, *Table 18* and *Table 19*.
11-Jan-2016	3	Updated: – *Introduction* – *Module configuration* – *Section 3.1* – *Section 5* Added: – *Section 1.3* – *Figure 2* – *Section 4.1.1*
02-Feb-2016	4	Updated *Section 5*
31-Mar-2017	5	Updated: – *Section 1.2: Module configuration*, *Section 1.3: Resource summary*, *Section 2: Module interfaces*, *Section 2.1.1: src236_reset function*, *Section 2.1.2: src236_setParam function*, *Section 2.1.3: src236_getParam function*, *Section 2.1.4: src236_setConfig function*, *Section 2.1.5: src236_getConfig function*, *Section 2.1.6: src236_process function*, *Section 4.1.1: Module integration example*, *Section 5: How to tune and run the application* – *Table 2: Resource summary*, *Table 3: src236_reset*, *Table 4: src236_setParam*, *Table 5: src236_getParam*, *Table 6: src236_setConfig*, *Table 7: src236_getConfig*, *Table 8: src236_process*, *Table 18: Frequency response analysis (dB) with a standard version*, *Table 19: Frequency response analysis (dB) with a high quality version*
10-Jan-2018	6	Replaced X-CUBE-AUDIO-F4, X-CUBE-AUDIO-F7 and X-CUBE-AUDIO-L4 with X-CUBE-AUDIO.