UM1640
User manual

Sampling rate conversion SRC441 library software expansion for STM32Cube

Introduction

The sampling rate conversion SRC441 library user manual describes the software interface and its requirements. It describes how to integrate the module into a main program, like the audio STM32Cube expansion software. It also provides a basic understanding of the underlying algorithm.

The SRC441 library is used to convert the sampling frequency from 44.1 kHz to 48 kHz.

The SRC441 library is part of X-CUBE-AUDIO firmware package.

Contents

List of tables

List of figures

1 Module overview

1.1 Algorithm functionality

The SRC441 module provides functions to handle the sampling rate conversion of mono and stereo signals from 44.1 kHz to 48 kHz.

Note: Conversions with other ratios are handled by another module named SRC236.

1.2 Module configuration

The SRC441 module supports mono and stereo interleaved 16-bit or 32-bit I/O data, with an input frame size of 147 samples (corresponding to 3.33 ms).

If needed, the processing can be looped three times to re-sample exactly 10 ms of data.

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

- SRC441_CM4_IAR.a / SRC441_CM4_GCC.a / SRC441_CM4_keil.lib:
  Standard configuration for low-MIPS and good quality requirements with 16-bit input/output buffers. It runs on any STM32 microcontroller featuring an Arm® core with Cortex®-M4 instruction set.
- SRC441HQ_CM4_IAR.a / SRC441HQ_CM4_GCC.a / SRC441HQ_CM4_keil.lib:
  Reserved for high quality needs (consumes more MIPS and memory as well) with 16-bit input/output buffers. It runs on any STM32 microcontroller featuring an Arm®core with Cortex®-M4 instruction set.
- SRC441_32b_CM4_IAR.a / SRC441_32b_CM4_GCC.a / SRC441_32b_CM4_keil.lib:
  Standard configuration for low-MIPS and good quality requirements with 32-bit input/output buffers. It runs on any STM32 microcontroller featuring an Arm® core with Cortex®-M4 instruction set.
- SRC441HQ_32b_CM4_IAR.a / SRC441HQ_32b_CM4_GCC.a / SRC441HQ_32b_CM4_keil.lib:
  Reserved for high quality needs (consumes more MIPS and memory as well), with 32-bit input/output buffers. It runs on any STM32 microcontroller featuring an Arm® core with Cortex®-M4 instruction set.
- SRC441_CM7_IAR.a / SRC441_CM7_GCC.a / SRC441_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 an Arm® core with Cortex®-M7 instruction set.
- SRC441HQ_CM7_IAR.a / SRC441HQ_CM7_GCC.a / SRC441HQ_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 an Arm® core with Cortex®-M7 instruction set.
- SRC441_32b_CM7_IAR.a / SRC441_32b_CM7_GCC.a / SRC441_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 an Arm® core with Cortex®-M7 instruction set.
- SRC441HQ_32b_CM7_IAR.a / SRC441HQ_32b_CM7_GCC.a / SRC441HQ_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 an Arm® core with Cortex®-M7 instruction set.

1.3 Resource summary

Table 1 contains the module requirements for the Flash, stack and RAM memories and frequency (MHz).

Table 1. Resource summary

Version	User Case	Core	Flash code (.text)	Flash data (.rodata)	Stack	Persistent RAM	Scratch RAM(1)	Frequency (MHz)
Standard	Mono	M4	3184 Bytes	8 Bytes	80 Bytes	232 Bytes	3228 Bytes	9.9
		M7	3442 Bytes					5.4
	Stereo	M4	3184 Bytes					14
		M7	3442 Bytes					7.5
High quality	Mono	M4	3878 Bytes	8 Bytes	80 Bytes	360 Bytes	3676 Bytes	15.3
		M7	3848 Bytes					7.9
	Stereo	M4	3878 Bytes					20.7
		M7	3848 Bytes					10.8
Standard 32-bit I/O	Mono	M4	3204 Bytes	8 Bytes	80 Bytes	232 Bytes	3228 Bytes	9.8
		M7	3678 Bytes					5.3
	Stereo	M4	3204 Bytes					13.7
		M7	3678 Bytes					7.3
High quality 32-bit I/O	Mono	M4	3894 Bytes	8 Bytes	80 Bytes	360 Bytes	3676 Bytes	15.2
		M7	4096 Bytes					7.9
	Stereo	M4	3894 Bytes					20.4
		M7	4096 Bytes					10.7

1. 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 SRC441 routines.

Note: The footprints are measured on board, using IAR Embedded Workbench for Arm® v7.40 (IAR Embedded Workbench common components v7.2). The footprints on STM32F7 are measured with stack and heap sections located in DTCM memory.

2 Module Interfaces

Two files are needed to integrate the SRC441 module: SRC441_xxx_CMy_zzz.a/.lib library and the *src441_glo.h* header file. They contain 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:

- src441_reset
- src441_setParam
- src441_getParam
- src441_setConfig
- src441_getConfig
- src441_process

2.1.1 src441_reset function

This procedure initializes the persistent memory of the SRC441 module and initializes static and dynamic parameters with default values.

```
int32_t src441_reset(void *persistent_mem_ptr, void *scratch_mem_ptr);
```

Table 2. src441_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 src441_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).

```
int32_t src441_setParam(src441_static_param_t *input_static_param_ptr, void
*persistent_mem_ptr);
```

Table 3. src441_setParam

I/O	Name	Type	Description
Input	input_static_param_ptr	src441_static_param_t*	Pointer to static parameters structure
Input	persistent_mem_ptr	void *	Pointer to internal persistent memory
Returned value	-	int32_t	Error value

Note: There is currently no static parameter, so no reason to call this routine in this module version.

2.1.3 src441_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 (i.e. the parameters with values which cannot be changed during the module processing).

```
int32_t src441_getParam(src441_static_param_t *input_static_param_ptr, void
*persistent_mem_ptr);
```

Table 4. src441_getParam

I/O	Name	Type	Description
Input	input_static_param_ptr	src441_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 src441_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).

```
int32_t src441_setConfig(src441_dynamic_param_t *input_dynamic_param_ptr,
void *persistent_mem_ptr);
```

Table 5. src441_setConfig

I/O	Name	Type	Description
Input	input_dynamic_param_ptr	src441_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: There is currently no dynamic parameter, so no reason to call this routine in this module version.

2.1.5 src441_getConfig function

This procedure gets module 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 src441_getConfig(src441_dynamic_param_t *input_dynamic_param_ptr,
void *static_mem_ptr);
```

Table 6. src441_getConfig

I/O	Name	Type	Description
Input	input_dynamic_param_ptr	src441_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 src441_process function

This procedure is the module's main processing routine.

It should be called at any time, to process each frame.

```
int32_t src441_process(buffer_t *input_buffer, buffer_t *output_buffer,
void *persistent_mem_ptr);
```

Table 7. src441_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 cannot run in place; the input_buffer data is modified during processing, thus it cannot be used as it is after any call to the src441_process() routine.

2.2 External definitions and types

2.2.1 Input and output buffers

The SRC441 library uses 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 each time, before calling the processing routine, otherwise an error 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 8. 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	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 9 lists the possible returned error values:

Table 9. Returned error values

Definition	Value	Description
SRC441_ERROR_NONE	0	OK - no error detected
SRC441_UNSUPPORTED_MODE	-1	If input data is not interleaved
SRC441_BAD_FRAME_SIZE	-2	If the number of input samples is not 147
SRC441_WRONG_NBBYTES_PER_SAMPLES	-3	Input data is neither 16-bit nor 32-bit value
SRC441_UNSUPPORTED_NB_CHANNELS	-4	Input data is neither mono nor stereo
SRC441_UNSUPPORTED_INPLACE_PROCESSING	-5	If input and output buffers are not different
SRC441_BAD_HW	-6	May happen if the library is not used with the right hardware

2.3 Static parameters structure

There is no static parameter to be used.

For compatibility with other structures, the static parameter structure contains a dummy field.

```
struct src441_static_param {
    int32_t empty;
}
typedef struct src441_static_param src441_static_param_t;
```

Table 10. Static parameters structure

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

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 src441_dynamic_param {
    int32_t empty;
}
typedef struct src441_dynamic_param src441_dynamic_param_t;
```

Table 11. Dynamic parameters structure

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

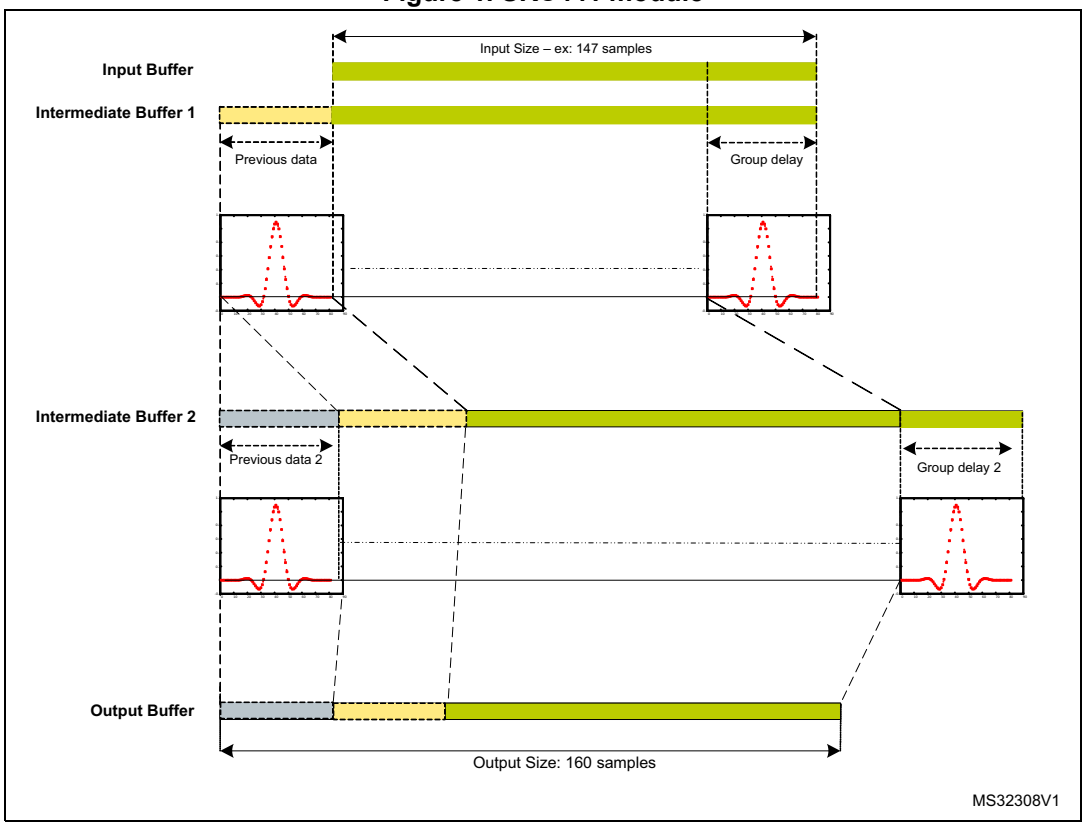
3 Algorithm high level view

3.1 Processing steps

The SRC441 module is a re-sampler based on a two-stage polyphase filter. This implementation has been optimized for Cortex® M4 and M7 cores using SIMD instructions set.

Figure 1 shows an example of re-sampling from 44.1 kHz to 48 kHz with a scheduling of 3.33 ms.

Figure 1. SRC441 module



3.2 Data formats

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

The input buffer size is fixed at 147 samples and will generate 160 output samples per frame.

3.3 Performance measurements

3.3.1 SINAD measurements

Quality measurement is done on a 16-bit input signal with 16-bit I/O library versions and on a 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 = 1/THDN = (Power of Input) / (Power of notched filtered output)

Table 12 summarizes the SINAD values in dB.

Table 12. SINAD values

Tests	Amp (dBFS)	-0.1	-1	-1	-1	-1	-1	-1	-1	-1
	Freq (Hz)	997	40	160	640	1280	2560	5120	10240	15997
Standard	44.1 to 48 kHz	95.2	94.2	94.3	94.1	94.0	94.1	94.5	94.6	94.6
High quality		95.4	94.4	94.2	94.0	94.1	94.5	94.8	95.5	94.8
Standard, 32 bits		107.1	107.2	107.2	107.1	107.2	108	108.7	107.1	107.6
High quality, 32 bits		131	132.4	132.4	131.5	131.1	132.7	132.3	131.2	126

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

3.3.2 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 13* summarizes the data with a standard or a high quality version.

Table 13. Frequency response values

Tests	Freq (Hz)	Max. ripple (dB)	Min. ripple (dB)	Frequency cut at -1 dB	Frequency cut at -3 dB	Filter group delay (ms)
Standard	44.1 to 48 kHz	0.1	-0.07	16200	17300	0.3
High quality		0.13	-0.05	17400	18400	0.47
Standard, 32 bits		0.1	-0.07	16200	17300	0.3
High quality, 32 bits		0.13	-0.05	17400	18400	0.47

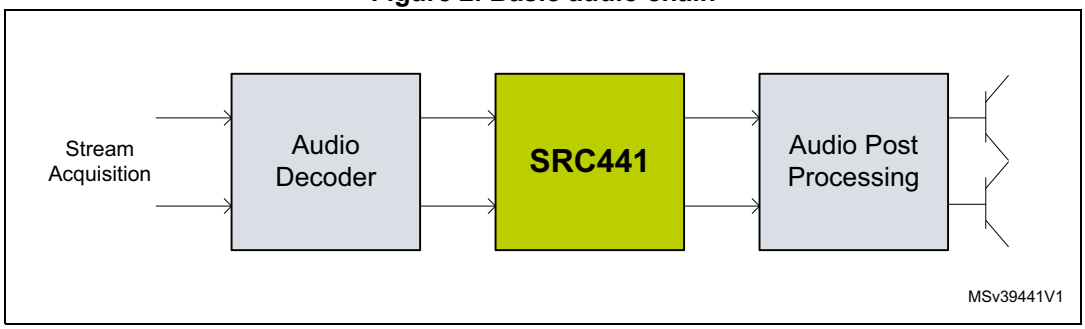
4 System requirements and hardware setup

SRC441 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 microcontrollers of STM32F4/STM32L4 or STM32F7 series, respectively. There is no other hardware dependency.

4.1 Recommendations for optimal setup

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. If needed, streams can be mixed now, or a post-processing can be applied. Samples are then played on the audio DAC. Refer to *Figure 2: Basic audio chain*.

Figure 2. Basic audio chain

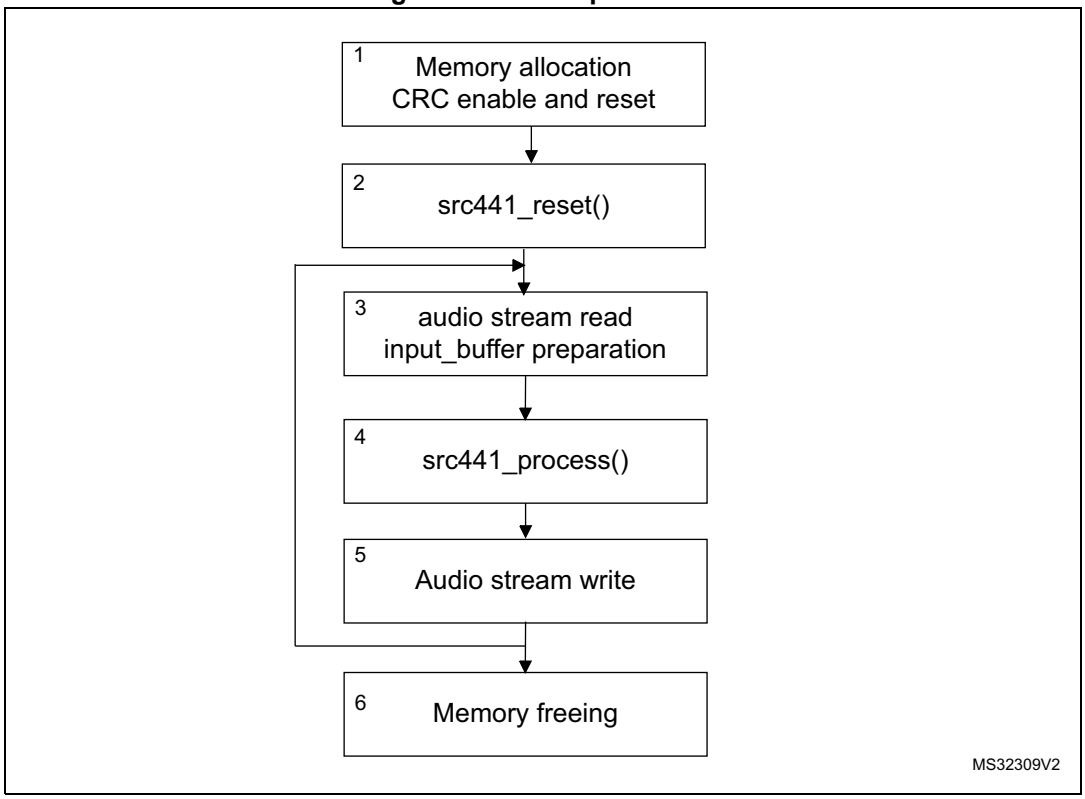


4.1.1 Module integration example

Cube expansion SRC441 integration examples are provided on STM32F746G-Discovery and STM32F469I-Discovery boards. Refer to provided integration code for more details.

4.1.2 Module integration summary

Figure 3. API call procedure



1. As explained above, SRC441 scratch and persistent memories have to be allocated, as well as the input and output buffer. Also, SRC441 library must run on STM32 devices so that CRC HW block must be enable and reset.
2. Once the memory has been allocated, the call to src441_reset() function will initialize the internal variables.
3. 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.
4. A call to the process will re-sample the stream in the output buffer.
5. The output audio stream can now be written in the proper interface.
6. Once the processing loop is over, the allocated memory has to be freed.

5 How to tune and run the application

There is no tuning available for the SRC441 module.

The only available choice is to link the right SRC441_xxx_CMy_zzz.a/.lib library with the src441_glo.h header file.

Once the module has been integrated into an audio framework to play samples at 48kHz, launch a player with a 44.1 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 14. Document revision history

Date	Revision	Changes
7-Jun-2013	1	Initial release.
26-Aug-2013	2	Changed all “stereo” occurrences into “mono and stereo”. Added “mono” values to *Table 1: Resource summary*.
28-Nov-2014	3	Classification changed from ST Restricted to public. Replaced the reference STSW-STM32APP by STM32-AUDIO100A.
08-Jan-2016	4	Updated: – *Table 1: Resource summary* – *Section 4.1.1: Module integration example* – *Section 5: How to tune and run the application* – *Figure 3: API call procedure* Added: – *Figure 2: Basic audio chain*
21-Jan-2016	5	Updated *Section 5: How to tune and run the application*
09-Jan-2018	6	Updated: – *Introduction* – *Section 1.2: Module configuration* – *Section 2: Module Interfaces* – *Section 4.1.1: Module integration example* – *Section 5: How to tune and run the application* – *Table 1: Resource summary* – *Table 13: Frequency response values*

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved