SSRCLIBUG

Synchronous Sample Rate Converter

Rev. 2.14.0 — 27 July 2023 **User guide**

Document Information

Information	Content
Keywords	Synchronous Sample Rate Converter, SSRC, Synchronous, Sample Rate
Abstract	The Synchronous Sample Rate Converter (SSRC) software module converts an audio signal (mono or stereo) with a certain sampling frequency to an audio signal with another sampling frequency.

1 Introduction

The Synchronous Sample Rate Converter (SSRC) software module converts a mono or stereo audio signal with a certain sampling frequency to an audio signal with a different sampling frequency. The sample rate converter works synchronously, meaning that input and output sampling rates are exactly known for a mutual clock reference.

To accomplish a professional sampling conversion quality and minimal system footprint, the SRC SW module contains highly optimized components.

The SSRC module supports the following features.

- Multiple instances of the sample rate converter can run at the same time.
- Supported sampling frequencies: 32 kHz, 44.1 kHz, and 48 kHz plus the halves and the quarters of these three sample rates. The input and outputsample rates are freely selectable out of the supported sampling rates
- Selectable Mono/Stereo Input/Output.
- Selectable quality level: high quality/ very high quality.

1.1 Acronyms

Table 1 lists the acronyms used in this document.

Table 1. Acronyms

Acronym	Description
Fs	Sampling Frequency
Fs_{LOWow}	Lowest sample rate used for the conversion Input sample rate for up sampling and the output sample rate for down sampling
Fs_{IN}	Input sample rate
Fs_{OUT}	Output sample rate
MIPS	Million Instructions Per Second
SSRC	Synchronous sample rate converter
THD+N	Total Harmonic Distortion plus Noise ***Note:*** *The THD+N is defined as the total power of the unwanted signal divided by the power of the wanted signal. The wanted signal is defined as a full scale, 1 kHz sine wave.*

1.2 Performance figures

The Total Harmonic Distortion Plus Noise (THD+N) of the converted signals is below - 76 (high-quality mode) and - 85 (very high-quality mode) for signal frequencies below 0.45*Fs_{LOW} (=90 % of the Nyquist range of the lowest sample clock)

Table 2 and Table 3 give the THD+N performance (Fs_{IN} on the vertical axis and Fs_{OUT} on the horizontal axis) for the two supported quality levels. The numbers in the tables give the worst-case THD+N measured for signal frequencies below 0.45*Fs_{LOW}. For each conversion ratio, 100 THD+N measurements were executed with signal frequencies linearly spread over the complete Nyquist range.

Table 2. THD+N performance: Quality level = High

Fs_{IN}/ Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	-92.1	-79.7	-80.1	-80.1	-79.6	-80.2	-79.4	-79.1	-79.2

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Table 2. THD+N performance: Quality level = High*...continued*

Fs_{IN}/ Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
11025	-79	-92.9	-80	-79.9	-80.2	-79.8	-79.9	-79.5	-78.9
12000	-79	-79.2	-92.7	-80.1	-79.8	-80.3	-79.8	-79.8	-79.5
16000	-81.7	-78.8	-80.2	-93	-78.3	-77.7	-78.3	-78.3	-77.9
22050	-77.5	-81.8	-78.2	-79	-93	-79.9	-79.8	-80.3	-79.9
24000	-77.4	-77.9	-81.2	-79.1	-79.2	-92.5	-80.1	-79.8	-79.9
32000	-81	-77.5	-78.9	-81.2	-78.7	-80.1	-92.9	-79.7	-79.2
44100	-79.1	-81.2	-76.7	-77.8	-82	-78.2	-79.1	-93	-79.7
48000	-78.7	-78.8	-81.1	-77.6	-77.9	-81.8	-79.1	-79.3	-93

Table 3. THD+N performance: Quality level = Very high

Fs_{IN}/ Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	-92.1	-86.6	-88.6	-91.5	-86.4	-89	-89.7	-89.3	-89.3
11025	-89.1	-92.9	-86.3	-86.3	-91.6	-86.3	-86.5	-89.7	-89.3
12000	-91.4	-88.4	-92.7	-89.6	-86.6	-91.5	-86.8	-86.6	-89.7
16000	-93.1	-88.4	-90.4	-93	-86.6	-88.8	-91.5	-86.5	-89.4
22050	-90.7	-93.5	-89.7	-89.3	-93	-86.5	-86.3	-91.5	-86.6
24000	-93.8	-90.5	-93.5	-91.7	-88.4	-92.5	-89.7	-86.6	-91.5
32000	-93.8	-91	-91.2	-93.3	-88.4	-90.5	-92.9	-86.7	-89
44100	-93.7	-93.6	-91.5	-90.6	-93.8	-89.8	-89.3	-93	-86.5
48000	-94.1	-92.6	-94	-94	-90.1	-93.7	-91.8	-88.4	-93

1.3 Resource usage

This section lists the memory and processing requirements for the SSRC module.

1.3.1 Memory requirements

The following are the memory requirements for the SSRC module.

Table 4. Memory requirements

Memory item	Size in bytes
Instance memory (persistent)	548
Scratch memory (non-persistent)	15.536 [1]
Program memory for Arm9E and XScale	14k
Program memory for Arm7	15k

[1] Worst case number for I/O buffers of 40 ms. If smaller I/O buffers are used, this number is smaller. The required scratch memory is roughly equal to 2 times the buffer size on the highest sample rate.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

1.3.2 Processing requirements

The following tables give the MIPS performance of the SSRC module. The cycles are measured with zero wait state memory and for I/O buffers of 40 ms.

Note: *The user processing 32-bit processing must refer to the very high-quality MIPS results.*

1.3.2.1 On Arm7 and Arm9

Table 5. MIPS for stereo at high quality

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	0.13	4.77	5.17	1.84	6.75	7.33	3.55	9.1	9.89
11025	5.42	0.18	5.58	6.84	2.53	7.75	9.71	4.89	10.31
12000	5.85	6.39	0.2	7.01	8.97	2.76	9.89	12.94	5.32
16000	1.69	7.74	7.99	0.26	9.54	10.33	3.68	13.5	14.65
22050	7.2	2.33	10.09	10.83	0.36	11.17	13.67	5.07	15.49
24000	7.79	8.33	2.53	11.7	12.78	0.39	14.03	17.94	5.51
32000	3.12	10.32	10.58	3.38	15.48	15.98	0.52	19.08	20.66
44100	9.96	4.3	13.65	14.4	4.65	20.18	21.67	0.72	22.34
48000	10.8	11.34	4.68	15.58	16.67	5.06	23.4	25.56	0.78

Table 6. MIPS for mono at very high quality

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	0.07	7.71	8.24	2.28	10.5	11.28	4.41	13.44	14.48
11025	8.19	0.1	8.96	11.04	3.14	12	15.09	6.08	15.2
12000	8.76	9.52	0.1	11.3	14.48	3.41	15.36	20.07	6.61
16000	2.14	11.73	12.01	0.14	15.41	16.48	4.55	21	22.56
22050	10.78	2.94	15.39	16.38	0.19	17.92	22.08	6.27	24
24000	11.57	12.34	3.2	17.51	19.04	0.21	22.61	28.97	6.83
32000	4.19	15.48	15.77	4.27	23.46	24.01	0.28	30.83	32.96
44100	14.78	5.77	20.56	21.56	5.89	30.77	32.75	0.38	35.83
48000	15.92	16.7	6.28	23.15	24.69	6.41	35.02	38.08	0.42

Table 7. MIPS for stereo at very high quality

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	0.13	13.61	14.52	4.43	19.03	20.43	8.8	25.06	26.99
11025	14.85	0.18	15.91	19.47	6.1	21.82	27.35	12.13	28.38
12000	15.84	17.36	0.2	19.97	25.4	6.64	27.85	36.26	13.21
16000	4.25	21.24	21.79	0.26	27.22	29.03	8.86	38.07	40.85
22050	20.02	5.85	27.72	29.7	0.36	31.81	38.94	12.2	43.63
24000	21.45	22.98	6.37	31.68	34.71	0.39	39.94	50.8	13.28
32000	8.39	28.74	29.29	8.5	42.48	43.58	0.52	54.43	58.07
44100	28.11	11.57	38.05	40.03	11.71	55.43	59.4	0.72	63.62

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Table 7. MIPS for stereo at very high quality*...continued*

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
48000	30.19	31.71	12.59	42.9	45.96	12.74	63.36	69.42	0.78

1.3.2.2 On Arm9e and XScale

Table 8. MIPS for mono at high quality

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	0.03	1.14	1.25	0.54	1.95	2.14	1.04	3.85	4.23
11025	1.31	0.05	1.36	1.62	0.75	2.23	2.78	1.44	4.38
12000	1.43	1.57	0.05	1.68	2.13	0.82	2.84	3.72	1.57
16000	0.5	1.86	1.93	0.07	2.27	2.5	1.09	3.9	4.29
22050	2.19	0.69	2.42	2.61	0.1	2.72	3.24	1.5	4.46
24000	2.4	2.52	0.75	2.86	3.15	0.1	3.35	4.25	1.63
32000	0.92	3.12	3.18	1.01	3.72	3.86	0.14	4.55	4.99
44100	4.28	1.27	4.15	4.37	1.39	4.83	5.23	0.19	5.43
48000	4.7	4.9	1.39	4.8	5.03	1.51	5.72	6.3	0.21

Table 9. MIPS for stereo at high quality

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	0.06	1.87	2.02	1.07	3.09	3.36	2.07	6.09	6.63
11025	2.27	0.09	2.25	2.66	1.47	3.56	4.4	2.85	7.01
12000	2.45	2.76	0.09	2.75	3.43	1.6	4.5	5.83	3.1
16000	0.99	3.23	3.36	0.13	3.73	4.05	2.14	6.17	6.72
22050	3.69	1.36	4.14	4.55	0.17	4.51	5.31	2.95	7.13
24000	4.01	4.28	1.48	4.9	5.51	0.19	5.51	6.85	3.21
32000	1.83	5.26	5.39	1.98	6.46	6.71	0.25	7.47	8.09
44100	7.22	2.52	6.94	7.38	2.72	8.27	9.1	0.35	9.02
48000	7.85	8.33	2.74	8.02	8.57	2.97	9.81	11.03	0.38

Table 10. MIPS for mono at very high quality

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	0.03	1.21	1.33	0.61	2.08	2.29	1.17	4.1	4.51
11025	1.47	0.05	1.44	1.72	0.84	2.38	2.97	1.61	4.66
12000	1.62	1.76	0.05	1.78	2.26	0.91	3.03	3.98	1.75
16000	0.55	2.1	2.17	0.07	2.42	2.65	1.22	4.16	4.57
22050	2.49	0.76	2.73	2.95	0.1	2.88	3.45	1.68	4.75
24000	2.75	2.86	0.83	3.23	3.52	0.1	3.56	4.53	1.83

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Table 10. MIPS for mono at very high quality*...continued*

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
32000	1	3.56	3.63	1.11	4.2	4.34	0.14	4.84	5.3
44100	4.86	1.38	4.74	4.98	1.53	5.46	5.89	0.19	5.75
48000	5.38	5.55	1.5	5.5	5.71	1.66	6.47	7.05	0.21

Table 11. MIPS for stereo at very high quality

Fs_{IN} / Fs_{OUT}	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	0.06	2.11	2.29	1.2	3.55	3.86	2.31	6.99	7.61
11025	2.62	0.09	2.52	3.01	1.66	4.07	5.07	3.19	8
12000	2.85	3.15	0.09	3.11	3.9	1.81	5.17	6.75	3.47
16000	1.09	3.73	3.85	0.13	4.22	4.57	2.41	7.1	7.72
22050	4.32	1.5	4.79	5.23	0.17	5.05	6.02	3.32	8.15
24000	4.74	4.99	1.64	5.69	6.3	0.19	6.22	7.8	3.61
32000	1.98	6.18	6.3	2.18	7.45	7.71	0.25	8.44	9.14
44100	8.43	2.72	8.18	8.64	3.01	9.59	10.47	0.35	10.1
48000	9.26	9.66	2.97	9.49	9.97	3.27	11.39	12.59	0.38

1.3.2.3 On Cortex-A8 for worst case of 48000 Hz to 44100 Hz

Table 12. Worst case MIPs for Cortex-A8

Mode	MIPs
Mono at High Quality	3.13
Stereo at High Quality	3.61
Mono at Very High Quality	4.13
Stereo at Very High Quality	6.52

2 Application programmers interface (API)

This section describes the application programming interface (API) libraries of the SSRC module.

2.1 Type definitions

This section describes the type definitions of the SSRC module.

2.1.1 Types for allocation of instance and scratch memory

The instance memory is the memory that contains the state of one instance of the SSRC module. Multiple instances of the SSRC module can exist, each with it is own instance memory. S memory is the memory that is only used temporarily by the process function of the SSRC module. This memory can be used as scratch memory by any other function running in the same thread as the SSRC module. Different threads cannot share the scratch memories.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

The application must allocate both the instance and the scratch memory. The SSRC module does not allocate memory.

There is a data type available for both the instance and the scratch memory, namely `SSRC_Instance_t` and `SSRC_Scratch_t`. The instance type is defined as structures of the correct size in the SSRC header file. Both the instance and the scratch memory must be 4 bytes aligned.

2.1.2 LVM_Fs_en

Definition:

```
typedef enum
{
    LVM_FS_8000    = 0,
    LVM_FS_11025   = 1,
    LVM_FS_12000   = 2,
    LVM_FS_16000   = 3,
    LVM_FS_22050   = 4,
    LVM_FS_24000   = 5,
    LVM_FS_32000   = 6,
    LVM_FS_44100   = 7,
    LVM_FS_48000   = 8
} LVM_Fs_en;
```

Description:

Used to pass the input and the output sample rate to the SSRC.

2.1.3 LVM_Format_en

Definition:

```
typedef enum
{
    LVM_STEREO          = 0,
    LVM_MONOINSTEREO    = 1,
    LVM_MONO            = 2
} LVM_Format_en;
```

Description:

The `LVM_Format_en` enumerated type is used to set the value of the SSRC data format.

The SSRC supports input data in two formats Mono and Stereo. For an input buffer of `NumSamples = N` (meaning `N` sample pairs for Stereo and MonoInStereo or N samples for Mono), the format of data in the buffer is as listed in Table 13:

Table 13. SSRC data format

Sample Number	Stereo	MonoInStereo	Mono
0	Left(0)	Mono(0)	Mono(0)
1	Right(0)	Mono(0)	Mono(1)
2	Left(1)	Mono(1)	Mono(2)
3	Right(1)	Mono(1)	Mono(3)
4	Left(2)	Mono(2)	Mono(4)
“	“	“	“

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Table 13. SSRC data format *...continued*

Sample Number	Stereo	MonoInStereo	Mono
“	“	“	“
N-2	Left(N/2-1)	Mono(N/2-1)	Mono(N-2)
N-1	Right(N/2-1)	Mono(N/2-1)	Mono(N-1)
N	Left(N/2)	Mono(N/2)	Not Used
N+1	Right(N/2)	Mono(N/2)	Not Used
N+2	Left(N/2+1)	Mono(N/2+1)	Not Used
N+3	Right(N/2+1)	Mono(N/2+1)	Not Used
“	“	“	Not Used
“	“	“	Not Used
2*N-2	Left(N-1)	Mono(N-1)	Not Used

2.1.4 SSRC_Quality_en

Definition:

```
typedef enum
{
    SSRC_QUALITY_HIGH           = 0,
    SSRC_QUALITY_VERY_HIGH      = 1,
    SSRC_QUALITY_DUMMY          = LVM_MAXENUM
} SSRC_Quality_en;
```

Description:

Used to select the quality level of the SSRC. For details, see Section 1.2. Selecting the highest-quality level, comes with a cost in the SSRC processing requirements. Therefore, it should only be done for critical applications.

2.1.5 Instance parameters

Definition:

```
typedef struct
{
    SSRC_Quality_en     Quality;
    LVM_Fs_en           SSRC_Fs_In;
    LVM_Fs_en           SSRC_Fs_Out;
    LVM_Format_en       SSRC_NrOfChannels;
    short               NrSamplesIn;
    short               NrSamplesOut;
} SSRC_Params_t;
```

Description:

Used to pass the SSRC instance parameters to the SSRC module. It is a structure that contains the members for input sample rate, output sample rate, the number of channels, and the number of samples on the input and output audio stream.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

2.1.6 Nr of samples mode

Definition:

```
typedef enum
{
    SSRC_NR_SAMPLES_DEFAULT     = 0,
    SSRC_NR_SAMPLES_MIN         = 1,
    SSRC_NR_SAMPLES_DUMMY       = LVM_MAXENUM
} SSRC_NR_SAMPLES_MODE_en;
```

Description:

The `SSRC_NR_SAMPLES_MODE_en` enumerated type specifies the two different modes that can be used to retrieve the number of samples using the `SSRC_GetNrSamples` function.

2.1.7 Function return status

Definition:

```
typedef enum
{
    SSRC_OK                     = 0,
    SSRC_INVALID_FS             = 1,
    SSRC_INVALID_NR_CHANNELS    = 2,
    SSRC_NULL_POINTER           = 3,
    SSRC_WRONG_NR_SAMPLES       = 4,
    SSRC_ALLINGMENT_ERROR       = 5,
    SSRC_INVALID_MODE           = 6,
    SSRC_INVALID_VALUE          = 7,
    SSRC_ALLINGMENT_ERROR       = 8,
    LVXXX_RETURNSTATUS_DUMMY = LVM_MAXENUM
} SSRC_ReturnStatus_en;
```

Description:

The `SSRC_ReturnStatus_en` enumerated type specifies the different error codes returned by the API functions. For the exact meaning, see the individual function descriptions.

2.2 Functions

This section lists all the API functions of the SSRC module and explains their parameters.

2.2.1 SSRC_GetNrSamples

Prototype:

```
SSRC_ReturnStatus_en SSRC_GetNrSamples
   (SSRC_NR_SAMPLES_MODE_en  Mode,
    SSRC_Params_t*           pSSRC_Params );
```

Description:

This function retrieves the number of samples or sample pairs for stereo used as an input and as an output of the SSRC module.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Table 14. SSRC_GetNrSamples Parameters

Name	Type	Description
Mode	SSRC_NR_SAMPLES_MODE_en	There are two modes: • `SSRC_NR_SAMPLES_DEFAULT`: In this mode, the function returns the number of samples for 40 ms blocks • `SSRC_NR_SAMPLES_MIN`: the function returns the minimal number of samples supported for this conversion ratio. The SSRC_Init function accepts each integer multiple of this ratio. Formula: `blocksize (ms) = 1/gcd(Fs_In,Fs_Out)`
pSSRC_Params	SSRC_Params_t*	Pointer to the instance parameters. The application fills in the values of the input sample rate, the output sample rate, and the number of channels. Based on this input, the `SSRC_GetNrSamples` fills in the values for the number of samples for the input and the output audio stream.

Returns:

`SSRC_OK`	When the function call succeeds.
`SSRC_INVALID_FS`	When the requested input or output sampling rates are invalid.
`SSRC_INVALID_NR_CHANNELS`	When the channel format is not equal to `LVM_MONO` or `LVM_STEREO`.
`SSRC_NULL_POINTER`	When `pSSRC_Params` is a `NULL` pointer.
`SSRC_INVALID_MODE`	When mode is not a valid setting.

Note: *The SSRC_GetNrSamples function returns the values from the following tables. Instead of calling the SSRC_GetNrSamples function, use the values from these tables directly.*

Table 15. Number of samples for SSRC_NR_SAMPLES_DEFAULT

Sample rate	Nr of samples
8000	320
11025	441
12000	480
16000	640
22050	882
24000	960
32000	1280
44100	1764
48000	1920

Table 16. Number of samples for SSRC_NR_SAMPLES_MIN

In/Out	8000	11025	12000	16000	22050	24000	32000	44100	48000
8000	1 1	320 441	2 3	1 2	160 441	1 3	1 4	80 441	1 6
11025	441	1	147	441	1	147	441	1	147

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Table 16. Number of samples for SSRC_NR_SAMPLES_MIN*...continued*

In/Out	8000	11025	12000	16000	22050	24000	32000	44100	48000
	320	1	160	640	2	320	1280	4	640
12000	3 2	160 147	1 1	3 4	80 147	1 2	3 8	40 147	1 4
16000	2 1	640 441	4 3	1 1	320 441	2 3	1 2	160 441	1 3
22050	441 160	2 1	147 80	441 320	1 1	147 160	441 640	1 2	147 320
24000	3 1	320 147	2 1	3 2	160 147	1 1	3 4	80 147	1 2
32000	4 1	1280 441	8 3	2 1	640 441	4 3	1 1	320 441	2 3
44100	441 80	4 1	147 40	441 160	2 1	147 80	441 320	1 1	147 160
48000	6 1	640 147	4 1	3 1	320 147	2 1	3 2	160 147	1 1

2.2.2 SSRC_GetScratchSize

Prototype:

```
SSRC_ReturnStatus_en SSRC_GetScratchSize
   (SSRC_Params_t*    pSSRC_Params,
    LVM_INT32*        pScratchSize );
```

Description:

This function retrieves the scratch size for a given conversion ratio and for given buffer sizes at the input and at the output.

Table 17. SSRC_GetScratchSize Parameters

Name	Type	Description
`pSSRC_Params`	`SSRC_Params_t*`	Pointer to the instance parameters. All members should have a valid value.
`pScratchSize`	`LVM_INT32*`	Pointer to the scratch size. The `SSRC_GetScratchSize` function fills in the correct value (in bytes).

Returns:

`SSRC_OK`	When the function call succeeds.
`SSRC_INVALID_FS`	When the requested input or output sampling rates are invalid.
`SSRC_INVALID_NR_CHANNELS`	When the channel format is not equal to `LVM_MONO` or `LVM_STEREO`.
`SSRC_NULL_POINTER`	When `pSSRC_Params` or `pScratchSize` is a `NULL` pointer.
`SSRC_WRONG_NR_SAMPLES`	When the number of samples on the input or on the output are incorrect.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

2.2.3 SSRC_Init

Prototype:

```
SSRC_ReturnStatus_en SSRC_Init
    (SSRC_Instance_t* pSSRC_Instance,
     SSRC_Scratch_t*  pSSRC_Scratch,
     SSRC_Params_t*   pSSRC_Params,
     LVM_INT16**      ppInputInScratch,
     LVM_INT16**      ppOutputInScratch);
```

Description:

The `SSRC_Init` function initializes an instance of the SSRC module.

Table 18. SSRC_Init Parameters

Name	Type	Description
`pSSRC_Instance`	`SSRC_Instance_t*`	Pointer to the instance of the SSRC. This application must allocate the memory before calling the `SSRC_Init` function.
`pSSRC_Scratch`	`SSRC_Scratch_t*`	Pointer to the scratch memory. The pointer is saved inside the instance and is used by the `SSRC_Process` function. The application must allocate the scratch memory before calling the `SSRC_Init` function.
`pSSRC_Params`	`SSRC_Params_t*`	Pointer to the instance parameters.
`ppInputInScratch`	`LVM_INT16**`	The SSRC module can be called with the input samples located in scratch. This pointer points to a location that holds the pointer to the location in the scratch memory that can be used to store the input samples. For example, to save memory.
`ppOutputInScratch`	`LVM_INT16**`	The SSRC module can store the output samples in the scratch memory. This pointer points to a location that holds the pointer to the location in the scratch memory that can be used to store the output samples. For example, to save memory.

Returns:

`SSRC_OK`	When the function call succeeds.
`SSRC_INVALID_FS`	When the requested input or output sampling rates are invalid.
`SSRC_INVALID_NR_CHANNELS`	When the channel format is not equal to `LVM_MONO` or `LVM_STEREO`.
`SSRC_NULL_POINTER`	When `pSSRC_Params` or `pScratchSize` is a `NULL` pointer.
`SSRC_WRONG_NR_SAMPLES`	When the number of samples on the input or on the output are incorrect.
`SSRC_ALIGNMENT_ERROR`	When the instance memory or the scratch memory is not 4 bytes aligned.

2.2.4 SSRC_SetGains

Prototype:

```
SSRC_ReturnStatus_en SSRC_SetGains
    (SSRC_Instance_t* pSSRC_Instance,
     LVM_Mode_en      bHeadroomGainEnabled,
     LVM_Mode_en      bOutputGainEnabled,
     LVM_INT16       OutputGain);
```


All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Description:

This function sets headroom gain and the post gain of the SSRC. The `SSRC_SetGains` function is an optional function that should be used only in rare cases. Preferably, use the default settings.

Table 19. SSRC_Process Parameters

Name	Type	Description
`pSSRC_Instance`	`SSRC_Instance_t*`	Pointer to the instance of the SSRC.
`bHeadroomGainEnabled`	`LVM_Mode_en`	Parameter to enable or disable the headroom gain of the SSRC. The default value is `LVM_MODE_ON`. `LVM_MODE_OFF` can be used if it can be guaranteed that the input level is below - 6 in all cases (the default headroom is -6 dB).
`bOutputGainEnabled`	`LVM_Mode_en`	Parameter to enable or disable the output gain. The default value is `LVM_MODE_ON`.
`OutputGain`	`LVM_INT16`	The value of the output gain. The output gain is a linear gain value. 0x7FFF is equal to +6 dB and 0x0000 corresponds to -inf dB. By default, a 3 dB gain is applied (OutputGain = 23197), resulting in an overall gain of -3dB (-6dB headroom +3dB output gain).

Unit	Q format	Data Range	Default value
Linear gain	Q1.14	[0;32767]	23197

Returns:

`SSRC_OK`	When the function call succeeds
`SSRC_NULL_POINTER`	When `pSSRC_Instance` is a NULL pointer
`SSRC_INVALID_MODE`	Wrong value used for the `bHeadroomGainEnabled` or the `OutputGainEnabled` parameters.
`SSRC_INVALID_VALUE`	When `OutputGain` is out of the range [0;32767].

2.2.5 SSRC_Process

Prototype:

```
SSRC_ReturnStatus_en SSRC_Process
   (SSRC_Instance_t* pSSRC_Instance,
    LVM_INT16*       pSSRC_AudioIn,
    LVM_INT16*       pSSRC_AudioOut);
```

Description:

Process function for the SSRC module. The function takes pointers as input and output audio buffers. The sample format used for the input and output buffers is 16-bit little-endian. Stereo buffers are interleaved (L1, R1, L2, R2, and so on), mono buffers are deinterleaved (L1, L2, and so on).

Table 20. SSRC_Process Parameters

Name	Type	Description
`pSSRC_Instance`	`SSRC_Instance_t*`	Pointer to the instance of the SSRC.
`pSSRC_AudioIn`	`LVM_INT16*`	Pointer to the input samples.
`pSSRC_AudioOut`	`LVM_INT16*`	Pointer to the output samples.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Returns:

SSRC_OK	When the function call succeeds.
SSRC_NULL_POINTER	When one of pSSRC_Instance, pSSRC_AudioIn, or pSSRC_AudioOut is NULL.

2.2.6 SSRC_Process_D32

Prototype:

```
SSRC_ReturnStatus_en SSRC_Process_D32
   (SSRC_Instance_t* pSSRC_Instance,
    LVM_INT32*       pSSRC_AudioIn,
    LVM_INT32*       pSSRC_AudioOut);
```

Description:

Process function for the SSRC module. The function takes pointers as input and output audio buffers. The sample format used for the input and output buffers is 32-bit little-endian. Stereo buffers are interleaved (L1, R1, L2, R2, and so on), mono buffers are deinterleaved (L1, L2, and so on).

Table 21. SSRC_Process Parameters

Name	Type	Description
pSSRC_Instance	SSRC_Instance_t*	Pointer to the instance of the SSRC.
pSSRC_AudioIn	LVM_INT32*	Pointer to the input samples.
pSSRC_AudioOut	LVM_INT32*	Pointer to the output samples.

Returns:

SSRC_OK	When the function call succeeds.
SSRC_NULL_POINTER	When one of pSSRC_Instance, pSSRC_AudioIn, or pSSRC_AudioOut is NULL.

2.3 Dynamic function usage

This chapter explains how and when the SSRC functions are or can be used.

2.3.1 Define the number of samples to be used on input and output

Call the function SSRC_GetNrSamples. Each integer multiple of the returned number of samples can be used.

2.3.2 Allocate scratch memory

To calculate the required size of the scratch memory, call the SSRC_GetScratchSize function. Allocate memory for the returned size.

2.3.3 Initialize the SSRC instance

Call the SSRC_Init function.

2.3.4 Process samples

The SSRC_Process function can now be called any number of times.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

2.3.5 Destroy the SSRC instance

When the processing is completed, the allocated memory for the instance and the scratch can be freed.

2.4 Reentrancy

None of the SSRC functions are re-entrant.

3 Additional user information

This section provides information on the Attenuation of the signal and Notes on integration.

3.1 Attenuation of the signal

When a fully saturated or clipped input is applied to an SRC module, the aliases after the sample rate conversion, although sufficiently suppressed, can still result in a clipped output. To prevent clipped output, the output of the SSRC module is by default attenuated with 3 dB. Although not advised, this gain value can be changed using the SSRC_SetGains function.

3.2 Notes on integration

Although the sample rate converter module works with audio signals on different sampling rates, it is a synchronous module. The module takes a block of input samples, consumes the input completely, and produces a full buffer with output samples. As a result, the SSRC only accepts a limited number of input and output block sizes. To flush last, incomplete, block of an audio stream, the block is padded with zeros until it is full before the SSRC processes it.

4 Example application

The source code of the example application can be found in the `.\EX_APP\APP_FileIO\SRC` directory of the release package. The `.\EX_APP\APP_FileIO\MAKE` directory contains a make file that can be used to build the example application. When building the application, an executable is generated in the `.\EX_APP\APP_FileIO\EXE` directory.

The example application takes as command-line input parameters:

1. The path toward the input PCM file. It assumes raw 16 bit signed little-endianput. Stereo input samples should be interleaved (L1, L2 R1, R2,…), mono samples should be deinterleaved (L1,L2, and so on).
2. The path toward the output PCM file.
3. The input sample rate.
4. The output sample rate.
5. The channel format (mono or stereo).

5 Integration test

A correct integration of the SSRC module can be verified in two ways.

- Bit accurate test
- THD+N measurement

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

5.1 Bit accurate test

The TestFiles directory of the release package contains a test input (sampled at 44,100 Hz) and several expected output files (sample rates from 8000 Hz to 48,000 Hz). If the same test input file is applied to the SRC after integration in the target platform, the output is bit accurate with the expected output file that matches the output-sample rate

5.2 THD+N measurement

Produce a swept sine and feed it through the SSRC module. Do a THD+N measurement on the obtained output signal. The THD+N of the converted signals should be below - 77 in the interval [0 - 0.45] Fs_{LOW}.

6 Revision history

Table 22 summarizes revisions to this document.

Table 22. Revision history

Revision number	Date	Substantive changes
2.14.0	27 July 2023	Initial public release

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

7 Legal information

7.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

7.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile — are trademarks and/or registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.

All information provided in this document is subject to legal disclaimers.

© 2023 NXP B.V. All rights reserved.

Contents

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2023 NXP B.V. **All rights reserved.**

For more information, please visit: http://www.nxp.com

Date of release: 27 July 2023
Document identifier: SSRCLIBUG