RX Family RXv2 CPU Products

RX DSP Library Version 5.0 Additional Information

R01AN4361EJ0100
Rev.1.00
Jan 21, 2019

Introduction

This application note describes the build condition, internal functions, resource requirements and execution cycle counts of RX DSP Library Version 5.0.

Refer to the following document for the API specification of RX DSP Library Version 5.0.

- RX DSP Library APIs Version 5.0 User's Manual: Software (R01UW0200EJ0100)

Target Device

RX Family RXv2 CPU products

Contents

1. RX DSP Library

This application note describes the build condition, internal functions, resource requirements and execution cycle counts of RX DSP Library Version 5.0.

1.1 Composition of DSP Library

The DSP Library provides eight types of files based on FPU support, endianness, and the presence or absence of error checking. The library filenames corresponding to the respective conditions are shown in Table 1.1.

Table 1.1 DSP Library File Name List

FPU	Endianness	Error Checking	Library File Name
Not supported	Little-endian	None	RX_DSP_NOFPU_LE.lib
		Available	RX_DSP_NOFPU_LE_Check.lib
	Big-endian	None	RX_DSP_NOFPU_BE.lib
		Available	RX_DSP_NOFPU_BE_Check.lib
Supported	Little-endian	None	RX_DSP_FPU_LE.lib
		Available	RX_DSP_FPU_LE_Check.lib
	Big-endian	None	RX_DSP_FPU_BE.lib
		Available	RX_DSP_FPU_BE_Check.lib

1.2 RX DSP Library Build Condition

1.2.1 Toolchain

Building and testing of the DSP library are performed in the following environment.

- Renesas RX Standard Toolchain V2.01.00

1.2.2 Build condition

The build conditions for the respective DSP library files are shown below.

(1) RX_DSP_NOFPU_LE

FPU Not Supported, Little Endian, Without Error Checking

C/C++ Compiler options	Assembly options	Linker options
-isa=rxv2	-isa=rxv2	-noprelink
-define=R_DSP_PARAMETER_CHECK=0	-define=R_DSP_OVERFLOW_CHECK=0	-form=library=u
-optimize=max	-nofpu	-nomessage
-speed	-chkfpu	-nologo
-nofpu	-nologo	-output=”RX_DSP_NOFPU_LE.lib"
-nologo		

(2) RX_DSP_NOFPU_LE_Check

FPU Not Supported, Little Endian, With Error Checking

C/C++ Compiler options	Assembly options	Linker options
-isa=rxv2	-isa=rxv2	-noprelink
-define=R_DSP_PARAMETER_CHECK=1	-define=R_DSP_OVERFLOW_CHECK=1	-form=library=u
-optimize=max	-nofpu	-nomessage
-speed	-nologo	-nologo
-nofpu	-chkfpu	-output=”RX_DSP_NOFPU_LE_Check.lib"
-nologo		

(3) RX_DSP_NOFPU_BE

FPU Not Supported, Big Endian, Without Error Checking

C/C++ Compiler options	Assembly options	Linker options
-isa=rxv2	- isa=rxv2	-noprelink
-define=R_DSP_PARAMETER_CHECK=0	-define=R_DSP_OVERFLOW_CHECK=0	-form=library=u
-optimize=max	-nofpu	-nomessage
-speed	-nologo	-nologo
-nofpu	-chkfpu	-output=”RX_DSP_NOFPU_BE.lib"
-endian=big	-endian=big	
-nologo		

(4) RX_DSP_NOFPU_BE_Check

FPU Not Supported, Big Endian, With Error Checking

C/C++ Compiler options	Assembly options	Linker options
-isa=rxv2	-isa=rxv2	-noprelink
-define=R_DSP_PARAMETER_CHECK=1	-define=R_DSP_OVERFLOW_CHECK=1	-form=library=u
-optimize=max	-nofpu	-nomessage
-speed	-nologo	-nologo
-nofpu	-chkfpu	-output=”RX_DSP_NOFPU_BE_Check.lib"
-endian=big	-endian=big	
-nologo		

(5) RX_DSP_FPU_LE

FPU Supported, Little Endian, Without Error Checking

C/C++ Compiler options	Assembly options	Linker options
-isa=rxv2	-isa=rxv2	-noprelink
-define=R_DSP_PARAMETER_CHECK=0	-define=R_DSP_OVERFLOW_CHECK=0	-form=library=u
-optimize=max	-fpu	-nomessage
-speed	-define=__FPU=1	-nologo
-fpu	-nologo	-output=”RX_DSP_FPU_LE.lib"
-nologo		

(6) RX_DSP_FPU_LE_Check

FPU Supported, Little Endian, With Error Checking

C/C++ Compiler options	Assembly options	Linker options
-isa=rxv2	-isa=rxv2	-noprelink
-define=R_DSP_PARAMETER_CHECK=1	-define=R_DSP_OVERFLOW_CHECK=1	-form=library=u
-optimize=max	-fpu	-nomessage
-speed	-define=__FPU=1	-nologo
-fpu	-nologo	-output=”RX_DSP_FPU_LE_Check.lib"
-nologo		

(7) RX_DSP_FPU_BE

FPU Supported, Big Endian, Without Error Checking

C/C++ Compiler options	Assembly options	Linker options
-isa=rxv2	-isa=rxv2	-noprelink
-define=R_DSP_PARAMETER_CHECK=0	-define=R_DSP_OVERFLOW_CHECK=0	-form=library=u
-optimize=max	-fpu	-nomessage
-speed	-define=__FPU=1	-nologo
-fpu	-nologo	-output=”RX_DSP_FPU_BE.lib"
-endian=big	-endian=big	
-nologo		

(8) RX_DSP_FPU_BE_Check

FPU Supported, Big Endian, With Error Checking

C/C++ Compiler options	Assembly options	Linker options
-isa=rxv2	-isa=rxv2	-noprelink
-define=R_DSP_PARAMETER_CHECK=1	-define=R_DSP_OVERFLOW_CHECK=1	-form=library=u
-optimize=max	-fpu	-nomessage
-speed	-define=__FPU=1	-nologo
-fpu	-nologo	-output="RX_DSP_FPU_BE_Check.lib"
-endian=big	-endian=big	-exit
-nologo		

2. Internal Functions

The internal functions implement algorithms of the DSP library. Some public functions call internal functions according to the specified options. The internal functions are implemented by assembly language with registers and/or stacks, and return the status to the caller program.

The internal functions which can be directly called by the user program are shown below.

Filter operation API :

- Generic FIR filter
- IIR Biquad filter
- Single-Pole IIR filter

Matrix operation API :

- Matrix multiplication
- Matrix real number multiplication

Linear Transform API :

- Complex FFT
- Complex IFFT
- Real FFT
- Complex conjugate symmetric IFFT

2.1 Internal Function of Filter Operation API

This section describes the internal functions of the following filter operation API.

- Generic FIR filter
- IIR Biquad filter
- Single-Pole IIR filter

2.1.1 Generic FIR Filter

Format

```
r_dsp_status_t R_DSP_FIR_<intype><outtype>_asm_<option>(
     const r_dsp_firfilter_t * handle,
     const vector_t * input,
     vector_t * output
)
```

Arguments

handle	Pointer to an r_dsp_firfilter_t data structure. All members other than options of the structure are referred to. For details, see the User's Manual. The input data is stored in an array to which the member "state" has been set.
input	Pointer to the vector_t to input to the filter. The following member is referred to.
input->n	Input data count.
output	Pointer to the vector_t that stores the filter output. The following members are referred to.
output->n	Number of elements in the array to which the data member points. Must be greater than or equal to the input data count.
output->data	Pointer to the beginning of the array that stores the filter output.

Return Values

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (in case of fixed-point functions of the library with "_Check") .

Public Functions list

public function	<option>			internal function
	SATURATE	ROUNDING	Scaling	
R_DSP_FIR_i16i16	NO SATURAT	TRUNC	-	R_DSP_FIR_i16i16_asm_nt(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST		R_DSP_FIR_i16i16_asm_n2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC		R_DSP_FIR_i16i16_asm_st(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST		R_DSP_FIR_i16i16_asm_s2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)

public function	<option>			internal function
	SATURATE	ROUNDING	Scaling	
R_DSP_FIR_ci16ci16	NO SATURATE	TRUNC	No Scaling (scale=15)	R_DSP_FIR_ci16ci16_asm_ntn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<15)	R_DSP_FIR_ci16ci16_asm_ntu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>15)	R_DSP_FIR_ci16ci16_asm_ntd(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=15)	R_DSP_FIR_ci16ci16_asm_n2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<15)	R_DSP_FIR_ci16ci16_asm_n2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>15)	R_DSP_FIR_ci16ci16_asm_n2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	No Scaling (scale=15)	R_DSP_FIR_ci16ci16_asm_stn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<15)	R_DSP_FIR_ci16ci16_asm_stu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>15)	R_DSP_FIR_ci16ci16_asm_std(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=15)	R_DSP_FIR_ci16ci16_asm_s2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<15)	R_DSP_FIR_ci16ci16_asm_s2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>15)	R_DSP_FIR_ci16ci16_asm_s2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	No Scaling (scale=0)	R_DSP_FIR_i16i32_asm_ntn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<0)	R_DSP_FIR_i16i32_asm_ntu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>0)	R_DSP_FIR_i16i32_asm_ntd(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=0)	R_DSP_FIR_i16i32_asm_n2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<0)	R_DSP_FIR_i16i32_asm_n2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>0)	R_DSP_FIR_i16i32_asm_n2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	No Scaling (scale=0)	R_DSP_FIR_i16i32_asm_stn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<0)	R_DSP_FIR_i16i32_asm_stu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>0)	R_DSP_FIR_i16i32_asm_std(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=0)	R_DSP_FIR_i16i32_asm_s2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<0)	R_DSP_FIR_i16i32_asm_s2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>0)	R_DSP_FIR_i16i32_asm_s2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)

public function	<option>			internal function
	SATURATE	ROUNDING	Scaling	
R_DSP_FIR_ci16ci32	NO SATURATE	TRUNC	No Scaling (scale=0)	R_DSP_FIR_ci16ci32_asm_ntn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<0)	R_DSP_FIR_ci16ci32_asm_ntu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>0)	R_DSP_FIR_ci16ci32_asm_ntd(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=0)	R_DSP_FIR_ci16ci32_asm_n2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<0)	R_DSP_FIR_ci16ci32_asm_n2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>0)	R_DSP_FIR_ci16ci32_asm_n2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	No Scaling (scale=0)	R_DSP_FIR_ci16ci32_asm_stn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<0)	R_DSP_FIR_ci16ci32_asm_stu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>0)	R_DSP_FIR_ci16ci32_asm_std(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=0)	R_DSP_FIR_ci16ci32_asm_s2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<0)	R_DSP_FIR_ci16ci32_asm_s2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>0)	R_DSP_FIR_ci16ci32_asm_s2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	No Scaling (scale=31)	R_DSP_FIR_i32i32_asm_ntn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<31)	R_DSP_FIR_i32i32_asm_ntu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>31)	R_DSP_FIR_i32i32_asm_ntd(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=31)	R_DSP_FIR_i32i32_asm_n2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<31)	R_DSP_FIR_i32i32_asm_n2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>31)	R_DSP_FIR_i32i32_asm_n2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	No Scaling (scale=31)	R_DSP_FIR_i32i32_asm_stn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<31)	R_DSP_FIR_i32i32_asm_stu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>31)	R_DSP_FIR_i32i32_asm_std(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=31)	R_DSP_FIR_i32i32_asm_s2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<31)	R_DSP_FIR_i32i32_asm_s2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>31)	R_DSP_FIR_i32i32_asm_s2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)

public function	<option>			internal function
	SATURATE	ROUNDING	Scaling	
R_DSP_FIR_ci32ci32	NO SATURATE	TRUNC	No Scaling (scale=31)	R_DSP_FIR_ci32ci32_asm_ntn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<31)	R_DSP_FIR_ci32ci32_asm_ntu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>31)	R_DSP_FIR_ci32ci32_asm_ntd(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=31)	R_DSP_FIR_ci32ci32_asm_n2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<31)	R_DSP_FIR_ci32ci32_asm_n2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>31)	R_DSP_FIR_ci32ci32_asm_n2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	No Scaling (scale=31)	R_DSP_FIR_ci32ci32_asm_stn(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<31)	R_DSP_FIR_ci32ci32_asm_stu(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>31)	R_DSP_FIR_ci32ci32_asm_std(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	No Scaling (scale=31)	R_DSP_FIR_ci32ci32_asm_s2n(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Up (scale<31)	R_DSP_FIR_ci32ci32_asm_s2u(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
			Scaling Down (scale>31)	R_DSP_FIR_ci32ci32_asm_s2d(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
R_DSP_FIR_f32f32	-	-	-	R_DSP_FIR_f32f32_asm(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
R_DSP_FIR_cf32cf32	-	-	-	R_DSP_FIR_cf32cf32_asm(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)

2.1.2 IIR Biquad Filter

Format

```
r_dsp_status_t R_DSP_IIRBiquad_<intype><outtype>_asm_<option>(
     const r_dsp_iirbiquad_t * handle,
     const vector_t * input,
     vector_t * output
)
```

Arguments

handle	Pointer to an r_dsp_iirbiquad_t data structure. All members other than options of the structure are referred to. For details, see the User's Manual.
input	Pointer to the vector_t to input to the filter. The following members are referred to.
input->n	Input data count.
input->data	Pointer to the beginning of an array that stores the input data.
output	Pointer to the vector_t that stores the filter output. The following members are referred to.
output->n	Number of elements in the array to which the data member points. Must be greater than or equal to the input data count.
output->data	Pointer to the beginning of the array that stores the filter output.

Return Values

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (in case of fixed-point operation of library with __Check").

Public Functions list

public function	<option>				internal function
	SATURATE	ROUNDING	qint	Scaling	
R_DSP_IIRBiquad_i16i16	NO SATURATE	TRUNC	0	No Scaling (scale=15)	R_DSP_IIRBiquad_i16i16_asm_nt0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<15)	R_DSP_IIRBiquad_i16i16_asm_nt0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>15)	R_DSP_IIRBiquad_i16i16_asm_nt0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=14)	R_DSP_IIRBiquad_i16i16_asm_nt1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<14)	R_DSP_IIRBiquad_i16i16_asm_nt1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>14)	R_DSP_IIRBiquad_i16i16_asm_nt1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=15)	R_DSP_IIRBiquad_i16i16_asm_n20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<15)	R_DSP_IIRBiquad_i16i16_asm_n20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>15)	R_DSP_IIRBiquad_i16i16_asm_n20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=14)	R_DSP_IIRBiquad_i16i16_asm_n21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<14)	R_DSP_IIRBiquad_i16i16_asm_n21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>14)	R_DSP_IIRBiquad_i16i16_asm_n21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	0	No Scaling (scale=15)	R_DSP_IIRBiquad_i16i16_asm_st0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<15)	R_DSP_IIRBiquad_i16i16_asm_st0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>15)	R_DSP_IIRBiquad_i16i16_asm_st0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=14)	R_DSP_IIRBiquad_i16i16_asm_st1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<14)	R_DSP_IIRBiquad_i16i16_asm_st1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>14)	R_DSP_IIRBiquad_i16i16_asm_st1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=15)	R_DSP_IIRBiquad_i16i16_asm_s20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<15)	R_DSP_IIRBiquad_i16i16_asm_s20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>15)	R_DSP_IIRBiquad_i16i16_asm_s20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=14)	R_DSP_IIRBiquad_i16i16_asm_s21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<14)	R_DSP_IIRBiquad_i16i16_asm_s21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>14)	R_DSP_IIRBiquad_i16i16_asm_s21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)

public function	<option>				internal function
	SATURATE	ROUNDING	qint	Scaling	
R_DSP_IIRBiquad_ci16ci16	NO SATURATE	TRUNC	0	No Scaling (scale=15)	R_DSP_IIRBiquad_ci16ci16_asm_nt0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<15)	R_DSP_IIRBiquad_ci16ci16_asm_nt0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>15)	R_DSP_IIRBiquad_ci16ci16_asm_nt0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=14)	R_DSP_IIRBiquad_ci16ci16_asm_nt1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<14)	R_DSP_IIRBiquad_ci16ci16_asm_nt1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>14)	R_DSP_IIRBiquad_ci16ci16_asm_nt1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=15)	R_DSP_IIRBiquad_ci16ci16_asm_n20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<15)	R_DSP_IIRBiquad_ci16ci16_asm_n20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>15)	R_DSP_IIRBiquad_ci16ci16_asm_n20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=14)	R_DSP_IIRBiquad_ci16ci16_asm_n21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<14)	R_DSP_IIRBiquad_ci16ci16_asm_n21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>14)	R_DSP_IIRBiquad_ci16ci16_asm_n21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	0	No Scaling (scale=15)	R_DSP_IIRBiquad_ci16ci16_asm_st0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<15)	R_DSP_IIRBiquad_ci16ci16_asm_st0u const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>15)	R_DSP_IIRBiquad_ci16ci16_asm_st0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=14)	R_DSP_IIRBiquad_ci16ci16_asm_st1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<14)	R_DSP_IIRBiquad_ci16ci16_asm_st1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>14)	R_DSP_IIRBiquad_ci16ci16_asm_st1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=15)	R_DSP_IIRBiquad_ci16ci16_asm_s20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<15)	R_DSP_IIRBiquad_ci16ci16_asm_s20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>15)	R_DSP_IIRBiquad_ci16ci16_asm_s20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=14)	R_DSP_IIRBiquad_ci16ci16_asm_s21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<14)	R_DSP_IIRBiquad_ci16ci16_asm_s21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>14)	R_DSP_IIRBiquad_ci16ci16_asm_s21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)

public function	<option>				internal function
	SATURATE	ROUNDING	qint	Scaling	
R_DSP_IIRBiquad_i16i32ci16	NO SATURTAE	TRUNC	0	No Scaling (scale=0)	R_DSP_IIRBiquad_i16i32_asm_nt0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_i16i32_asm_nt0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_i16i32_asm_nt0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=0)	R_DSP_IIRBiquad_i16i32_asm_nt1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_i16i32_asm_nt1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_i16i32_asm_nt1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=0)	R_DSP_IIRBiquad_i16i32_asm_n20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_i16i32_asm_n20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_i16i32_asm_n20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=0)	R_DSP_IIRBiquad_i16i32_asm_n21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_i16i32_asm_n21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_i16i32_asm_n21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	0	No Scaling (scale=0)	R_DSP_IIRBiquad_i16i32_asm_st0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_i16i32_asm_st0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_i16i32_asm_st0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=0)	R_DSP_IIRBiquad_i16i32_asm_st1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_i16i32_asm_st1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_i16i32_asm_st1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=0)	R_DSP_IIRBiquad_i16i32_asm_s20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_i16i32_asm_s20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_i16i32_asm_s20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=0)	R_DSP_IIRBiquad_i16i32_asm_s21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_i16i32_asm_s21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_i16i32_asm_s21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)

public function	<option>				internal function
	SATURATE	ROUNDING	qint	Scaling	
R_DSP_IIRBiquad_ci16ci32	NO SATURATE	TRUNC	0	No Scaling (scale=0)	R_DSP_IIRBiquad_ci16ci32_asm_nt0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_ci16ci32_asm_nt0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_ci16ci32_asm_nt0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=0)	R_DSP_IIRBiquad_ci16ci32_asm_nt1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_ci16ci32_asm_nt1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_ci16ci32_asm_nt1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=0)	R_DSP_IIRBiquad_ci16ci32_asm_n20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_ci16ci32_asm_n20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_ci16ci32_asm_n20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=0)	R_DSP_IIRBiquad_ci16ci32_asm_n21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_ci16ci32_asm_n21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_ci16ci32_asm_n21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	0	No Scaling (scale=0)	R_DSP_IIRBiquad_ci16ci32_asm_st0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_ci16ci32_asm_st0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_ci16ci32_asm_st0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=0)	R_DSP_IIRBiquad_ci16ci32_asm_st1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_ci16ci32_asm_st1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_ci16ci32_asm_st1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=0)	R_DSP_IIRBiquad_ci16ci32_asm_s20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_ci16ci32_asm_s20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_ci16ci32_asm_s20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=0)	R_DSP_IIRBiquad_ci16ci32_asm_s21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<0)	R_DSP_IIRBiquad_ci16ci32_asm_s21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>0)	R_DSP_IIRBiquad_ci16ci32_asm_s21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)

public function	<option>				internal function
	SATURATE	ROUNDING	qint	Scaling	
R_DSP_IIRBiquad_i32i32	NOSATURATE	TRUNC	0	No Scaling (scale=31)	R_DSP_IIRBiquad_i32i32_asm_nt0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<31)	R_DSP_IIRBiquad_i32i32_asm_nt0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>31)	R_DSP_IIRBiquad_i32i32_asm_nt0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=30)	R_DSP_IIRBiquad_i32i32_asm_nt1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<30)	R_DSP_IIRBiquad_i32i32_asm_nt1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>30)	R_DSP_IIRBiquad_i32i32_asm_nt1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=31)	R_DSP_IIRBiquad_i32i32_asm_n20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<31)	R_DSP_IIRBiquad_i32i32_asm_n20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>31)	R_DSP_IIRBiquad_i32i32_asm_n20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=30)	R_DSP_IIRBiquad_i32i32_asm_n21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<30)	R_DSP_IIRBiquad_i32i32_asm_n21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>30)	R_DSP_IIRBiquad_i32i32_asm_n21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	0	No Scaling (scale=31)	R_DSP_IIRBiquad_i32i32_asm_st0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<31)	R_DSP_IIRBiquad_i32i32_asm_st0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>31)	R_DSP_IIRBiquad_i32i32_asm_st0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=30)	R_DSP_IIRBiquad_i32i32_asm_st1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<30)	R_DSP_IIRBiquad_i32i32_asm_st1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>30)	R_DSP_IIRBiquad_i32i32_asm_st1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=31)	R_DSP_IIRBiquad_i32i32_asm_s20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<31)	R_DSP_IIRBiquad_i32i32_asm_s20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>31)	R_DSP_IIRBiquad_i32i32_asm_s20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=30)	R_DSP_IIRBiquad_i32i32_asm_s21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<30)	R_DSP_IIRBiquad_i32i32_asm_s21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>30)	R_DSP_IIRBiquad_i32i32_asm_s21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)

public function	<option>				internal function
	SATURATE	ROUNDING	qint	Scaling	
R_DSP_IIRBiquad_ci32ci32	NO SATURATE	TRUNC	0	No Scaling (scale=31)	R_DSP_IIRBiquad_ci32ci32_asm_nt0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<31)	R_DSP_IIRBiquad_ci32ci32_asm_nt0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>31)	R_DSP_IIRBiquad_ci32ci32_asm_nt0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=30)	R_DSP_IIRBiquad_ci32ci32_asm_nt1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<30)	R_DSP_IIRBiquad_ci32ci32_asm_nt1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>30)	R_DSP_IIRBiquad_ci32ci32_asm_nt1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=31)	R_DSP_IIRBiquad_ci32ci32_asm_n20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<31)	R_DSP_IIRBiquad_ci32ci32_asm_n20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>31)	R_DSP_IIRBiquad_ci32ci32_asm_n20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=30)	R_DSP_IIRBiquad_ci32ci32_asm_n21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<30)	R_DSP_IIRBiquad_ci32ci32_asm_n21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>30)	R_DSP_IIRBiquad_ci32ci32_asm_n21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	0	No Scaling (scale=31)	R_DSP_IIRBiquad_ci32ci32_asm_st0n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<31)	R_DSP_IIRBiquad_ci32ci32_asm_st0u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>31)	R_DSP_IIRBiquad_ci32ci32_asm_st0d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=30)	R_DSP_IIRBiquad_ci32ci32_asm_st1n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<30)	R_DSP_IIRBiquad_ci32ci32_asm_st1u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>30)	R_DSP_IIRBiquad_ci32ci32_asm_st1d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	0	No Scaling (scale=31)	R_DSP_IIRBiquad_ci32ci32_asm_s20n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<31)	R_DSP_IIRBiquad_ci32ci32_asm_s20u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>31)	R_DSP_IIRBiquad_ci32ci32_asm_s20d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
			1	No Scaling (scale=30)	R_DSP_IIRBiquad_ci32ci32_asm_s21n(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Up (scale<30)	R_DSP_IIRBiquad_ci32ci32_asm_s21u(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
				Scaling Down (scale>30)	R_DSP_IIRBiquad_ci32ci32_asm_s21d(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
R_DSP_IIRBiquad_f32f32					R_DSP_IIRBiquad_f32f32_asm(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
R_DSP_IIRBiquad_cf32cf32					R_DSP_IIRBiquad_cf32cf32_asm(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)

2.1.3 Single-Pole IIR Filter

Format

```
r_dsp_status_t R_DSP_IIRSinglePole_<intype><outtype>_asm_<option>(
     const r_dsp_iirsinglepole_t * handle,
     const vector_t * input,
     vector_t * output
)
```

Arguments

handle	Pointer to an r_dsp_iirsinglepole_t data structure. All members other than options of the structure are referred to. For details, see the User's Manual.
input	Pointer to the vector_t to input to the filter. The following members are referred to.
input->n	Input data count.
input->data	Pointer to the beginning of an array that stores the input data.
output	Pointer to the vector_t that stores the filter output. The following members are referred to.
output->n	Number of elements in the array to which the data member points. Must be greater than or equal to the input data count.
output->data	Pointer to the beginning of the array that stores the filter output.

Return Values

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (in case of fixed-point operation of library with "_Check").

Public Functions list

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_IIRSinglePole_i16i16	NO SATURATE	TRUNC	R_DSP_IIRSinglePole_i16i16_asm_nt(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRSinglePole_i16i16_asm_n2(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRSinglePole_i16i16_asm_st(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRSinglePole_i16i16_asm_s2(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
R_DSP_IIRSinglePole_i16i32	NO SATURATE	TRUNC	R_DSP_IIRSinglePole_i16i32_asm_nt(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRSinglePole_i16i32_asm_n2(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRSinglePole_i16i32_asm_st(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRSinglePole_i16i32_asm_s2(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_IIRSinglePole_i32i32	NO SATURATE	TRUNC	R_DSP_IIRSinglePole_i32i32_asm_nt(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRSinglePole_i32i32_asm_n2(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRSinglePole_i32i32_asm_st(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRSinglePole_i32i32_asm_s2(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)
R_DSP_IIRSinglePole_f32f32	-	-	R_DSP_IIRSinglePole_f32f32_asm(const r_dsp_iirsinglepole_t * handle, const vector_t * input, vector_t * output)

2.2 Internal Function of Matrix Operation API

This section describes the internal functions of the following matrix operations.

- Matrix multiplication
- Matrix real number multiplication

2.2.1 Matrix Multiplication

Format

```
r_dsp_status_t R_DSP_MatrixMul_<intype><outtype>_asm_<option>(
     const matrix_t * inputA,
     const matrix_t * inputB,
     matrix_t * output,
     scale_t shift,
)
```

Arguments

inputA	Pointer to the multiplicand matrix. The matrix structure and the data to which the pointers in the structure point are not modified by these functions. The following members are referred to.
inputA->nRows	Number of rows in the matrices.
inputA->nCols	Number of columns in the matrices.
inputA->data	Pointer to the first element of a matrix.
inputB	Pointer to the multiplier matrix. The matrix structure and the data to which the pointers in the structure point are not modified by these functions. The following members are referred to.
inputB->nRows	Number of rows in the matrices.
inputB->nCols	Number of columns in the matrices.
inputB->data	Pointer to the first element of a matrix.
output	Pointer to the output matrix where operation results are stored. The following members are referred to.
output ->nRows	Number of rows in the matrices. The function updates this.
output ->nCols	Number of columns in the matrices. The function updates this.
output ->data	Pointer to the first element of a matrix.
shift	Output data scaling parameter. For details, see the User's Manual. For fixed-point operations, the operation result is right-shifted corresponding to this value. The scaling parameter is an integer, and the valid value ranges are as follows. i32i32, ci32ci32 format: 1 to 62 i16i16, ci16ci16 format: 1 to 30 i16i32, ci16ci32 format: -31 to +31 (negative values indicate left-shifting) For floating-point operations, the operation results are multiplied by this value. The scaling parameter is a floating-point value. When the value is greater than 1.0, the results are amplified. When the value is smaller than 1.0, the results are attenuated.

Return Value

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (in case of fixed-point functions of the library with "_Check").

Public Functions list

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_MatrixMul_i16i16	NO SATURATE	TRUNC	R_DSP_MatrixMul_i16i16_asm_nt(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_i16i16_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixMul_i16i16_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_i16i16_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixMul_ci16ci16	NO SATURATE	TRUNC	R_DSP_MatrixMul_ci16ci16_asm_nt(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_ci16ci16_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixMul_ci16ci16_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_ci16ci16_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixMul_i16i32	NO SATURATE	TRUNC	R_DSP_MatrixMul_i16i32_asm_nt(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_i16i32_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixMul_i16i32_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_i16i32_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixMul_ci16ci32	NO SATURATE	TRUNC	R_DSP_MatrixMul_ci16ci32_asm_nt(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_ci16ci32_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixMul_ci16ci32_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_ci16ci32_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixMul_i32i32	NO SATURATE	TRUNC	R_DSP_MatrixMul_i32i32_asm_nt(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_i32i32_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixMul_i32i32_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_i32i32_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_MatrixMul_ci32ci32	NO SATURATE	TRUNC	R_DSP_MatrixMul_ci32ci32_asm_nt(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_ci32ci32_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixMul_ci32ci32_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixMul_ci32ci32_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixMul_f32f32	-	-	R_DSP_MatrixMul_f32f32_asm_nt(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixMul_cf32cf32	-	-	R_DSP_MatrixMul_cf32cf32_asm_nt(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift, uint16_t options)

2.2.2 Matrix Real Number Multiplication

Format

```
r_dsp_status_t R_DSP_MatrixScale_<intype1><outtype>_asm_<option>(
     const matrix_t * input,
     const <intype2> scalar,
     matrix_t * output,
     scale_t shift,
)
```

Arguments

input	Pointer to the input matrix. The matrix structure and the data to which the pointers in the structure point are not modified by these functions. The following members are referred to.
input->nRows	Number of rows in the matrices.
input->nCols	Number of columns in the matrices.
input->data	Pointer to the first element of a matrix.
scalar	The value by which to multiply each element of the matrix. The same data type as one of the input data.
output	Pointer to the output matrix where operation results are stored. The following members are referred to.
output ->nRows	Number of rows in the matrices. The function updates this.
output ->nCols	Number of columns in the matrices. The function updates this.
output ->data	Pointer to the first element of a matrix.
shift	Output data scaling parameter. For details, see the User's Manual. For fixed-point operations, the operation result is right-shifted corresponding to this value. The scaling parameter is an integer, and the valid value ranges are as follows. i32i32, ci32ci32 format: 1 to 62 i16i16, ci16ci16 format: 1 to 30 i16i32, ci16ci32 format: -31 to +31 (negative values indicate left-shifting) For floating-point operations, the operation results are multiplied by this value. The scaling parameter is a floating-point value. When the value is greater than 1.0, the results are amplified. When the value is smaller than 1.0, the results are attenuated.

Return Value

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (in case of fixed-point functions of the library with "_Check".

Public Functions list

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_MatrixScale_i16i16	NO SATURATE	TRUNC	R_DSP_MatrixScale_i16i16_asm_nt(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_i16i16_asm_n2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixScale_i16i16_asm_st(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_i16i16_asm_s2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixScale_ci16ci16	NO SATURATE	TRUNC	R_DSP_MatrixScale_ci16ci16_asm_nt(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_ci16ci16_asm_n2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixScale_ci16ci16_asm_st(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_ci16ci16_asm_s2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixScale_i16i32	NO SATURATE	TRUNC	R_DSP_MatrixScale_i16i32_asm_nt(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_i16i32_asm_n2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixScale_i16i32_asm_st(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_i16i32_asm_s2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixScale_ci16ci32	NO SATURATE	TRUNC	R_DSP_MatrixScale_ci16ci32_asm_nt(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_ci16ci32_asm_n2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixScale_ci16ci32_asm_st(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_ci16ci32_asm_s2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixScale_i32i32	NO SATURATE	TRUNC	R_DSP_MatrixScale_i32i32_asm_nt(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_i32i32_asm_n2(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixScale_i32i32_asm_st(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_i32i32_asm_s2(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift, uint16_t options)

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_MatrixScale_ci32ci32	NO SATURATE	TRUNC	R_DSP_MatrixScale_ci32ci32_asm_nt(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_ci32ci32_asm_n2(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift, uint16_t options)
	SATURATE	TRUNC	R_DSP_MatrixScale_ci32ci32_asm_st(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift, uint16_t options)
		NEAREST	R_DSP_MatrixScale_ci32ci32_asm_s2(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixScale_f32f32	-	-	R_DSP_MatrixScale_f32f32_asm_nt(const matrix_t * input, const float scalar, matrix_t * output, scale_t shift, uint16_t options)
R_DSP_MatrixScale_cf32cf32	-	-	R_DSP_MatrixScale_cf32cf32_asm_nt(const matrix_t * input, const float scalar, matrix_t * output, scale_t shift, uint16_t options)

2.3 Internal Function of Linear Transform API

This section describes the internal functions of the following linear transform operations.

- Complex FFT
- Complex IFFT
- Real FFT
- Complex conjugate symmetric IFFT

2.3.1 Complex FFT Operation

Format

```
r_dsp_status_t R_cfft_<option>_<intype><outtype>(
     r_dsp_fft_t * handle,
     <intype> * src,
     <outtype> * dst
)
```

16/32/64 points floating point complex FFT functions remains for compatibility with previous versions of the DSP library. These functions permutate the data by bit reversal.

```
r_dsp_status_t R_cfft_<point>_cf32cf32(
     cplxf32_t * src,
     cplxf32_t * dst
)
```

Arguments

handle	Pointer to the FFT handle. For details, see the User's Manual. The following members are referred to.
handle->n	FFT point count.
handle->twiddles	Pointer to the twiddle factor array.
handle->bitrev	Pointer to the bit reversal table.
src	Pointer to the beginning of a complex number array where input data is stored.
dst	Pointer to the beginning of a complex number array where the calculation result is stored.

Return Value

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (in ase of fixed-point functions of the library with "_Check".

Public Functions list

public function	<option>		internal function
	TW32	SCALE	
R_DSP_FFT_ci16ci16	-	-	R_cfft_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
		SC	R_cfft_sc_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
		X2	R_cfft_x2_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
	TW32	-	R_cfft_tw32_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
		SC	R_cfft_sc_tw32_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
		X2	R_cfft_x2_tw32_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
R_DSP_FFT_ci16ci32	-	-	R_cfft_ci16ci32(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi32_t * dst)
		SC	R_cfft_sc_ci16ci32(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi32_t * dst)
		X2	R_cfft_x2_ci16ci32(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi32_t * dst)
	TW32	-	R_cfft_tw32_ci16ci32(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi32_t * dst)
		SC	R_cfft_sc_tw32_ci16ci32(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi32_t * dst)
		X2	R_cfft_x2_tw32_ci16ci32(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi32_t * dst)
R_DSP_FFT_ci32ci32	-	-	R_cfft_ci32ci32(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi32_t * dst)
		SC	R_cfft_sc_ci32ci32(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi32_t * dst)
		X2	R_cfft_x2_ci32ci32(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi32_t * dst)

public function	<option>		internal function
	TW32	SCALE	
R_DSP_FFT_cf32cf32	'	-	R_cfft_cf32cf32(r_dsp_fft_t * handle, const cplxf32_t * src, cplxf32_t * dst)
			R_cfft16_cf32cf32(const cplxf32_t * src, cplxf32_t * dst)
			R_cfft32_cf32cf32(const cplxf32_t * src, cplxf32_t * dst)
			R_cfft64_cf32cf32(const cplxf32_t * src, cplxf32_t * dst)

2.3.2 Complex IFFT Operation

Format

```
r_dsp_status_t R_icfft_<option>_<intype><outtype>(
     r_dsp_fft_t * handle,
     <intype> * src,
     <outtype> * dst
)
```

Arguments

handle	Pointer to the FFT handle. For details, see the User's Manual. The following members are referred to.
handle->n	IFFT point count.
handle->twiddles	Pointer to the twiddle factor array.
handle->bitrev	Pointer to the bit reversal table.
src	Pointer to the beginning of a complex number array where input data is stored.
dst	Pointer to the beginning of a complex number array where the calculation result is stored.

Return Value

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (when fixed-point operation of library with _Check).

Public Functions list

public function	<option>		internal function
	TW32	SCALE	
R_DSP_IFFT_ci16ci16	-	-	R_icfft_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
		SC	R_icfft_sc_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
		X2	R_icfft_x2_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
	TW32	-	R_icfft_tw32_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
		SC	R_icfft_sc_tw32_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
		X2	R_icfft_x2_tw32_ci16ci16(r_dsp_fft_t * handle, const cplxi16_t * src, cplxi16_t * dst)
R_DSP_IFFT_ci32ci16	-	-	R_icfft_ci32ci16(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi16_t * dst)
		SC	R_icfft_sc_ci32ci16(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi16_t * dst)
		X2	R_icfft_x2_ci32ci16(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi16_t * dst)
	TW32	-	R_icfft_tw32_ci32ci16(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi16_t * dst)
		SC	R_icfft_sc_tw32_ci32ci16(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi16_t * dst)
		X2	R_icfft_x2_tw32_ci32ci16(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi16_t * dst)
R_DSP_IFFT_ci32ci32	-	-	R_icfft_ci32ci32(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi32_t * dst)
		SC	R_icfft_sc_ci32ci32(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi32_t * dst)
		X2	R_icfft_x2_ci32ci32(r_dsp_fft_t * handle, const cplxi32_t * src, cplxi32_t * dst)
R_DSP_IFFT_cf32cf32	-	-	R_icfft_cf32cf32(r_dsp_fft_t * handle, const cplxf32_t * src, cplxf32_t * dst)

2.3.3 Real FFT Operation

Format

```
r_dsp_status_t R_rfft_<option>_<intype><outtype>(
     r_dsp_fft_t * handle,
     <intype> * src,
     <outtype> * dst
)
```

Arguments

handle	Pointer to the FFT handle. For details, see the User's Manual. The following members are referred to.
handle->n	FFT point count.
handle->twiddles	Pointer to the twiddle factor array.
handle->bitrev	Pointer to the bit reversal table.
handle->windows	Pointer to the coefficient array of the window function. If a window function is not used, specify null. Set the coefficients to be the same type as the input data.
src	Pointer to the beginning of a complex number array where input data is stored.
dst	Pointer to the beginning of a complex number array where the calculation result is stored. (N: FFT point count)

index	0		1		…	N/2-1	
	Real	Imag	Real	Imag	…	Real	Imag
value	R_0	$R_{N/2}$	R_1	I_1	…	$R_{N/2-1}$	$I_{N/2-1}$

Return Value

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (in case of fixed-point functions of the library with "_Check".

Public Functions list

public function	<option>		internal function
	TW32	SCALE	
R_DSP_FFT_i16ci16	-	-	R_rfft_i16ci16(r_dsp_fft_t * handle, const int16_t * src, cplxi16_t * dst)
		SC	R_rfft_sc_i16ci16(r_dsp_fft_t * handle, const int16_t * src, cplxi16_t * dst)
		X2	R_rfft_x2_i16ci16(r_dsp_fft_t * handle, const int16_t * src, cplxi16_t * dst)
	TW32	-	R_rfft_tw32_i16ci16(r_dsp_fft_t * handle, const int16_t * src, cplxi16_t * dst)
		SC	R_rfft_sc_tw32_i16ci16(r_dsp_fft_t * handle, const int16_t * src, cplxi16_t * dst)
		X2	R_rfft_x2_tw32_i16ci16(r_dsp_fft_t * handle, const int16_t * src, cplxi16_t * dst)
R_DSP_FFT_i16ci32	-	-	R_rfft_i16ci32(r_dsp_fft_t * handle, const int16_t * src, cplxi32_t * dst)
		SC	R_rfft_sc_i16ci32(r_dsp_fft_t * handle, const int16_t * src, cplxi32_t * dst)
		X2	R_rfft_x2_i16ci32(r_dsp_fft_t * handle, const int16_t * src, cplxi32_t * dst)
	TW32	-	R_rfft_tw32_i16ci32(r_dsp_fft_t * handle, const int16_t * src, cplxi32_t * dst)
		SC	R_rfft_sc_tw32_i16ci32(r_dsp_fft_t * handle, const int16_t * src, cplxi32_t * dst)
		X2	R_rfft_x2_tw32_i16ci32(r_dsp_fft_t * handle, const int16_t * src, cplxi32_t * dst)
R_DSP_FFT_i32ci32	-	-	R_rfft_i32ci32(r_dsp_fft_t * handle, const int32_t * src, cplxi32_t * dst)
		SC	R_rfft_sc_i32ci32(r_dsp_fft_t * handle, const int32_t * src, cplxi32_t * dst)
		X2	R_rfft_x2_i32ci32(r_dsp_fft_t * handle, const int32_t * src, cplxi32_t * dst)
R_DSP_FFT_f32cf32	-	-	R_rfft_f32cf32(r_dsp_fft_t * handle, const float * src, cplxf32_t * dst)

2.3.4 Complex Conjugate Symmetric IFFT Operation

Format

```
r_dsp_status_t R_irfft_<option>_<intype><outtype>(
     r_dsp_fft_t * handle,
     <intype> * src,
     <outtype> * dst
)
```

Arguments

handle	Pointer to the FFT handle. For details, see the User's Manual. The following members are referred to.
handle->n	IFFT point count.
handle->twiddles	Pointer to the twiddle factor array.
handle->bitrev	Pointer to the bit reversal table.
src	Pointer to the beginning of a complex number array where input data is stored. The storage order of the input data is as follows. (N: IFFT point count)
dst	Pointer to the beginning of a complex number array where the calculation result is stored.

index	0		1		…	N/2-1	
	Real	Imag	Real	Imag	…	Real	Imag
value	R_0	$R_{N/2}$	R_1	I_1	…	$R_{N/2-1}$	$I_{N/2-1}$

Return Value

R_DSP_STATUS_OK	Normal exit.
R_DSP_STATUS_OVERFLOW	Overflow occurrence (in case of fixed-point functions of the library with "_Check".

Public Functions list

public function	<option>		internal function
	TW32	SCALE	
R_DSP_IFFT_CCS_ci16i16	-	-	R_irfft_ci16i16(r_dsp_fft_t * handle, const cplxi16_t * src, int16_t * dst)
		SC	R_irfft_sc_ci16i16(r_dsp_fft_t * handle, const cplxi16_t * src, int16_t * dst)
		X2	R_irfft_x2_ci16i16(r_dsp_fft_t * handle, const cplxi16_t * src, int16_t * dst)
	TW32	-	R_irfft_tw32_ci16i16(r_dsp_fft_t * handle, const cplxi16_t * src, int16_t * dst)
		SC	R_irfft_sc_tw32_ci16i16(r_dsp_fft_t * handle, const cplxi16_t * src, int16_t * dst)
		X2	R_irfft_x2_tw32_ci16i16(r_dsp_fft_t * handle, const cplxi16_t * src, int16_t * dst)
R_DSP_IFFT_CCS_ci32i16	-	-	R_irfft_ci32i16(r_dsp_fft_t * handle, const cplxi32_t * src, int16_t * dst)
		SC	R_irfft_sc_ci32i16(r_dsp_fft_t * handle, const cplxi32_t * src, int16_t * dst)
		X2	R_irfft_x2_ci32i16(r_dsp_fft_t * handle, const cplxi32_t * src, int16_t * dst)
	TW32	-	R_irfft_tw32_ci32i16(r_dsp_fft_t * handle, const cplxi32_t * src, int16_t * dst)
		SC	R_irfft_sc_tw32_ci32i16(r_dsp_fft_t * handle, const cplxi32_t * src, int16_t * dst)
		X2	R_irfft_x2_tw32_ci32i16(r_dsp_fft_t * handle, const cplxi32_t * src, int16_t * dst)
R_DSP_IFFT_CCS_ci32i32	-	-	R_irfft_ci32i32(r_dsp_fft_t * handle, const cplxi32_t * src, int32_t * dst)
		SC	R_irfft_sc_ci32i32(r_dsp_fft_t * handle, const cplxi32_t * src, int32_t * dst)
		X2	R_irfft_x2_ci32i32(r_dsp_fft_t * handle, const cplxi32_t * src, int32_t * dst)
R_DSP_IFFT_CCS_cf32f32	-	-	R_irfft_cf32f32(r_dsp_fft_t * handle, const cplxf32_t * src, float * dst)

3. Resource Requirement

This section describes the ROM and stack size requirements for each function when using the library with error checking and no error checking.

3.1 Statistics Operation API

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
Mean value	i16	int16_t	R_DSP_Mean_i16	622	40	650	40
	i32	int32_t	R_DSP_Mean_i32	626	40	654	40
	f32	float	R_DSP_Mean_f32	361	20	391	20
Minimum value	i16	int16_t	R_DSP_Min_i16	387	16	417	16
	i32	int32_t	R_DSP_Min_i32	371	16	401	16
	f32	float	R_DSP_Min_f32	571	16	601	16
Maximum value	i16	int16_t	R_DSP_Max_i16	387	16	417	16
	i32	int32_t	R_DSP_Max_i32	371	16	401	16
	f32	float	R_DSP_Max_f32	571	16	601	16
Minimum value with index	i16	int16_t	R_DSP_ArgMin_i16	708	48	742	48
	i32	int32_t	R_DSP_ArgMin_i32	590	44	624	44
	f32	float	R_DSP_ArgMin_f32	625	44	659	44
Maximum value with index	i16	int16_t	R_DSP_ArgMax_i16	708	48	742	48
	i32	int32_t	R_DSP_ArgMax_i32	590	44	624	44
	f32	float	R_DSP_ArgMax_f32	625	44	659	44
Mean absolute value and maximum absolute value	i16	int16_t	R_DSP_MeanAbs_i16	429	20	459	20
	i32	int32_t	R_DSP_MeanAbs_i32	619	32	648	32
	f32	float	R_DSP_MeanAbs_f32	735	168	765	168
	i16	int16_t	R_DSP_MaxAbs_i16	455	16	491	16
	i32	int32_t	R_DSP_MaxAbs_i32	617	16	654	16
	f32	float	R_DSP_MaxAbs_f32	948	164	978	164
	i16	int16_t	R_DSP_MeanMaxAbs_i16	558	28	598	28
	i32	int32_t	R_DSP_MeanMaxAbs_i32	1137	168	1175	168
	f32	float	R_DSP_MeanMaxAbs_f32	1211	176	1244	176
Mean value and variance	i16	int16_t	R_DSP_MeanVar_i16	34	60	78	60
	i32	int32_t	R_DSP_MeanVar_i32	34	60	78	60
	f32	float	R_DSP_MeanVar_f32	34	40	78	40
Variance	i16	int16_t	R_DSP_Var_GivenMean_i16	14	12	48	12
	i32	int32_t	R_DSP_Var_GivenMean_i32	14	28	48	28
	f32	float	R_DSP_Var_GivenMean_f32	14	8	48	8
Histogram	i16	uint16_t	R_DSP_Histogram_i16ui16	856	44	1023	48
	i32	uint16_t	R_DSP_Histogram_i32ui16	1309	60	1516	64
	f32	uint16_t	R_DSP_Histogram_f32ui16	1166	36	1338	40
Mean value and mean absolute deviation (MAD)	i16	int16_t	R_DSP_MeanMAD_i16	34	60	78	60
	i32	int32_t	R_DSP_MeanMAD_i32	34	60	78	60
	f32	float	R_DSP_MeanMAD_f32	34	40	78	40
Mean absolute deviation (MAD)	i16	int16_t	R_DSP_MAD_GivenMean_i16	14	12	48	12
	i32	int32_t	R_DSP_MAD_GivenMean_i32	14	12	48	12
	f32	float	R_DSP_MAD_GivenMean_f32	14	8	48	8
Median functions	i16	int16_t	R_DSP_Median_InPlace_i16	69	36	111	36
	i32	int32_t	R_DSP_Median_InPlace_i32	78	36	120	36
	f32	float	R_DSP_Median_InPlace_f32	78	36	119	36
	i16	int16_t	R_DSP_Median_i16	184	36	230	36
	i32	int32_t	R_DSP_Median_i32	193	36	239	36
	f32	float	R_DSP_Median_f32	193	36	239	36

3.2 Filter Operation API

3.2.1 Generic FIR Filter

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
i16	i16	R_DSP_FIR_Init_i16i16	81	4	115	4
		R_DSP_FIR_i16i16	346	68	495	68
		R_DSP_FIR_i16i16_asm_nt	67	32	80	32
		R_DSP_FIR_i16i16_asm_n2	71	32	84	32
		R_DSP_FIR_i16i16_asm_st	84	32	90	32
		R_DSP_FIR_i16i16_asm_s2	88	32	94	32
	i32	R_DSP_FIR_Init_i16i32	81	4	115	4
		R_DSP_FIR_i16i32	678	68	868	76
		R_DSP_FIR_i16i32_asm_ntn	54	32	67	36
		R_DSP_FIR_i16i32_asm_ntu	62	32	79	36
		R_DSP_FIR_i16i32_asm_ntd	70	32	76	36
		R_DSP_FIR_i16i32_asm_n2n	same as ntn			
		R_DSP_FIR_i16i32_asm_n2u	same as ntu			
		R_DSP_FIR_i16i32_asm_n2d	74	32	84	36
		R_DSP_FIR_i16i32_asm_stn	70	32	76	36
		R_DSP_FIR_i16i32_asm_stu	80	32	86	36
		R_DSP_FIR_i16i32_asm_std	79	32	85	36
		R_DSP_FIR_i16i32_asm_s2n	same as stn			
		R_DSP_FIR_i16i32_asm_s2u	same as stu			
		R_DSP_FIR_i16i32_asm_s2d	85	32	91	36
ci16	ci16	R_DSP_FIR_Init_ci16ci16	123	4	157	4
		R_DSP_FIR_ci16ci16	2343	76	2855	84
		R_DSP_FIR_ci16ci16_asm_ntn	144	36	173	40
		R_DSP_FIR_ci16ci16_asm_ntu	180	36	211	40
		R_DSP_FIR_ci16ci16_asm_ntd	180	36	211	40
		R_DSP_FIR_ci16ci16_asm_n2n	157	36	217	40
		R_DSP_FIR_ci16ci16_asm_n2u	188	36	219	40
		R_DSP_FIR_ci16ci16_asm_n2d	188	36	219	40
		R_DSP_FIR_ci16ci16_asm_stn	151	36	179	40
		R_DSP_FIR_ci16ci16_asm_stu	220	36	235	40
		R_DSP_FIR_ci16ci16_asm_std	220	36	235	40
		R_DSP_FIR_ci16ci16_asm_s2n	151	36	211	40
		R_DSP_FIR_ci16ci16_asm_s2u	228	36	243	40
		R_DSP_FIR_ci16ci16_asm_s2d	228	36	243	40

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
ci16	ci32	R_DSP_FIR_Init_ci16ci32	123	4	157	4
		R_DSP_FIR_ci16ci32	1567	76	1849	84
		R_DSP_FIR_ci16ci32_asm_ntn	144	36	170	40
		R_DSP_FIR_ci16ci32_asm_ntu	156	36	192	40
		R_DSP_FIR_ci16ci32_asm_ntd	180	36	195	40
		R_DSP_FIR_ci16ci32_asm_n2n	same as ntn			
		R_DSP_FIR_ci16ci32_asm_n2u	same as ntu			
		R_DSP_FIR_ci16ci32_asm_n2d	188	36	213	40
		R_DSP_FIR_ci16ci32_asm_stn	180	36	195	40
		R_DSP_FIR_ci16ci32_asm_stu	195	36	210	40
		R_DSP_FIR_ci16ci32_asm_std	204	36	219	40
		R_DSP_FIR_ci16ci32_asm_s2n	same as stn			
		R_DSP_FIR_ci16ci32_asm_s2u	same as stu			
		R_DSP_FIR_ci16ci32_asm_s2d	216	36	231	40
i32	i32	R_DSP_FIR_Init_i32i32	93	4	127	4
		R_DSP_FIR_i32i32	1605	76	1965	84
		R_DSP_FIR_i32i32_asm_ntn	101	36	116	40
		R_DSP_FIR_i32i32_asm_ntu	123	36	143	40
		R_DSP_FIR_i32i32_asm_ntd	124	36	134	40
		R_DSP_FIR_i32i32_asm_n2n	110	36	142	40
		R_DSP_FIR_i32i32_asm_n2u	127	36	154	40
		R_DSP_FIR_i32i32_asm_n2d	128	36	143	40
		R_DSP_FIR_i32i32_asm_stn	104	36	119	40
		R_DSP_FIR_i32i32_asm_stu	143	36	154	40
		R_DSP_FIR_i32i32_asm_std	136	36	147	40
		R_DSP_FIR_i32i32_asm_s2n	104	36	136	40
		R_DSP_FIR_i32i32_asm_s2u	150	36	160	40
		R_DSP_FIR_i32i32_asm_s2d	143	36	154	40
ci32	ci32	R_DSP_FIR_Init_ci32ci32	123	4	157	4
		R_DSP_FIR_ci32ci32	2614	76	3113	84
		R_DSP_FIR_ci32ci32_asm_ntn	170	36	195	40
		R_DSP_FIR_ci32ci32_asm_ntu	203	36	238	40
		R_DSP_FIR_ci32ci32_asm_ntd	204	36	219	40
		R_DSP_FIR_ci32ci32_asm_n2n	188	36	247	40
		R_DSP_FIR_ci32ci32_asm_n2u	211	36	256	40
		R_DSP_FIR_ci32ci32_asm_n2d	212	36	237	40
		R_DSP_FIR_ci32ci32_asm_stn	176	36	201	40
		R_DSP_FIR_ci32ci32_asm_stu	241	36	256	40
		R_DSP_FIR_ci32ci32_asm_std	228	36	243	40
		R_DSP_FIR_ci32ci32_asm_s2n	176	36	235	40
		R_DSP_FIR_ci32ci32_asm_s2u	253	36	268	40
		R_DSP_FIR_ci32ci32_asm_s2d	240	36	255	40
f32	f32	R_DSP_FIR_Init_f32f32	93	4	127	4
		R_DSP_FIR_f32f32	117	76	220	76
		R_DSP_FIR_f32f32_asm	113	36	113	36
cf32	cf32	R_DSP_FIR_Init_cf32cf32	123	4	157	4
		R_DSP_FIR_cf32cf32	226	92	329	92
		R_DSP_FIR_cf32cf32_asm	222	44	222	44

3.2.2 IIR Biquad Filter

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
i16	i16	R_DSP_IIRBiquad_StateSize_i16i16	19	4	42	4
		R_DSP_IIRBiquad_Init_i16i16	96	4	131	4
		R_DSP_IIRBiquad_i16i16	6479	92	8472	100
		R_DSP_IIRBiquad_i16i16_asm_nt0n	339	44	402	48
		R_DSP_IIRBiquad_i16i16_asm_nt0d	389	44	468	48
		R_DSP_IIRBiquad_i16i16_asm_nt0u	same as nt0d			
		R_DSP_IIRBiquad_i16i16_asm_n20n	393	44	556	48
		R_DSP_IIRBiquad_i16i16_asm_n20d	442	44	605	48
		R_DSP_IIRBiquad_i16i16_asm_n20u	same as n20d			
		R_DSP_IIRBiquad_i16i16_asm_st0n	357	44	420	48
		R_DSP_IIRBiquad_i16i16_asm_st0d	444	44	507	48
		R_DSP_IIRBiquad_i16i16_asm_st0u	same as st0d			
		R_DSP_IIRBiquad_i16i16_asm_s20n	357	44	520	48
		R_DSP_IIRBiquad_i16i16_asm_s20d	452	44	599	48
		R_DSP_IIRBiquad_i16i16_asm_s20u	same as s20d			
		R_DSP_IIRBiquad_i16i16_asm_nt1n	339	44	402	48
		R_DSP_IIRBiquad_i16i16_asm_nt1d	389	44	468	48
		R_DSP_IIRBiquad_i16i16_asm_nt1u	same as nt1d			
		R_DSP_IIRBiquad_i16i16_asm_n21n	393	44	556	48
		R_DSP_IIRBiquad_i16i16_asm_n21d	442	44	605	48
		R_DSP_IIRBiquad_i16i16_asm_n21u	same as n21d			
		R_DSP_IIRBiquad_i16i16_asm_st1n	357	44	420	48
		R_DSP_IIRBiquad_i16i16_asm_st1d	444	44	507	48
		R_DSP_IIRBiquad_i16i16_asm_st1u	same as st1d			
		R_DSP_IIRBiquad_i16i16_asm_s21n	357	44	520	48
		R_DSP_IIRBiquad_i16i16_asm_s21d	452	44	599	48
		R_DSP_IIRBiquad_i16i16_asm_s21u	same as s21d			

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
i16	i32	R_DSP_IIRBiquad_StateSize_i16i32	19	4	42	4
		R_DSP_IIRBiquad_Init_i16i32	96	4	131	4
		R_DSP_IIRBiquad_i16i32	10353	76	13252	84
		R_DSP_IIRBiquad_i16i32_asm_nt0n	364	36	437	40
		R_DSP_IIRBiquad_i16i32_asm_nt0u	385	36	468	40
		R_DSP_IIRBiquad_i16i32_asm_nt0d	411	36	484	40
		R_DSP_IIRBiquad_i16i32_asm_n20n	409	36	566	40
		R_DSP_IIRBiquad_i16i32_asm_n20u	430	36	597	40
		R_DSP_IIRBiquad_i16i32_asm_n20d	464	36	621	40
		R_DSP_IIRBiquad_i16i32_asm_st0n	413	36	476	40
		R_DSP_IIRBiquad_i16i32_asm_st0u	438	36	501	40
		R_DSP_IIRBiquad_i16i32_asm_st0d	454	36	517	40
		R_DSP_IIRBiquad_i16i32_asm_s20n	413	36	560	40
		R_DSP_IIRBiquad_i16i32_asm_s20u	438	36	585	40
		R_DSP_IIRBiquad_i16i32_asm_s20d	462	36	609	40
		R_DSP_IIRBiquad_i16i32_asm_nt1n	364	36	437	40
		R_DSP_IIRBiquad_i16i32_asm_nt1u	385	36	468	40
		R_DSP_IIRBiquad_i16i32_asm_nt1d	411	36	484	40
		R_DSP_IIRBiquad_i16i32_asm_n21n	409	36	566	40
		R_DSP_IIRBiquad_i16i32_asm_n21u	430	36	597	40
		R_DSP_IIRBiquad_i16i32_asm_n21d	464	36	621	40
		R_DSP_IIRBiquad_i16i32_asm_st1n	413	36	476	40
		R_DSP_IIRBiquad_i16i32_asm_st1u	438	36	501	40
		R_DSP_IIRBiquad_i16i32_asm_st1d	454	36	517	40
		R_DSP_IIRBiquad_i16i32_asm_s21n	413	36	560	40
		R_DSP_IIRBiquad_i16i32_asm_s21u	438	36	585	40
		R_DSP_IIRBiquad_i16i32_asm_s21d	462	36	609	40

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
ci16	ci16	R_DSP_IIRBiquad_StateSize_ci16ci16	19	4	42	4
		R_DSP_IIRBiquad_Init_ci16ci16	108	4	144	4
		R_DSP_IIRBiquad_ci16ci16	11575	108	15302	116
		R_DSP_IIRBiquad_ci16ci16_asm_nt0n	598	48	724	52
		R_DSP_IIRBiquad_ci16ci16_asm_nt0d	701	52	857	56
		R_DSP_IIRBiquad_ci16ci16_asm_nt0u	same as nt0d			
		R_DSP_IIRBiquad_ci16ci16_asm_n20n	708	48	1026	52
		R_DSP_IIRBiquad_ci16ci16_asm_n20d	808	52	1124	56
		R_DSP_IIRBiquad_ci16ci16_asm_n20u	same as n20d			
		R_DSP_IIRBiquad_ci16ci16_asm_st0n	634	48	760	52
		R_DSP_IIRBiquad_ci16ci16_asm_st0d	811	52	935	56
		R_DSP_IIRBiquad_ci16ci16_asm_st0u	same as st0d			
		R_DSP_IIRBiquad_ci16ci16_asm_s20n	634	48	954	52
		R_DSP_IIRBiquad_ci16ci16_asm_s20d	827	52	1112	56
		R_DSP_IIRBiquad_ci16ci16_asm_s20u	same as s20d			
		R_DSP_IIRBiquad_ci16ci16_asm_nt1n	598	48	724	52
		R_DSP_IIRBiquad_ci16ci16_asm_nt1d	701	52	857	56
		R_DSP_IIRBiquad_ci16ci16_asm_nt1u	same as nt1d			
		R_DSP_IIRBiquad_ci16ci16_asm_n21n	708	48	1026	52
		R_DSP_IIRBiquad_ci16ci16_asm_n21d	808	52	1124	56
		R_DSP_IIRBiquad_ci16ci16_asm_n21u	same as n21d			
		R_DSP_IIRBiquad_ci16ci16_asm_st1n	634	48	760	52
		R_DSP_IIRBiquad_ci16ci16_asm_st1d	811	52	935	56
		R_DSP_IIRBiquad_ci16ci16_asm_st1u	same as st1d			
		R_DSP_IIRBiquad_ci16ci16_asm_s21n	634	48	954	52
		R_DSP_IIRBiquad_ci16ci16_asm_s21d	827	52	1112	56
		R_DSP_IIRBiquad_ci16ci16_asm_s21u	same as s21d			

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
ci16	ci32	R_DSP_IIRBiquad_StateSize_ci16ci32	19	4	42	4
		R_DSP_IIRBiquad_Init_ci16ci32	108	4	144	4
		R_DSP_IIRBiquad_ci16ci32	18313	100	23794	108
		R_DSP_IIRBiquad_ci16ci32_asm_nt0n	640	48	787	52
		R_DSP_IIRBiquad_ci16ci32_asm_nt0u	673	48	840	52
		R_DSP_IIRBiquad_ci16ci32_asm_nt0d	729	48	874	52
		R_DSP_IIRBiquad_ci16ci32_asm_n20n	731	48	1038	52
		R_DSP_IIRBiquad_ci16ci32_asm_n20u	764	48	1091	52
		R_DSP_IIRBiquad_ci16ci32_asm_n20d	836	48	1141	52
		R_DSP_IIRBiquad_ci16ci32_asm_st0n	740	48	865	52
		R_DSP_IIRBiquad_ci16ci32_asm_st0u	781	48	906	52
		R_DSP_IIRBiquad_ci16ci32_asm_st0d	815	48	940	52
		R_DSP_IIRBiquad_ci16ci32_asm_s20n	740	48	1026	52
		R_DSP_IIRBiquad_ci16ci32_asm_s20u	781	48	1067	52
		R_DSP_IIRBiquad_ci16ci32_asm_s20d	831	48	1117	52
		R_DSP_IIRBiquad_ci16ci32_asm_nt1n	640	48	787	52
		R_DSP_IIRBiquad_ci16ci32_asm_nt1u	673	48	840	52
		R_DSP_IIRBiquad_ci16ci32_asm_nt1d	729	48	874	52
		R_DSP_IIRBiquad_ci16ci32_asm_n21n	731	48	1038	52
		R_DSP_IIRBiquad_ci16ci32_asm_n21u	764	48	1091	52
		R_DSP_IIRBiquad_ci16ci32_asm_n21d	836	48	1141	52
		R_DSP_IIRBiquad_ci16ci32_asm_st1n	740	48	865	52
		R_DSP_IIRBiquad_ci16ci32_asm_st1u	781	48	906	52
		R_DSP_IIRBiquad_ci16ci32_asm_st1d	815	48	940	52
		R_DSP_IIRBiquad_ci16ci32_asm_s21n	740	48	1026	52
		R_DSP_IIRBiquad_ci16ci32_asm_s21u	781	48	1067	52
		R_DSP_IIRBiquad_ci16ci32_asm_s21d	831	48	1117	52

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
i32	i32	R_DSP_IIRBiquad_StateSize_i32i32	19	4	42	4
		R_DSP_IIRBiquad_Init_i32i32	108	4	144	4
		R_DSP_IIRBiquad_i32i32	10529	100	14483	108
		R_DSP_IIRBiquad_i32i32_asm_nt0n	358	44	463	48
		R_DSP_IIRBiquad_i32i32_asm_nt0u	412	48	545	52
		R_DSP_IIRBiquad_i32i32_asm_nt0d	413	48	512	52
		R_DSP_IIRBiquad_i32i32_asm_n20n	412	44	623	48
		R_DSP_IIRBiquad_i32i32_asm_n20u	465	48	697	52
		R_DSP_IIRBiquad_i32i32_asm_n20d	466	48	664	52
		R_DSP_IIRBiquad_i32i32_asm_st0n	376	44	481	48
		R_DSP_IIRBiquad_i32i32_asm_st0u	479	48	578	52
		R_DSP_IIRBiquad_i32i32_asm_st0d	452	48	551	52
		R_DSP_IIRBiquad_i32i32_asm_s20n	376	44	587	48
		R_DSP_IIRBiquad_i32i32_asm_s20u	491	48	679	52
		R_DSP_IIRBiquad_i32i32_asm_s20d	464	48	652	52
		R_DSP_IIRBiquad_i32i32_asm_nt1n	358	44	463	48
		R_DSP_IIRBiquad_i32i32_asm_nt1u	412	48	545	52
		R_DSP_IIRBiquad_i32i32_asm_nt1d	413	48	512	52
		R_DSP_IIRBiquad_i32i32_asm_n21n	412	44	623	48
		R_DSP_IIRBiquad_i32i32_asm_n21u	465	48	697	52
		R_DSP_IIRBiquad_i32i32_asm_n21d	466	48	664	52
		R_DSP_IIRBiquad_i32i32_asm_st1n	376	44	481	48
		R_DSP_IIRBiquad_i32i32_asm_st1u	479	48	578	52
		R_DSP_IIRBiquad_i32i32_asm_st1d	452	48	551	52
		R_DSP_IIRBiquad_i32i32_asm_s21n	376	44	587	48
		R_DSP_IIRBiquad_i32i32_asm_s21u	491	48	679	52
		R_DSP_IIRBiquad_i32i32_asm_s21d	464	48	652	52

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
ci32	ci32	R_DSP_IIRBiquad_StateSize_ci32ci32	19	4	42	4
		R_DSP_IIRBiquad_Init_ci32ci32	141	4	176	4
		R_DSP_IIRBiquad_ci32ci32	21481	108	29003	116
		R_DSP_IIRBiquad_ci32ci32_asm_nt0n	752	48	962	52
		R_DSP_IIRBiquad_ci32ci32_asm_nt0u	845	52	1107	56
		R_DSP_IIRBiquad_ci32ci32_asm_nt0d	846	52	1040	56
		R_DSP_IIRBiquad_ci32ci32_asm_n20n	862	48	1274	52
		R_DSP_IIRBiquad_ci32ci32_asm_n20u	952	52	1403	56
		R_DSP_IIRBiquad_ci32ci32_asm_n20d	953	52	1336	56
		R_DSP_IIRBiquad_ci32ci32_asm_st0n	788	48	998	52
		R_DSP_IIRBiquad_ci32ci32_asm_st0u	979	52	1173	56
		R_DSP_IIRBiquad_ci32ci32_asm_st0d	924	52	1118	56
		R_DSP_IIRBiquad_ci32ci32_asm_s20n	788	48	1202	52
		R_DSP_IIRBiquad_ci32ci32_asm_s20u	1003	52	1367	56
		R_DSP_IIRBiquad_ci32ci32_asm_s20d	948	52	1312	56
		R_DSP_IIRBiquad_ci32ci32_asm_nt1n	752	48	962	52
		R_DSP_IIRBiquad_ci32ci32_asm_nt1u	845	52	1107	56
		R_DSP_IIRBiquad_ci32ci32_asm_nt1d	846	52	1040	56
		R_DSP_IIRBiquad_ci32ci32_asm_n21n	862	48	1274	52
		R_DSP_IIRBiquad_ci32ci32_asm_n21u	952	52	1403	56
		R_DSP_IIRBiquad_ci32ci32_asm_n21d	953	52	1336	56
		R_DSP_IIRBiquad_ci32ci32_asm_st1n	788	48	998	52
		R_DSP_IIRBiquad_ci32ci32_asm_st1u	979	52	1173	56
		R_DSP_IIRBiquad_ci32ci32_asm_st1d	924	52	1118	56
		R_DSP_IIRBiquad_ci32ci32_asm_s21n	788	48	1202	52
		R_DSP_IIRBiquad_ci32ci32_asm_s21u	1003	52	1367	56
		R_DSP_IIRBiquad_ci32ci32_asm_s21d	948	52	1312	56
f32	f32	R_DSP_IIRBiquad_StateSize_f32f32	19	4	42	4
		R_DSP_IIRBiquad_Init_f32f32	110	4	143	4
		R_DSP_IIRBiquad_f32f32	409	108	519	108
		R_DSP_IIRBiquad_f32f32_asm	405	52	405	52
cf32	cf32	R_DSP_IIRBiquad_StateSize_cf32cf32	19	4	42	4
		R_DSP_IIRBiquad_Init_cf32cf32	140	4	173	4
		R_DSP_IIRBiquad_cf32cf32	969	124	1079	124
		R_DSP_IIRBiquad_cf32cf32_asm	965	60	965	60

3.2.3 Single-Pole IIR Filter

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
i16	i16	R_DSP_IIRSinglePole_i16i16	416	36	566	44
		R_DSP_IIRSinglePole_i16i16_asm_nt	81	16	90	20
		R_DSP_IIRSinglePole_i16i16_asm_n2	91	16	100	20
		R_DSP_IIRSinglePole_i16i16_asm_st	102	16	108	20
		R_DSP_IIRSinglePole_i16i16_asm_s2	106	16	112	20
i16	i32	R_DSP_IIRSinglePole_i16i32	459	44	605	52
		R_DSP_IIRSinglePole_i16i32_asm_nt	94	20	101	24
		R_DSP_IIRSinglePole_i16i32_asm_n2	105	20	111	24
		R_DSP_IIRSinglePole_i16i32_asm_st	110	20	116	24
		R_DSP_IIRSinglePole_i16i32_asm_s2	114	20	120	24
i32	i32	R_DSP_IIRSinglePole_i32i32	705	60	893	68
		R_DSP_IIRSinglePole_i32i32_asm_nt	150	28	167	32
		R_DSP_IIRSinglePole_i32i32_asm_n2	166	28	186	32
		R_DSP_IIRSinglePole_i32i32_asm_st	173	28	187	32
		R_DSP_IIRSinglePole_i32i32_asm_s2	180	28	200	32
f32	f32	R_DSP_IIRSinglePole_f32f32	90	44	217	44
		R_DSP_IIRSinglePole_f32f32_asm	86	20	86	20

3.3 Linear Transform API

3.3.1 Discrete Fourier Transform (DFT) / Inverse Discrete Fourier Transform (IDFT)

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
Complex DFT	ci16	ci16	R_DSP_DFT_ci16ci16	50	136	91	136
		ci32	R_DSP_DFT_ci16ci32	50	140	91	140
	ci32		R_DSP_DFT_ci32ci32	50	140	91	140
	cf32	cf32	R_DSP_DFT_cf32cf32	50	236	91	236
Complex IDFT	ci16	ci16	R_DSP_IDFT_ci16ci16	50	136	91	136
	ci32		R_DSP_IDFT_ci32ci16	50	140	91	140
		ci32	R_DSP_IDFT_ci32ci32	50	140	91	140
	cf32	cf32	R_DSP_IDFT_cf32cf32	50	240	91	240
Real DFT	i16	ci16	R_DSP_DFT_i16ci16	50	136	95	136
		ci32	R_DSP_DFT_i16ci32	50	136	95	136
	i32		R_DSP_DFT_i32ci32	50	136	95	136
	f32	cf32	R_DSP_DFT_f32cf32	50	204	95	204
Complex Conjugate Symmetry IDFT	ci16	ci16	R_DSP_IDFT_CCS_ci16i16	52	136	94	136
	ci32		R_DSP_IDFT_CCS_ci32i16	52	136	94	136
		ci32	R_DSP_IDFT_CCS_ci32i32	52	136	94	136
	cf32	cf32	R_DSP_IDFT_CCS_cf32f32	52	232	94	232

3.3.2 FFT / IFFT Memory Size Acquisition Functions

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
i16	ci16	R_DSP_FFT_BufSize_i16ci16	172	12	206	12
	ci32	R_DSP_FFT_BufSize_i16ci32	172	12	206	12
i32	ci32	R_DSP_FFT_BufSize_i32ci32	162	8	198	8
f32	cf32	R_DSP_FFT_BufSize_f32cf32	162	8	181	8
ci16	i16	R_DSP_FFT_BufSize_ci16i16	172	12	206	12
	ci16	R_DSP_FFT_BufSize_ci16ci16	162	8	197	8
	ci32	R_DSP_FFT_BufSize_ci16ci32	162	8	197	8
ci32	i16	R_DSP_FFT_BufSize_ci32i16	172	12	206	12
	i32	R_DSP_FFT_BufSize_ci32i32	same as i32ci32			
	ci16	R_DSP_FFT_BufSize_ci32ci16	162	8	197	8
	ci32	R_DSP_FFT_BufSize_ci32ci32	142	4	182	4
cf32	f32	R_DSP_FFT_BufSize_cf32f32	162	8	181	8
	cf32	R_DSP_FFT_BufSize_cf32cf32	142	4	167	4

3.3.3 FFT / IFFT Initialization Functions

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
i16	ci16	R_DSP_FFT_Init_i16ci16	1331	68	1371	68
	ci32	R_DSP_FFT_Init_i16ci32	1331	68	1371	68
i32	ci32	R_DSP_FFT_Init_i32ci32	775	68	815	68
f32	cf32	R_DSP_FFT_Init_f32cf32	844	76	870	76
ci16	i16	R_DSP_FFT_Init_ci16i16	1303	68	1343	68
	ci16	R_DSP_FFT_Init_ci16ci16	418	44	457	44
	ci32	R_DSP_FFT_Init_ci16ci32	418	44	457	44
ci32	i16	R_DSP_FFT_Init_ci32i16	1303	68	1343	68
	i32	R_DSP_FFT_Init_ci32i32	763	68	803	68
	ci16	R_DSP_FFT_Init_ci32ci16	418	44	457	44
	ci32	R_DSP_FFT_Init_ci32ci32	323	44	363	44
cf32	f32	R_DSP_FFT_Init_cf32f32	674	76	700	76
	cf32	R_DSP_FFT_Init_cf32cf32	335	68	361	68

3.3.4 FFT / IFFT Operation Functions

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
cFFT	ci16	ci16	R_DSP_FFT_ci16ci16	7599	116	10001	116
			R_cfft_ci16ci16	1247	56	1672	56
			R_cfft_sc_ci16ci16	1243	56	1596	56
			R_cfft_x2_ci16ci16	1223	56	1576	56
			R_cfft_tw32_ci16ci16	1261	56	1654	56
			R_cfft_sc_tw32_ci16ci16	1281	56	1674	56
			R_cfft_x2_tw32_ci16ci16	1277	56	1670	56
		ci32	R_DSP_FFT_ci16ci32	6090	116	7984	116
			R_cfft_ci16ci32	1010	56	1385	56
			R_cfft_sc_ci16ci32	1026	56	1289	56
			R_cfft_x2_ci16ci32	986	56	1249	56
			R_cfft_tw32_ci16ci32	1003	56	1378	56
			R_cfft_sc_tw32_ci16ci32	1019	56	1282	56
			R_cfft_x2_tw32_ci16ci32	979	56	1242	56
	ci32	ci32	R_DSP_FFT_ci32ci32	3908	116	4920	116
			R_cfft_ci32ci32	1295	56	1672	56
			R_cfft_sc_ci32ci32	1311	56	1576	56
			R_cfft_x2_ci32ci32	1271	56	1536	56
	cf32	cf32	R_DSP_FFT_cf32cf32	1272	116	1367	116
			R_cfft_cf32cf32	1264	56	1264	56
icFFT	ci16	ci16	R_DSP_IFFT_ci16ci16	7599	116	10001	116
			R_icfft_ci16ci16	1247	56	1672	56
			R_icfft_sc_ci16ci16	1243	56	1596	56
			R_icfft_x2_ci16ci16	1223	56	1576	56
			R_icfft_tw32_ci16ci16	1261	56	1654	56
			R_icfft_sc_tw32_ci16ci16	1281	56	1674	56
			R_icfft_x2_tw32_ci16ci16	1277	56	1670	56
	ci32		R_DSP_IFFT_ci32ci16	6465	116	8855	116
			R_icfft_ci32ci16	1058	56	1481	56
			R_icfft_sc_ci32ci16	1054	56	1405	56
			R_icfft_x2_ci32ci16	1034	56	1385	56
			R_icfft_tw32_ci32ci16	1072	56	1463	56
			R_icfft_sc_tw32_ci32ci16	1092	56	1483	56
			R_icfft_x2_tw32_ci32ci16	1088	56	1479	56
		ci32	R_DSP_IFFT_ci32ci32	3908	116	4920	116
			R_icfft_ci32ci32	1295	56	1672	56
			R_icfft_sc_ci32ci32	1311	56	1576	56
			R_icfft_x2_ci32ci32	1271	56	1536	56
	cf32	cf32	R_DSP_IFFT_cf32cf32	1272	116	1367	116
			R_icfft_cf32cf32	1264	56	1264	56

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
rFFT	i16	ci16	R_DSP_FFT_i16ci16	10182	116	13102	116
			R_rfft_i16ci16	1665	56	2176	56
			R_rfft_sc_i16ci16	1661	56	2100	56
			R_rfft_x2_i16ci16	1641	56	2080	56
			R_rfft_tw32_i16ci16	1704	56	2183	56
			R_rfft_sc_tw32_i16ci16	1724	56	2203	56
			R_rfft_x2_tw32_i16ci16	1720	56	2199	56
		ci32	R_DSP_FFT_i16ci32	7686	116	9849	116
			R_rfft_i16ci32	1281	56	1700	56
			R_rfft_sc_i16ci32	1297	56	1604	56
			R_rfft_x2_i16ci32	1257	56	1564	56
			R_rfft_tw32_i16ci32	1264	56	1684	56
			R_rfft_sc_tw32_i16ci32	1280	56	1588	56
			R_rfft_x2_tw32_i16ci32	1240	56	1548	56
	i32	ci32	R_DSP_FFT_i32ci32	5261	116	6521	116
			R_rfft_i32ci32	1746	56	2205	56
			R_rfft_sc_i32ci32	1762	56	2109	56
			R_rfft_x2_i32ci32	1722	56	2069	56
	f32	cf32	R_DSP_FFT_f32cf32	1646	116	1743	116
			R_rfft_f32cf32	1638	56	1638	56
irFFT	ci16	i16	R_DSP_IFFT_CCS_ci16i16	10182	116	13103	116
			R_irfft_ci16i16	1665	56	2176	56
			R_irfft_sc_ci16i16	1661	56	2100	56
			R_irfft_x2_ci16i16	1641	56	2080	56
			R_irfft_tw32_ci16i16	1704	56	2183	56
			R_irfft_sc_tw32_ci16i16	1724	56	2203	56
			R_irfft_x2_tw32_ci16i16	1720	56	2199	56
	ci32		R_DSP_IFFT_CCS_ci32i16	7497	116	10112	116
			R_irfft_ci32i16	1225	56	1685	56
			R_irfft_sc_ci32i16	1221	56	1609	56
			R_irfft_x2_ci32i16	1201	56	1589	56
			R_irfft_tw32_ci32i16	1249	56	1677	56
			R_irfft_sc_tw32_ci32i16	1269	56	1697	56
			R_irfft_x2_tw32_ci32i16	1265	56	1693	56
		i32	R_DSP_IFFT_CCS_ci32i32	5261	116	6522	116
			R_irfft_ci32i32	1746	56	2205	56
			R_irfft_sc_ci32i32	1762	56	2109	56
			R_irfft_x2_ci32i32	1722	56	2069	56
	cf32	f32	R_DSP_IFFT_CCS_cf32f32	1646	116	1744	116
			R_irfft_cf32f32	1638	56	1638	56

3.4 Complex Number Operation API

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
Complex number magnitude	ci16	i16	R_DSP_CplxMag_ci16i16	18	4	18	4
		i32	R_DSP_CplxMag_ci16i32	18	4	18	4
	ci32		R_DSP_CplxMag_ci32i32	21	4	21	4
	cf32	f32	R_DSP_CplxMag_cf32f32	13	4	13	4
	ci16	i16	R_DSP_CplxMag_Fast_ci16i16	35	4	35	4
		i32	R_DSP_CplxMag_Fast_ci16i32	31	4	31	4
	ci32		R_DSP_CplxMag_Fast_ci32i32	90	4	90	4
	cf32	f32	R_DSP_CplxMag_Fast_cf32f32	53	4	53	4
	ci16	i16	R_DSP_VecCplxMag_ci16i16	846	72	904	72
		i32	R_DSP_VecCplxMag_ci16i32	848	72	906	72
	ci32		R_DSP_VecCplxMag_ci32i32	607	64	665	64
	cf32	f32	R_DSP_VecCplxMag_cf32f32	607	64	665	64
	ci16	i16	R_DSP_VecCplxMag_Fast_ci16i16	846	72	904	72
		i32	R_DSP_VecCplxMag_Fast_ci16i32	848	72	906	72
	ci32		R_DSP_VecCplxMag_Fast_ci32i32	607	64	665	64
	cf32	f32	R_DSP_VecCplxMag_Fast_cf32f32	607	64	665	64
Complex number magnitude squared	ci16	i16	R_DSP_CplxMagSquared_ci16i16	14	4	14	4
		i32	R_DSP_CplxMagSquared_ci16i32	12	4	12	4
	ci32		R_DSP_CplxMagSquared_ci32i32	14	4	14	4
	cf32	f32	R_DSP_CplxMagSquared_cf32f32	10	4	10	4
	ci16	i16	R_DSP_VecCplxMagSquared_ci16i16	846	72	904	72
		i32	R_DSP_VecCplxMagSquared_ci16i32	848	72	906	72
	ci32		R_DSP_VecCplxMagSquared_ci32i32	607	64	665	64
	cf32	f32	R_DSP_VecCplxMagSquared_cf32f32	607	64	665	64
Complex number phase	ci16	i16	R_DSP_CplxPhase_ci16i16	91	28	91	28
	ci32	i32	R_DSP_CplxPhase_ci32i32	92	28	92	28
	cf32	f32	R_DSP_CplxPhase_cf32f32	61	60	61	60
	ci16	i16	R_DSP_VecCplxPhase_ci16i16	1145	100	1204	100
	ci32	i32	R_DSP_VecCplxPhase_ci32i32	910	100	968	100
	cf32	f32	R_DSP_VecCplxPhase_cf32f32	1083	136	1142	136
Complex number addition	ci16	ci16	R_DSP_ComplexAdd_ci16ci16	19	4	40	4
	ci32	ci32	R_DSP_ComplexAdd_ci32ci32	21	8	39	8
	cf32	cf32	R_DSP_ComplexAdd_cf32cf32	23	8	31	8
Complex number subtraction	ci16	ci16	R_DSP_ComplexSub_ci16ci16	19	4	40	4
	ci32	ci32	R_DSP_ComplexSub_ci32ci32	21	8	39	8
	cf32	cf32	R_DSP_ComplexSub_cf32cf32	23	8	31	8
Complex number multiplication	ci16	ci16	R_DSP_ComplexMul_ci16ci16	25	4	72	4
	ci32		R_DSP_ComplexMul_ci16ci32	25	4	72	4
		ci32	R_DSP_ComplexMul_ci32ci32	35	8	83	8
	cf32	cf32	R_DSP_ComplexMul_cf32cf32	39	8	47	8
Complex conjugate	ci16	ci16	R_DSP_ComplexConjg_ci16ci16	14	4	29	4
	ci32	ci32	R_DSP_ComplexConjg_ci32ci32	9	4	22	4
	cf32	cf32	R_DSP_ComplexConjg_cf32cf32	10	4	17	4
	ci16	ci16	R_DSP_VecCplxConjg_ci16ci16	862	72	905	72
	ci32	ci32	R_DSP_VecCplxConjg_ci32ci32	711	76	753	76
	cf32	cf32	R_DSP_VecCplxConjg_cf32cf32	651	68	691	68

3.5 Matrix Operation API

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
Matrix addition	i16	i16	R_DSP_MatrixAdd_i16i16	125	24	260	24
		i32	R_DSP_MatrixAdd_i16i32	123	20	209	20
	i32		R_DSP_MatrixAdd_i32i32	125	24	239	24
	f32	f32	R_DSP_MatrixAdd_f32f32	130	24	216	24
	ci16	ci16	R_DSP_MatrixAdd_ci16ci16	202	24	388	24
		ci32	R_DSP_MatrixAdd_ci16ci32	200	20	286	20
	ci32		R_DSP_MatrixAdd_ci32ci32	202	24	344	24
	cf32	cf32	R_DSP_MatrixAdd_cf32cf32	214	24	300	24
Matrix subtraction	i16	i16	R_DSP_MatrixSub_i16i16	125	24	260	24
		i32	R_DSP_MatrixSub_i16i32	123	20	209	20
	i32		R_DSP_MatrixSub_i32i32	125	24	239	24
	f32	f32	R_DSP_MatrixSub_f32f32	130	24	216	24
	ci16	ci16	R_DSP_MatrixSub_ci16ci16	202	24	388	24
		ci32	R_DSP_MatrixSub_ci16ci32	200	20	286	20
	ci32		R_DSP_MatrixSub_ci32ci32	202	24	344	24
	cf32	cf32	R_DSP_MatrixSub_cf32cf32	214	24	300	24
Matrix multiplication	i16	i16	R_DSP_MatrixMul_i16i16	775	76	940	88
			R_DSP_MatrixMul_i16i16_asm_nt	172	36	192	40
			R_DSP_MatrixMul_i16i16_asm_n2	176	36	196	40
			R_DSP_MatrixMul_i16i16_asm_st	193	36	205	40
			R_DSP_MatrixMul_i16i16_asm_s2	198	36	209	40
		i32	R_DSP_MatrixMul_i16i32	860	76	1031	88
			R_DSP_MatrixMul_i16i32_asm_nt	194	36	215	40
			R_DSP_MatrixMul_i16i32_asm_n2	198	36	224	40
			R_DSP_MatrixMul_i16i32_asm_st	213	36	224	40
			R_DSP_MatrixMul_i16i32_asm_s2	219	36	230	40
	i32	i32	R_DSP_MatrixMul_i32i32	1110	76	1389	88
			R_DSP_MatrixMul_i32i32_asm_nt	230	36	300	40
			R_DSP_MatrixMul_i32i32_asm_n2	239	36	324	40
			R_DSP_MatrixMul_i32i32_asm_st	295	36	306	40
			R_DSP_MatrixMul_i32i32_asm_s2	310	36	321	40
	f32	f32	R_DSP_MatrixMul_f32f32	186	76	272	76
			R_DSP_MatrixMul_f32f32_asm	182	36	182	36
	ci16	ci16	R_DSP_MatrixMul_ci16ci16	1204	76	1398	88
			R_DSP_MatrixMul_ci16ci16_asm_nt	268	36	299	40
			R_DSP_MatrixMul_ci16ci16_asm_n2	276	36	307	40
			R_DSP_MatrixMul_ci16ci16_asm_st	308	36	323	40
			R_DSP_MatrixMul_ci16ci16_asm_s2	316	36	331	40
		ci32	R_DSP_MatrixMul_ci16ci32	1336	76	1544	88
			R_DSP_MatrixMul_ci16ci32_asm_nt	302	36	335	40
			R_DSP_MatrixMul_ci16ci32_asm_n2	310	36	353	40
			R_DSP_MatrixMul_ci16ci32_asm_st	338	36	353	40
			R_DSP_MatrixMul_ci16ci32_asm_s2	350	36	365	40

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
Matrix multiplication	ci32	ci32	R_DSP_MatrixMul_ci32ci32	1928	84	2348	96
			R_DSP_MatrixMul_ci32ci32_asm_nt	396	40	527	44
			R_DSP_MatrixMul_ci32ci32_asm_n2	414	40	575	44
			R_DSP_MatrixMul_ci32ci32_asm_st	526	40	539	44
			R_DSP_MatrixMul_ci32ci32_asm_s2	556	40	569	44
	cf32	cf32	R_DSP_MatrixMul_cf32cf32	378	100	464	100
			R_DSP_MatrixMul_cf32cf32_asm	374	48	374	48
Matrix transposition	i16	i16	R_DSP_MatrixTrans_i16i16	104	36	150	36
		i32	R_DSP_MatrixTrans_i16i32	105	36	151	36
	i32		R_DSP_MatrixTrans_i32i32	104	36	150	36
	f32	f32	R_DSP_MatrixTrans_f32f32	104	36	150	36
	ci16	ci16	R_DSP_MatrixTrans_ci16ci16	150	36	196	36
		ci32	R_DSP_MatrixTrans_ci16ci32	150	36	196	36
	ci32		R_DSP_MatrixTrans_ci32ci32	150	36	196	36
	cf32	cf32	R_DSP_MatrixTrans_cf32cf32	150	36	196	36
Matrix real number multiplication	i16	i16	R_DSP_MatrixScale_i16i16	632	40	751	48
			R_DSP_MatrixScale_i16i16_asm_nt	125	12	141	20
			R_DSP_MatrixScale_i16i16_asm_n2	133	12	149	20
			R_DSP_MatrixScale_i16i16_asm_st	161	16	171	20
			R_DSP_MatrixScale_i16i16_asm_s2	169	16	179	20
		i32	R_DSP_MatrixScale_i16i32	688	40	795	48
			R_DSP_MatrixScale_i16i32_asm_nt	146	16	156	20
			R_DSP_MatrixScale_i16i32_asm_n2	154	16	164	20
			R_DSP_MatrixScale_i16i32_asm_st	168	16	178	20
			R_DSP_MatrixScale_i16i32_asm_s2	176	16	186	20
	i32	i32	R_DSP_MatrixScale_i32i32	1066	72	1277	80
			R_DSP_MatrixScale_i32i32_asm_nt	192	24	230	28
			R_DSP_MatrixScale_i32i32_asm_n2	224	24	278	28
			R_DSP_MatrixScale_i32i32_asm_st	274	28	292	32
			R_DSP_MatrixScale_i32i32_asm_s2	340	32	374	36
	f32	f32	R_DSP_MatrixScale_f32f32	128	48	186	48
			R_DSP_MatrixScale_f32f32_asm	124	20	124	20
	ci16	ci16	R_DSP_MatrixScale_ci16ci16	876	40	951	48
			R_DSP_MatrixScale_ci16ci16_asm_nt	208	12	210	20
			R_DSP_MatrixScale_ci16ci16_asm_n2	208	12	210	20
			R_DSP_MatrixScale_ci16ci16_asm_st	208	16	210	20
			R_DSP_MatrixScale_ci16ci16_asm_s2	208	16	210	20
		ci32	R_DSP_MatrixScale_ci16ci32	888	40	963	48
			R_DSP_MatrixScale_ci16ci32_asm_nt	211	16	213	20
			R_DSP_MatrixScale_ci16ci32_asm_n2	211	16	213	20
			R_DSP_MatrixScale_ci16ci32_asm_st	211	16	213	20
			R_DSP_MatrixScale_ci16ci32_asm_s2	211	16	213	20
	ci32	ci32	R_DSP_MatrixScale_ci32ci32	924	72	999	80
			R_DSP_MatrixScale_ci32ci32_asm_nt	222	24	224	28
			R_DSP_MatrixScale_ci32ci32_asm_n2	222	24	224	28
			R_DSP_MatrixScale_ci32ci32_asm_st	222	28	224	32
			R_DSP_MatrixScale_ci32ci32_asm_s2	222	32	224	36
	cf32	cf32	R_DSP_MatrixScale_cf32cf32	212	48	270	48
			R_DSP_MatrixScale_cf32cf32_asm	208	20	208	20

4. Execution Cycle Count

This section shows the result of execution cycle counts for each function.

The measurement conditions are

- Device: RX64M Group
- Library: R_DSP_FPU_LE.lib
- Code allocation: Code flash memory
- Data allocation: Internal RAM (Data is allocated to 4-byte boundary alignment sections)

Target functions

Filter operation APIs

- Generic FIR (real number)
- IIR Biquad (real number)

Transform kernels

- Complex FFT
- Complex IFFT
- Real FFT
- Complex conjugate symmetric IFFT

4.1 Filter operation API

4.1.1 Generic FIR

Target functions

- R_DSP_FIR_i16i16 and its internal functions
- R_DSP_FIR_i16i32 and its internal functions
- R_DSP_FIR_i32i32 and its internal functions
- R_DSP_FIR_f32f32 and its internal functions

(1) **R_DSP_FIR_i16i16**

Taps=16, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	497	953	1865	3689	7337	14633
		NEAREST	502	966	1894	3750	7462	14886
	SATURATE	TRUNC	527	1015	1991	3943	7847	15655
		NEAREST	534	1030	2022	4006	7974	15910
R_DSP_FIR_i16i16_asm_nt	-		485	941	1853	3677	7325	14621
R_DSP_FIR_i16i16_asm_n2			492	956	1884	3740	7452	14876
R_DSP_FIR_i16i16_asm_st			517	1005	1981	3933	7837	15645
R_DSP_FIR_i16i16_asm_s2			525	1021	2013	3997	7965	15901

Taps=32, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	833	1625	3209	6377	12713	25385
		NEAREST	838	1638	3238	6438	12838	25638
	SATURATE	TRUNC	863	1687	3335	6631	13223	26407
		NEAREST	870	1702	3366	6694	13350	26662
R_DSP_FIR_i16i16_asm_nt	-		821	1613	3197	6365	12701	25373
R_DSP_FIR_i16i16_asm_n2			828	1628	3228	6428	12828	25628
R_DSP_FIR_i16i16_asm_st			853	1677	3325	6621	13213	26397
R_DSP_FIR_i16i16_asm_s2			861	1693	3357	6685	13341	26653

Taps=64, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	1505	2969	5897	11753	23465	46889
		NEAREST	1510	2982	5926	11814	23590	47142
	SATURATE	TRUNC	1535	3031	6023	12007	23975	47911
		NEAREST	1542	3046	6054	12070	24102	48166
R_DSP_FIR_i16i16_asm_nt	-		1493	2957	5885	11741	23453	46877
R_DSP_FIR_i16i16_asm_n2			1500	2972	5916	11804	23580	47132
R_DSP_FIR_i16i16_asm_st			1525	3021	6013	11997	23965	47901
R_DSP_FIR_i16i16_asm_s2			1533	3037	6045	12061	24093	48157

Taps=128, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	2849	5657	11273	22505	44969	89897
		NEAREST	2854	5670	11302	22566	45094	90150
	SATURATE	TRUNC	2879	5719	11399	22759	45479	90919
		NEAREST	2886	5734	11430	22822	45606	91174
R_DSP_FIR_i16i16_asm_nt	-		2837	5645	11261	22493	44957	89885
R_DSP_FIR_i16i16_asm_n2			2844	5660	11292	22556	45084	90140
R_DSP_FIR_i16i16_asm_st			2869	5709	11389	22749	45469	90909
R_DSP_FIR_i16i16_asm_s2			2877	5725	11421	22813	45597	91165

Taps=256, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	5537	11033	22025	44009	87977	175913
		NEAREST	5542	11046	22054	44070	88102	176166
	SATURATE	TRUNC	5567	11095	22151	44263	88487	176935
		NEAREST	5574	11110	22182	44326	88614	177190
R_DSP_FIR_i16i16_asm_nt	-		5525	11021	22013	43997	87965	175901
R_DSP_FIR_i16i16_asm_n2			5532	11036	22044	44060	88092	176156
R_DSP_FIR_i16i16_asm_st			5557	11085	22141	44253	88477	176925
R_DSP_FIR_i16i16_asm_s2			5565	11101	22173	44317	88605	177181

(2) **R_DSP_FIR_i16i32**

Taps=16

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	NO (scale=0)	490	930	1810	3570	7090	14130
			UP (scale=-1)	494	942	1838	3630	7214	14382
			DOWN (scale=1)	504	960	1872	3696	7344	14640
		NEAREST	NO (scale=0)	486	926	1806	3566	7086	14126
			UP (scale=-1)	490	938	1834	3626	7210	14378
			DOWN (scale=1)	509	973	1901	3757	7469	14893
	SATURATE	TRUNC	NO (scale=0)	517	989	1933	3821	7597	15149
			UP (scale=-1)	529	1017	1993	3945	7849	15657
			DOWN (scale=1)	516	988	1932	3820	7596	15148
		NEAREST	NO (scale=0)	514	986	1930	3818	7594	15146
			UP (scale=-1)	526	1014	1990	3942	7846	15654
			DOWN (scale=1)	520	1000	1960	3880	7720	15400
R_DSP_FIR_i16i32_asm_ntn	-		0	468	908	1788	3548	7068	14108
R_DSP_FIR_i16i32_asm_ntu			-1	477	925	1821	3613	7197	14365
R_DSP_FIR_i16i32_asm_ntd			1	484	940	1852	3676	7324	14620
R_DSP_FIR_i16i32_asm_n2n			0	same as ntn					
R_DSP_FIR_i16i32_asm_n2u			-1	same as ntu					
R_DSP_FIR_i16i32_asm_n2d			1	493	957	1885	3741	7453	14877
R_DSP_FIR_i16i32_asm_stn			0	500	972	1916	3804	7580	15132
R_DSP_FIR_i16i32_asm_stu			-1	517	1005	1981	3933	7837	15645
R_DSP_FIR_i16i32_asm_std			1	501	973	1917	3805	7581	15133
R_DSP_FIR_i16i32_asm_s2n			0	same as stn					
R_DSP_FIR_i16i32_asm_s2u			-1	same as stu					
R_DSP_FIR_i16i32_asm_s2d			1	508	988	1948	3868	7708	15388

Taps=32

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	NO (scale=0)	826	1602	3154	6258	12466	24882
			UP (scale=-1)	830	1614	3182	6318	12590	25134
			DOWN (scale=1)	840	1632	3216	6384	12720	25392
		NEAREST	NO (scale=0)	822	1598	3150	6254	12462	24878
			UP (scale=-1)	826	1610	3178	6314	12586	25130
			DOWN (scale=1)	845	1645	3245	6445	12845	25645
	SATURATE	TRUNC	NO (scale=0)	853	1661	3277	6509	12973	25901
			UP (scale=-1)	865	1689	3337	6633	13225	26409
			DOWN (scale=1)	852	1660	3276	6508	12972	25900
		NEAREST	NO (scale=0)	850	1658	3274	6506	12970	25898
			UP (scale=-1)	862	1686	3334	6630	13222	26406
			DOWN (scale=1)	856	1672	3304	6568	13096	26152
R_DSP_FIR_i16i32_asm_ntn	-		0	804	1580	3132	6236	12444	24860
R_DSP_FIR_i16i32_asm_ntu			-1	813	1597	3165	6301	12573	25117
R_DSP_FIR_i16i32_asm_ntd			1	820	1612	3196	6364	12700	25372
R_DSP_FIR_i16i32_asm_n2n			0	same as ntn					
R_DSP_FIR_i16i32_asm_n2u			-1	same as ntu					
R_DSP_FIR_i16i32_asm_n2d			1	829	1629	3229	6429	12829	25629
R_DSP_FIR_i16i32_asm_stn			0	836	1644	3260	6492	12956	25884
R_DSP_FIR_i16i32_asm_stu			-1	853	1677	3325	6621	13213	26397
R_DSP_FIR_i16i32_asm_std			1	837	1645	3261	6493	12957	25885
R_DSP_FIR_i16i32_asm_s2n			0	same as stn					
R_DSP_FIR_i16i32_asm_s2u			-1	same as stu					
R_DSP_FIR_i16i32_asm_s2d			1	844	1660	3292	6556	13084	26140

Taps=64

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	NO (scale=0)	1498	2946	5842	11634	23218	46386
			UP (scale=-1)	1502	2958	5870	11694	23342	46638
			DOWN (scale=1)	1512	2976	5904	11760	23472	46896
		NEAREST	NO (scale=0)	1494	2942	5838	11630	23214	46382
			UP (scale=-1)	1498	2954	5866	11690	23338	46634
			DOWN (scale=1)	1517	2989	5933	11821	23597	47149
	SATURATE	TRUNC	NO (scale=0)	1525	3005	5965	11885	23725	47405
			UP (scale=-1)	1537	3033	6025	12009	23977	47913
			DOWN (scale=1)	1524	3004	5964	11884	23724	47404
		NEAREST	NO (scale=0)	1522	3002	5962	11882	23722	47402
			UP (scale=-1)	1534	3030	6022	12006	23974	47910
			DOWN (scale=1)	1528	3016	5992	11944	23848	47656
R_DSP_FIR_i16i32_asm_ntn	-		0	1476	2924	5820	11612	23196	46364
R_DSP_FIR_i16i32_asm_ntu			-1	1485	2941	5853	11677	23325	46621
R_DSP_FIR_i16i32_asm_ntd			1	1492	2956	5884	11740	23452	46876
R_DSP_FIR_i16i32_asm_n2n			0	same as ntn					
R_DSP_FIR_i16i32_asm_n2u			-1	same as ntu					
R_DSP_FIR_i16i32_asm_n2d			1	1501	2973	5917	11805	23581	47133
R_DSP_FIR_i16i32_asm_stn			0	1508	2988	5948	11868	23708	47388
R_DSP_FIR_i16i32_asm_stu			-1	1525	3021	6013	11997	23965	47901
R_DSP_FIR_i16i32_asm_std			1	1509	2989	5949	11869	23709	47389
R_DSP_FIR_i16i32_asm_s2n			0	same as stn					
R_DSP_FIR_i16i32_asm_s2u			-1	same as stu					
R_DSP_FIR_i16i32_asm_s2d			1	1516	3004	5980	11932	23836	47644

Taps=128

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	NO (scale=0)	2842	5634	11218	22386	44722	89394
			UP (scale=-1)	2846	5646	11246	22446	44846	89646
			DOWN (scale=1)	2856	5664	11280	22512	44976	89904
		NEAREST	NO (scale=0)	2838	5630	11214	22382	44718	89390
			UP (scale=-1)	2842	5642	11242	22442	44842	89642
			DOWN (scale=1)	2861	5677	11309	22573	45101	90157
	SATURATE	TRUNC	NO (scale=0)	2869	5693	11341	22637	45229	90413
			UP (scale=-1)	2881	5721	11401	22761	45481	90921
			DOWN (scale=1)	2868	5692	11340	22636	45228	90412
		NEAREST	NO (scale=0)	2866	5690	11338	22634	45226	90410
			UP (scale=-1)	2878	5718	11398	22758	45478	90918
			DOWN (scale=1)	2872	5704	11368	22696	45352	90664
R_DSP_FIR_i16i32_asm_ntn	-		0	2820	5612	11196	22364	44700	89372
R_DSP_FIR_i16i32_asm_ntu			-1	2829	5629	11229	22429	44829	89629
R_DSP_FIR_i16i32_asm_ntd			1	2836	5644	11260	22492	44956	89884
R_DSP_FIR_i16i32_asm_n2n			0	same as ntn					
R_DSP_FIR_i16i32_asm_n2u			-1	same as ntu					
R_DSP_FIR_i16i32_asm_n2d			1	2845	5661	11293	22557	45085	90141
R_DSP_FIR_i16i32_asm_stn			0	2852	5676	11324	22620	45212	90396
R_DSP_FIR_i16i32_asm_stu			-1	2869	5709	11389	22749	45469	90909
R_DSP_FIR_i16i32_asm_std			1	2853	5677	11325	22621	45213	90397
R_DSP_FIR_i16i32_asm_s2n			0	same as stn					
R_DSP_FIR_i16i32_asm_s2u			-1	same as stu					
R_DSP_FIR_i16i32_asm_s2d			1	2860	5692	11356	22684	45340	90652

Taps=256

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	NO (scale=0)	5530	11010	21970	43890	87730	175410
			UP (scale=-1)	5534	11022	21998	43950	87854	175662
			DOWN (scale=1)	5544	11040	22032	44016	87984	175920
		NEAREST	NO (scale=0)	5526	11006	21966	43886	87726	175406
			UP (scale=-1)	5530	11018	21994	43946	87850	175658
			DOWN (scale=1)	5549	11053	22061	44077	88109	176173
	SATURATE	TRUNC	NO (scale=0)	5557	11069	22093	44141	88237	176429
			UP (scale=-1)	5569	11097	22153	44265	88489	176937
			DOWN (scale=1)	5556	11068	22092	44140	88236	176428
		NEAREST	NO (scale=0)	5554	11066	22090	44138	88234	176426
			UP (scale=-1)	5566	11094	22150	44262	88486	176934
			DOWN (scale=1)	5560	11080	22120	44200	88360	176680
R_DSP_FIR_i16i32_asm_ntn	-		0	5508	10988	21948	43868	87708	175388
R_DSP_FIR_i16i32_asm_ntu			-1	5517	11005	21981	43933	87837	175645
R_DSP_FIR_i16i32_asm_ntd			1	5524	11020	22012	43996	87964	175900
R_DSP_FIR_i16i32_asm_n2n			0	same as ntn					
R_DSP_FIR_i16i32_asm_n2u			-1	same as ntu					
R_DSP_FIR_i16i32_asm_n2d			1	5533	11037	22045	44061	88093	176157
R_DSP_FIR_i16i32_asm_stn			0	5540	11052	22076	44124	88220	176412
R_DSP_FIR_i16i32_asm_stu			-1	5557	11085	22141	44253	88477	176925
R_DSP_FIR_i16i32_asm_std			1	5541	11053	22077	44125	88221	176413
R_DSP_FIR_i16i32_asm_s2n			0	same as stn					
R_DSP_FIR_i16i32_asm_s2u			-1	same as stu					
R_DSP_FIR_i16i32_asm_s2d			1	5548	11068	22108	44188	88348	176668

(3) **R_DSP_FIR_i32i32**

Taps=16

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	NO (scale=31)	544	1040	2032	4016	7984	15920
			UP (scale=30)	559	1071	2095	4143	8239	16431
			DOWN (scale=32)	561	1073	2097	4145	8241	16433
		NEAREST	NO (scale=31)	565	1085	2125	4205	8365	16685
			UP (scale=30)	564	1084	2124	4204	8364	16684
			DOWN (scale=32)	567	1087	2127	4207	8367	16687
	SATURATE	TRUNC	NO (scale=31)	550	1054	2062	4078	8110	16174
			UP (scale=30)	596	1148	2252	4460	8876	17708
			DOWN (scale=32)	583	1119	2191	4335	8623	17199
		NEAREST	NO (scale=31)	546	1050	2058	4074	8106	16170
			UP (scale=30)	611	1179	2315	4587	9131	18219
			DOWN (scale=32)	595	1147	2251	4459	8875	17707
R_DSP_FIR_i32i32_asm_ntn	-		31	524	1020	2012	3996	7964	15900
R_DSP_FIR_i32i32_asm_ntu			30	544	1056	2080	4128	8224	16416
R_DSP_FIR_i32i32_asm_ntd			32	543	1055	2079	4127	8223	16415
R_DSP_FIR_i32i32_asm_n2n			31	548	1068	2108	4188	8348	16668
R_DSP_FIR_i32i32_asm_n2u			30	552	1072	2112	4192	8352	16672
R_DSP_FIR_i32i32_asm_n2d			32	552	1072	2112	4192	8352	16672
R_DSP_FIR_i32i32_asm_stn			31	533	1037	2045	4061	8093	16157
R_DSP_FIR_i32i32_asm_stu			30	584	1136	2240	4448	8864	17696
R_DSP_FIR_i32i32_asm_std			32	568	1104	2176	4320	8608	17184
R_DSP_FIR_i32i32_asm_s2n			31	532	1036	2044	4060	8092	16156
R_DSP_FIR_i32i32_asm_s2u			30	600	1168	2304	4576	9120	18208
R_DSP_FIR_i32i32_asm_s2d			32	583	1135	2239	4447	8863	17695

Taps=32

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	NO (scale=31)	992	1936	3824	7600	15152	30256
			UP (scale=30)	1007	1967	3887	7727	15407	30767
			DOWN (scale=32)	1009	1969	3889	7729	15409	30769
		NEAREST	NO (scale=31)	1013	1981	3917	7789	15533	31021
			UP (scale=30)	1012	1980	3916	7788	15532	31020
			DOWN (scale=32)	1015	1983	3919	7791	15535	31023
	SATURATE	TRUNC	NO (scale=31)	998	1950	3854	7662	15278	30510
			UP (scale=30)	1044	2044	4044	8044	16044	32044
			DOWN (scale=32)	1031	2015	3983	7919	15791	31535
		NEAREST	NO (scale=31)	994	1946	3850	7658	15274	30506
			UP (scale=30)	1059	2075	4107	8171	16299	32555
			DOWN (scale=32)	1043	2043	4043	8043	16043	32043
R_DSP_FIR_i32i32_asm_ntn	-		31	972	1916	3804	7580	15132	30236
R_DSP_FIR_i32i32_asm_ntu			30	992	1952	3872	7712	15392	30752
R_DSP_FIR_i32i32_asm_ntd			32	991	1951	3871	7711	15391	30751
R_DSP_FIR_i32i32_asm_n2n			31	996	1964	3900	7772	15516	31004
R_DSP_FIR_i32i32_asm_n2u			30	1000	1968	3904	7776	15520	31008
R_DSP_FIR_i32i32_asm_n2d			32	1000	1968	3904	7776	15520	31008
R_DSP_FIR_i32i32_asm_stn			31	981	1933	3837	7645	15261	30493
R_DSP_FIR_i32i32_asm_stu			30	1032	2032	4032	8032	16032	32032
R_DSP_FIR_i32i32_asm_std			32	1016	2000	3968	7904	15776	31520
R_DSP_FIR_i32i32_asm_s2n			31	980	1932	3836	7644	15260	30492
R_DSP_FIR_i32i32_asm_s2u			30	1048	2064	4096	8160	16288	32544
R_DSP_FIR_i32i32_asm_s2d			32	1031	2031	4031	8031	16031	32031

Taps=64

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	NO (scale=31)	1888	3728	7408	14768	29488	58928
			UP (scale=30)	1903	3759	7471	14895	29743	59439
			DOWN (scale=32)	1905	3761	7473	14897	29745	59441
		NEAREST	NO (scale=31)	1909	3773	7501	14957	29869	59693
			UP (scale=30)	1908	3772	7500	14956	29868	59692
			DOWN (scale=32)	1911	3775	7503	14959	29871	59695
	SATURATE	TRUNC	NO (scale=31)	1894	3742	7438	14830	29614	59182
			UP (scale=30)	1940	3836	7628	15212	30380	60716
			DOWN (scale=32)	1927	3807	7567	15087	30127	60207
		NEAREST	NO (scale=31)	1890	3738	7434	14826	29610	59178
			UP (scale=30)	1955	3867	7691	15339	30635	61227
			DOWN (scale=32)	1939	3835	7627	15211	30379	60715
R_DSP_FIR_i32i32_asm_ntn	-		31	1868	3708	7388	14748	29468	58908
R_DSP_FIR_i32i32_asm_ntu			30	1888	3744	7456	14880	29728	59424
R_DSP_FIR_i32i32_asm_ntd			32	1887	3743	7455	14879	29727	59423
R_DSP_FIR_i32i32_asm_n2n			31	1892	3756	7484	14940	29852	59676
R_DSP_FIR_i32i32_asm_n2u			30	1896	3760	7488	14944	29856	59680
R_DSP_FIR_i32i32_asm_n2d			32	1896	3760	7488	14944	29856	59680
R_DSP_FIR_i32i32_asm_stn			31	1877	3725	7421	14813	29597	59165
R_DSP_FIR_i32i32_asm_stu			30	1928	3824	7616	15200	30368	60704
R_DSP_FIR_i32i32_asm_std			32	1912	3792	7552	15072	30112	60192
R_DSP_FIR_i32i32_asm_s2n			31	1876	3724	7420	14812	29596	59164
R_DSP_FIR_i32i32_asm_s2u			30	1944	3856	7680	15328	30624	61216
R_DSP_FIR_i32i32_asm_s2d			32	1927	3823	7615	15199	30367	60703

Taps=128

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	NO (scale=31)	3680	7312	14576	29104	58160	116272
			UP (scale=30)	3695	7343	14639	29231	58415	116783
			DOWN (scale=32)	3697	7345	14641	29233	58417	116785
		NEAREST	NO (scale=31)	3701	7357	14669	29293	58541	117037
			UP (scale=30)	3700	7356	14668	29292	58540	117036
			DOWN (scale=32)	3703	7359	14671	29295	58543	117039
	SATURATE	TRUNC	NO (scale=31)	3686	7326	14606	29166	58286	116526
			UP (scale=30)	3732	7420	14796	29548	59052	118060
			DOWN (scale=32)	3719	7391	14735	29423	58799	117551
		NEAREST	NO (scale=31)	3682	7322	14602	29162	58282	116522
			UP (scale=30)	3747	7451	14859	29675	59307	118571
			DOWN (scale=32)	3731	7419	14795	29547	59051	118059
R_DSP_FIR_i32i32_asm_ntn	-		31	3660	7292	14556	29084	58140	116252
R_DSP_FIR_i32i32_asm_ntu			30	3680	7328	14624	29216	58400	116768
R_DSP_FIR_i32i32_asm_ntd			32	3679	7327	14623	29215	58399	116767
R_DSP_FIR_i32i32_asm_n2n			31	3684	7340	14652	29276	58524	117020
R_DSP_FIR_i32i32_asm_n2u			30	3688	7344	14656	29280	58528	117024
R_DSP_FIR_i32i32_asm_n2d			32	3688	7344	14656	29280	58528	117024
R_DSP_FIR_i32i32_asm_stn			31	3669	7309	14589	29149	58269	116509
R_DSP_FIR_i32i32_asm_stu			30	3720	7408	14784	29536	59040	118048
R_DSP_FIR_i32i32_asm_std			32	3704	7376	14720	29408	58784	117536
R_DSP_FIR_i32i32_asm_s2n			31	3668	7308	14588	29148	58268	116508
R_DSP_FIR_i32i32_asm_s2u			30	3736	7440	14848	29664	59296	118560
R_DSP_FIR_i32i32_asm_s2d			32	3719	7407	14783	29535	59039	118047

Taps=256

Function name	option			Samples					
	SATURATE	ROUNDING	SCALING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	NO (scale=31)	7264	14480	28912	57776	115504	230960
			UP (scale=30)	7279	14511	28975	57903	115759	231471
			DOWN (scale=32)	7281	14513	28977	57905	115761	231473
		NEAREST	NO (scale=31)	7285	14525	29005	57965	115885	231725
			UP (scale=30)	7284	14524	29004	57964	115884	231724
			DOWN (scale=32)	7287	14527	29007	57967	115887	231727
	SATURATE	TRUNC	NO (scale=31)	7270	14494	28942	57838	115630	231214
			UP (scale=30)	7316	14588	29132	58220	116396	232748
			DOWN (scale=32)	7303	14559	29071	58095	116143	232239
		NEAREST	NO (scale=31)	7266	14490	28938	57834	115626	231210
			UP (scale=30)	7331	14619	29195	58347	116651	233259
			DOWN (scale=32)	7315	14587	29131	58219	116395	232747
R_DSP_FIR_i32i32_asm_ntn	-		31	7244	14460	28892	57756	115484	230940
R_DSP_FIR_i32i32_asm_ntu			30	7264	14496	28960	57888	115744	231456
R_DSP_FIR_i32i32_asm_ntd			32	7263	14495	28959	57887	115743	231455
R_DSP_FIR_i32i32_asm_n2n			31	7268	14508	28988	57948	115868	231708
R_DSP_FIR_i32i32_asm_n2u			30	7272	14512	28992	57952	115872	231712
R_DSP_FIR_i32i32_asm_n2d			32	7272	14512	28992	57952	115872	231712
R_DSP_FIR_i32i32_asm_stn			31	7253	14477	28925	57821	115613	231197
R_DSP_FIR_i32i32_asm_stu			30	7304	14576	29120	58208	116384	232736
R_DSP_FIR_i32i32_asm_std			32	7288	14544	29056	58080	116128	232224
R_DSP_FIR_i32i32_asm_s2n			31	7252	14476	28924	57820	115612	231196
R_DSP_FIR_i32i32_asm_s2u			30	7320	14608	29184	58336	116640	233248
R_DSP_FIR_i32i32_asm_s2d			32	7303	14575	29119	58207	116383	232735

(4) **R_DSP_FIR_f32f32**

Scale=1.0f

Function name	Taps	Samples					
		8	16	32	64	128	256
R_DSP_FIR_f32f32	16	840	1648	3264	6496	12960	25888
R_DSP_FIR_f32f32_asm		837	1645	3261	6493	12957	25885
R_DSP_FIR_f32f32	32	1608	3184	6336	12640	25248	50464
R_DSP_FIR_f32f32_asm		1605	3181	6333	12637	25245	50461
R_DSP_FIR_f32f32	64	3144	6256	12480	24928	49824	99616
R_DSP_FIR_f32f32_asm		3141	6253	12477	24925	49821	99613
R_DSP_FIR_f32f32	128	6216	12400	24768	49504	98976	197920
R_DSP_FIR_f32f32_asm		6213	12397	24765	49501	98973	197917
R_DSP_FIR_f32f32	256	12360	24688	49344	98656	197280	394528
R_DSP_FIR_f32f32_asm		12357	24685	49341	98653	197277	394525

4.1.2 Biquad IIR

Target functions

- R_DSP_IIRBiquad_i16i16 and its internal functions
- R_DSP_IIRBiquad_i16i32 and its internal functions
- R_DSP_IIRBiquad_i32i32 and its internal functions
- R_DSP_IIRBiquad_f32f32 and its internal functions

Measurement conditions

- Filter type: default

  Fixed point function: form-I

  Floating point function: form-II
- qint: 1

(1) **R_DSP_IIRBiquad_i16i16**

stages=1

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i16	NO SATURATE	TRUNC	14	79	89	107	143	215	359	647	1223	2375
			13	90	106	136	196	316	556	1036	1996	3916
		NEAREST	14	81	94	116	160	248	424	776	1480	2888
			13	89	109	145	217	361	649	1225	2377	4681
	SATURATE	TRUNC	14	78	90	110	150	230	390	710	1350	2630
			13	94	115	155	235	395	715	1355	2635	5195
		NEAREST	14	75	86	106	146	226	386	706	1346	2626
			13	92	114	156	240	408	744	1416	2760	5448
R_DSP_IIRBiquad_i16i16_asm_nt1n	-		14	58	68	86	122	194	338	626	1202	2354
R_DSP_IIRBiquad_i16i16_asm_nt1u			13	67	83	113	173	293	533	1013	1973	3893
R_DSP_IIRBiquad_i16i16_asm_nt1d			same as nt1u									
R_DSP_IIRBiquad_i16i16_asm_n21n			14	63	76	98	142	230	406	758	1462	2870
R_DSP_IIRBiquad_i16i16_asm_n21u			13	69	89	125	197	341	629	1205	2357	4661
R_DSP_IIRBiquad_i16i16_asm_n21d			same as n21u									
R_DSP_IIRBiquad_i16i16_asm_st1n			14	60	72	92	132	212	372	692	1332	2612
R_DSP_IIRBiquad_i16i16_asm_st1u			13	74	95	135	215	375	695	1335	2615	5175
R_DSP_IIRBiquad_i16i16_asm_st1d			same as st1u									
R_DSP_IIRBiquad_i16i16_asm_s21n			14	60	71	91	131	211	371	691	1331	2611
R_DSP_IIRBiquad_i16i16_asm_s21u			13	74	96	138	222	390	726	1398	2742	5430
R_DSP_IIRBiquad_i16i16_asm_s21d			same as s21u									

stages=2

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i16	NO SATURATE	TRUNC	14	97	116	150	218	354	626	1170	2258	4434
			13	108	133	179	271	455	823	1559	3031	5975
		NEAREST	14	101	126	170	258	434	786	1490	2898	5714
			13	110	142	200	316	548	1012	1940	3796	7508
	SATURATE	TRUNC	14	96	118	156	232	384	688	1296	2512	4944
			13	112	143	201	317	549	1013	1941	3797	7509
		NEAREST	14	93	114	152	228	380	684	1292	2508	4940
			13	111	143	203	323	563	1043	2003	3923	7763
R_DSP_IIRBiquad_i16i16_asm_nt1n	-		14	76	95	129	197	333	605	1149	2237	4413
R_DSP_IIRBiquad_i16i16_asm_nt1u			13	85	110	156	248	432	800	1536	3008	5952
R_DSP_IIRBiquad_i16i16_asm_nt1d			same as nt1u									
R_DSP_IIRBiquad_i16i16_asm_n21n			14	83	108	152	240	416	768	1472	2880	5696
R_DSP_IIRBiquad_i16i16_asm_n21u			13	90	122	180	296	528	992	1920	3776	7488
R_DSP_IIRBiquad_i16i16_asm_n21d			same as n21u									
R_DSP_IIRBiquad_i16i16_asm_st1n			14	78	100	138	214	366	670	1278	2494	4926
R_DSP_IIRBiquad_i16i16_asm_st1u			13	92	123	181	297	529	993	1921	3777	7489
R_DSP_IIRBiquad_i16i16_asm_st1d			same as st1u									
R_DSP_IIRBiquad_i16i16_asm_s21n			14	78	99	137	213	365	669	1277	2493	4925
R_DSP_IIRBiquad_i16i16_asm_s21u			13	93	125	185	305	545	1025	1985	3905	7745
R_DSP_IIRBiquad_i16i16_asm_s21d			same as s21u									

stages=4

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i16	NO SATURATE	TRUNC	14	141	176	242	374	638	1166	2222	4334	8558
			13	152	193	271	427	739	1363	2611	5107	10099
		NEAREST	14	151	198	286	462	814	1518	2926	5742	11374
			13	161	215	317	521	929	1745	3377	6641	13169
	SATURATE	TRUNC	14	142	182	256	404	700	1292	2476	4844	9580
			13	158	207	301	489	865	1617	3121	6129	12145
		NEAREST	14	139	178	252	400	696	1288	2472	4840	9576
			13	157	207	303	495	879	1647	3183	6255	12399
R_DSP_IIRBiquad_i16i16_asm_nt1n	-		14	120	155	221	353	617	1145	2201	4313	8537
R_DSP_IIRBiquad_i16i16_asm_nt1u			13	129	170	248	404	716	1340	2588	5084	10076
R_DSP_IIRBiquad_i16i16_asm_nt1d			same as nt1u									
R_DSP_IIRBiquad_i16i16_asm_n21n			14	133	180	268	444	796	1500	2908	5724	11356
R_DSP_IIRBiquad_i16i16_asm_n21u			13	141	195	297	501	909	1725	3357	6621	13149
R_DSP_IIRBiquad_i16i16_asm_n21d			same as n21u									
R_DSP_IIRBiquad_i16i16_asm_st1n			14	124	164	238	386	682	1274	2458	4826	9562
R_DSP_IIRBiquad_i16i16_asm_st1u			13	138	187	281	469	845	1597	3101	6109	12125
R_DSP_IIRBiquad_i16i16_asm_st1d			same as st1u									
R_DSP_IIRBiquad_i16i16_asm_s21n			14	124	163	237	385	681	1273	2457	4825	9561
R_DSP_IIRBiquad_i16i16_asm_s21u			13	139	189	285	477	861	1629	3165	6237	12381
R_DSP_IIRBiquad_i16i16_asm_s21d			same as s21u									

(2) **R_DSP_IIRBiquad_i16i32**

stages=1

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i32	NO SATURATE	TRUNC	0	82	94	120	170	270	470	870	1670	3270
			-1	90	104	134	192	308	540	1004	1932	3788
			1	98	115	151	221	361	641	1201	2321	4561
		NEAREST	0	80	94	124	182	298	530	994	1922	3778
			-1	89	106	140	206	338	602	1130	2186	4298
			1	94	115	157	239	403	731	1387	2699	5323
	SATURATE	TRUNC	0	84	100	134	200	332	596	1124	2180	4292
			-1	92	111	152	230	386	698	1322	2570	5066
			1	96	117	161	247	419	763	1451	2827	5579
		NEAREST	0	83	98	133	199	331	595	1123	2179	4291
			-1	88	107	147	225	381	693	1317	2565	5061
			1	95	117	163	253	433	793	1513	2953	5833
R_DSP_IIRBiquad_i16i32_asm_nt1n	-		0	62	74	100	150	250	450	850	1650	3250
R_DSP_IIRBiquad_i16i32_asm_nt1u			-1	67	81	111	169	285	517	981	1909	3765
R_DSP_IIRBiquad_i16i32_asm_nt1d			1	73	90	126	196	336	616	1176	2296	4536
R_DSP_IIRBiquad_i16i32_asm_n21n			0	63	77	107	165	281	513	977	1905	3761
R_DSP_IIRBiquad_i16i32_asm_n21u			-1	69	86	120	186	318	582	1110	2166	4278
R_DSP_IIRBiquad_i16i32_asm_n21d			1	72	93	135	217	381	709	1365	2677	5301
R_DSP_IIRBiquad_i16i32_asm_st1n			0	66	82	116	182	314	578	1106	2162	4274
R_DSP_IIRBiquad_i16i32_asm_st1u			-1	71	90	131	209	365	677	1301	2549	5045
R_DSP_IIRBiquad_i16i32_asm_st1d			1	73	94	138	224	396	740	1428	2804	5556
R_DSP_IIRBiquad_i16i32_asm_s21n			0	69	84	119	185	317	581	1109	2165	4277
R_DSP_IIRBiquad_i16i32_asm_s21u			-1	71	90	130	208	364	676	1300	2548	5044
R_DSP_IIRBiquad_i16i32_asm_s21d			1	76	98	144	234	414	774	1494	2934	5814

stages=2

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i32	NO SATURATE	TRUNC	0	99	121	163	245	409	737	1393	2705	5329
			-1	108	131	177	267	447	807	1527	2967	5847
			1	115	142	194	296	500	908	1724	3356	6620
		NEAREST	0	101	127	179	281	485	893	1709	3341	6605
			-1	109	139	195	305	525	965	1845	3605	7125
			1	115	148	212	338	590	1094	2102	4118	8150
	SATURATE	TRUNC	0	103	129	181	283	487	895	1711	3343	6607
			-1	111	140	199	313	541	997	1909	3733	7381
			1	115	146	208	330	574	1062	2038	3990	7894
		NEAREST	0	102	127	180	282	486	894	1710	3342	6606
			-1	107	136	194	308	536	992	1904	3728	7376
			1	114	147	211	337	589	1093	2101	4117	8149
R_DSP_IIRBiquad_i16i32_asm_nt1n	-		0	79	101	143	225	389	717	1373	2685	5309
R_DSP_IIRBiquad_i16i32_asm_nt1u			-1	85	108	154	244	424	784	1504	2944	5824
R_DSP_IIRBiquad_i16i32_asm_nt1d			1	90	117	169	271	475	883	1699	3331	6595
R_DSP_IIRBiquad_i16i32_asm_n21n			0	84	110	162	264	468	876	1692	3324	6588
R_DSP_IIRBiquad_i16i32_asm_n21u			-1	89	119	175	285	505	945	1825	3585	7105
R_DSP_IIRBiquad_i16i32_asm_n21d			1	93	126	190	316	568	1072	2080	4096	8128
R_DSP_IIRBiquad_i16i32_asm_st1n			0	85	111	163	265	469	877	1693	3325	6589
R_DSP_IIRBiquad_i16i32_asm_st1u			-1	90	119	178	292	520	976	1888	3712	7360
R_DSP_IIRBiquad_i16i32_asm_st1d			1	92	123	185	307	551	1039	2015	3967	7871
R_DSP_IIRBiquad_i16i32_asm_s21n			0	88	113	166	268	472	880	1696	3328	6592
R_DSP_IIRBiquad_i16i32_asm_s21u			-1	90	119	177	291	519	975	1887	3711	7359
R_DSP_IIRBiquad_i16i32_asm_s21d			1	95	128	192	318	570	1074	2082	4098	8130

stages=4

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i32	NO SATURATE	TRUNC	0	141	179	253	399	691	1275	2443	4779	9451
			-1	150	189	267	421	729	1345	2577	5041	9969
			1	157	200	284	450	782	1446	2774	5430	10742
		NEAREST	0	150	198	294	484	864	1624	3144	6184	12264
			-1	157	209	309	507	903	1695	3279	6447	12783
			1	163	218	326	540	968	1824	3536	6960	13808
	SATURATE	TRUNC	0	147	191	279	453	801	1497	2889	5673	11241
			-1	155	202	297	483	855	1599	3087	6063	12015
			1	159	208	306	500	888	1664	3216	6320	12528
		NEAREST	0	146	189	278	452	800	1496	2888	5672	11240
			-1	151	198	292	478	850	1594	3082	6058	12010
			1	158	209	309	507	903	1695	3279	6447	12783
R_DSP_IIRBiquad_i16i32_asm_nt1n	-		0	121	159	233	379	671	1255	2423	4759	9431
R_DSP_IIRBiquad_i16i32_asm_nt1u			-1	127	166	244	398	706	1322	2554	5018	9946
R_DSP_IIRBiquad_i16i32_asm_nt1d			1	132	175	259	425	757	1421	2749	5405	10717
R_DSP_IIRBiquad_i16i32_asm_n21n			0	133	181	277	467	847	1607	3127	6167	12247
R_DSP_IIRBiquad_i16i32_asm_n21u			-1	137	189	289	487	883	1675	3259	6427	12763
R_DSP_IIRBiquad_i16i32_asm_n21d			1	141	196	304	518	946	1802	3514	6938	13786
R_DSP_IIRBiquad_i16i32_asm_st1n			0	129	173	261	435	783	1479	2871	5655	11223
R_DSP_IIRBiquad_i16i32_asm_st1u			-1	134	181	276	462	834	1578	3066	6042	11994
R_DSP_IIRBiquad_i16i32_asm_st1d			1	136	185	283	477	865	1641	3193	6297	12505
R_DSP_IIRBiquad_i16i32_asm_s21n			0	132	175	264	438	786	1482	2874	5658	11226
R_DSP_IIRBiquad_i16i32_asm_s21u			-1	134	181	275	461	833	1577	3065	6041	11993
R_DSP_IIRBiquad_i16i32_asm_s21d			1	139	190	290	488	884	1676	3260	6428	12764

(3) **R_DSP_IIRBiquad_i32i32**

stages=1

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i32i32	NO SATURATE	TRUNC	30	79	89	107	143	215	359	647	1223	2375
			29	89	104	132	188	300	524	972	1868	3660
			31	90	105	131	183	287	495	911	1743	3407
		NEAREST	30	76	88	110	154	242	418	770	1474	2882
			29	91	109	143	211	347	619	1163	2251	4427
			31	90	107	139	203	331	587	1099	2123	4171
	SATURATE	TRUNC	30	76	87	107	147	227	387	707	1347	2627
			29	95	119	163	251	427	779	1483	2889	5693
			31	91	108	140	204	332	588	1100	2124	4172
		NEAREST	30	74	86	106	146	226	386	706	1346	2626
			29	96	121	167	251	419	755	1429	2781	5472
			31	92	111	147	219	363	651	1227	2379	4683
R_DSP_IIRBiquad_i32i32_asm_nt1n	-		30	59	69	87	123	195	339	627	1203	2355
R_DSP_IIRBiquad_i32i32_asm_nt1u			29	66	81	109	165	277	501	949	1845	3637
R_DSP_IIRBiquad_i32i32_asm_nt1d			31	65	80	106	158	262	470	886	1718	3382
R_DSP_IIRBiquad_i32i32_asm_n21n			30	59	71	93	137	225	401	753	1457	2865
R_DSP_IIRBiquad_i32i32_asm_n21u			29	71	89	123	191	327	599	1143	2231	4407
R_DSP_IIRBiquad_i32i32_asm_n21d			31	68	85	117	181	309	565	1077	2101	4149
R_DSP_IIRBiquad_i32i32_asm_st1n			30	59	70	90	130	210	370	690	1330	2610
R_DSP_IIRBiquad_i32i32_asm_st1u			29	75	99	143	231	407	759	1463	2869	5673
R_DSP_IIRBiquad_i32i32_asm_st1d			31	69	86	118	182	310	566	1078	2102	4150
R_DSP_IIRBiquad_i32i32_asm_s21n			30	60	72	92	132	212	372	692	1332	2612
R_DSP_IIRBiquad_i32i32_asm_s21u			29	78	103	149	233	401	737	1411	2763	5454
R_DSP_IIRBiquad_i32i32_asm_s21d			31	72	91	127	199	343	631	1207	2359	4663

stages=2

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i32i32	NO SATURATE	TRUNC	30	98	117	151	219	355	627	1171	2259	4435
			29	108	132	176	264	440	792	1496	2904	5720
			31	109	134	176	260	428	764	1436	2780	5468
		NEAREST	30	97	122	166	254	430	782	1486	2894	5710
			29	112	142	198	310	534	982	1878	3670	7254
			31	111	141	195	303	519	951	1815	3543	6999
	SATURATE	TRUNC	30	96	117	155	231	383	687	1295	2511	4943
			29	115	149	211	335	583	1079	2071	4049	7997
			31	110	137	187	287	487	887	1687	3287	6487
		NEAREST	30	94	116	154	230	382	686	1294	2510	4942
			29	116	151	215	343	583	1063	2023	3952	7803
			31	112	141	195	303	519	951	1815	3543	6999
R_DSP_IIRBiquad_i32i32_asm_nt1n	-		30	78	97	131	199	335	607	1151	2239	4415
R_DSP_IIRBiquad_i32i32_asm_nt1u			29	85	109	153	241	417	769	1473	2881	5697
R_DSP_IIRBiquad_i32i32_asm_nt1d			31	84	109	151	235	403	739	1411	2755	5443
R_DSP_IIRBiquad_i32i32_asm_n21n			30	80	105	149	237	413	765	1469	2877	5693
R_DSP_IIRBiquad_i32i32_asm_n21u			29	92	122	178	290	514	962	1858	3650	7234
R_DSP_IIRBiquad_i32i32_asm_n21d			31	89	119	173	281	497	929	1793	3521	6977
R_DSP_IIRBiquad_i32i32_asm_st1n			30	79	100	138	214	366	670	1278	2494	4926
R_DSP_IIRBiquad_i32i32_asm_st1u			29	95	129	191	315	563	1059	2051	4029	7977
R_DSP_IIRBiquad_i32i32_asm_st1d			31	88	115	165	265	465	865	1665	3265	6465
R_DSP_IIRBiquad_i32i32_asm_s21n			30	80	102	140	216	368	672	1280	2496	4928
R_DSP_IIRBiquad_i32i32_asm_s21u			29	98	133	197	325	565	1045	2005	3934	7785
R_DSP_IIRBiquad_i32i32_asm_s21d			31	92	121	175	283	499	931	1795	3523	6979

stages=4

Function name	option			Samples								
	SATURATE	ROUNDING	SCALING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i32i32	NO SATURATE	TRUNC	30	145	180	246	378	642	1170	2226	4338	8562
			29	155	195	271	423	727	1335	2551	4983	9847
			31	156	197	271	419	715	1307	2491	4859	9595
		NEAREST	30	148	195	283	459	811	1515	2923	5739	11371
			29	164	216	316	516	916	1716	3316	6516	12916
			31	162	214	312	508	900	1684	3252	6388	12660
	SATURATE	TRUNC	30	145	184	258	406	702	1294	2478	4846	9582
			29	165	217	315	511	903	1687	3255	6385	12641
			31	159	204	290	462	806	1494	2870	5622	11126
		NEAREST	30	144	184	258	406	702	1294	2478	4846	9582
			29	166	219	319	519	917	1685	3221	6300	12462
			31	161	208	298	478	838	1558	2998	5878	11638
R_DSP_IIRBiquad_i32i32_asm_nt1n	-		30	125	160	226	358	622	1150	2206	4318	8542
R_DSP_IIRBiquad_i32i32_asm_nt1u			29	132	172	248	400	704	1312	2528	4960	9824
R_DSP_IIRBiquad_i32i32_asm_nt1d			31	131	172	246	394	690	1282	2466	4834	9570
R_DSP_IIRBiquad_i32i32_asm_n21n			30	131	178	266	442	794	1498	2906	5722	11354
R_DSP_IIRBiquad_i32i32_asm_n21u			29	144	196	296	496	896	1696	3296	6496	12896
R_DSP_IIRBiquad_i32i32_asm_n21d			31	140	192	290	486	878	1662	3230	6366	12638
R_DSP_IIRBiquad_i32i32_asm_st1n			30	128	167	241	389	685	1277	2461	4829	9565
R_DSP_IIRBiquad_i32i32_asm_st1u			29	145	197	295	491	883	1667	3235	6365	12621
R_DSP_IIRBiquad_i32i32_asm_st1d			31	137	182	268	440	784	1472	2848	5600	11104
R_DSP_IIRBiquad_i32i32_asm_s21n			30	130	170	244	392	688	1280	2464	4832	9568
R_DSP_IIRBiquad_i32i32_asm_s21u			29	148	201	301	501	899	1667	3203	6282	12444
R_DSP_IIRBiquad_i32i32_asm_s21d			31	141	188	278	458	818	1538	2978	5858	11618

(4) **R_DSP_IIRBiquad_f32f32**

Scale=1.0f

Function name	Stages	Samples								
		1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_f32f32	1	71	88	118	178	298	538	1018	1978	3898
R_DSP_IIRBiquad_f32f32_asm		68	85	115	175	295	535	1015	1975	3895
R_DSP_IIRBiquad_f32f32	2	94	127	185	301	533	997	1925	3781	7493
R_DSP_IIRBiquad_f32f32_asm		91	124	182	298	530	994	1922	3778	7490
R_DSP_IIRBiquad_f32f32	4	149	210	324	552	1008	1920	3744	7392	14688
R_DSP_IIRBiquad_f32f32_asm		146	207	321	549	1005	1917	3741	7389	14685

4.2 Linear Transform API

Target functions

- R_DSP_FFT (complex and real) and its internal functions
- R_DSP_IFFT and its internal functions
- R_DSP_IFFT_CCS and its internal functions

Measurement conditions

- Window function: no window

(1) Complex FFT

points = 16 - 256

in	out	Function name	option		points				
			TW32	Scale	16	32	64	128	256
ci16	ci16	R_DSP_FFT_ci16ci16	-	-	541	1283	3011	7035	16017
				SC	551	1317	3014	7102	15892
				X2	538	1280	2947	6907	15520
			TW32	-	559	1336	3113	7314	16599
				SC	574	1383	3175	7504	16853
				X2	569	1361	3174	7439	16867
		R_cfft_ci16ci16	-		528	1270	2998	7022	16004
		R_cfft_sc_ci16ci16			538	1304	3001	7089	15879
		R_cfft_x2_ci16ci16			528	1270	2937	6897	15510
		R_cfft_tw32_ci16ci16			543	1319	3097	7298	16583
		R_cfft_sc_tw32_ci16ci16			557	1365	3158	7487	16836
		R_cfft_x2_tw32_ci16ci16			557	1348	3162	7427	16855
	ci32	R_DSP_FFT_ci16ci32	-	-	625	1455	3433	7895	18043
				SC	635	1503	3466	8086	18172
				X2	615	1435	3334	7698	17400
			TW32	-	629	1460	3440	7901	18065
				SC	635	1505	3471	8092	18193
				X2	614	1436	3338	7703	17420
		R_cfft_ci16ci32	-		611	1441	3419	7881	18029
		R_cfft_sc_ci16ci32			620	1488	3451	8071	18157
		R_cfft_x2_ci16ci32			604	1424	3323	7687	17389
		R_cfft_tw32_ci16ci32			612	1442	3423	7884	18048
		R_cfft_sc_tw32_ci16ci32			618	1487	3454	8075	18176
		R_cfft_x2_tw32_ci16ci32			601	1422	3325	7690	17407
ci32	ci32	R_DSP_FFT_ci32ci32	-	-	613	1433	3413	7857	18039
				SC	623	1485	3431	8020	18041
				X2	605	1420	3316	7665	17398
		R_cfft_ci32ci32	-		605	1425	3405	7849	18031
		R_cfft_sc_ci32ci32			613	1475	3421	8010	18031
		R_cfft_x2_ci32ci32			595	1409	3306	7655	17388
cf32	cf32	R_DSP_FFT_cf32cf32	-		637	1502	3706	8486	19884
		R_cfft_cf32cf32	-		632	1497	3701	8481	19879

points = 512 - 8192

in	out	Function name	option		points				
			TW32	Scale	512	1024	2048	4096	8192
ci16	ci16	R_DSP_FFT_ci16ci16	-	-	36373	80731	179159	390365	851913
				SC	36376	79710	178138	384224	843724
				X2	35364	77737	173093	374314	819478
			TW32	-	37852	83809	186590	405731	887760
				SC	38618	84831	189660	409825	900046
				X2	38376	84908	188713	410157	896282
		R_cfft_ci16ci16	-		36360	80718	179146	390352	851900
		R_cfft_sc_ci16ci16			36363	79697	178125	384211	843711
		R_cfft_x2_ci16ci16			35354	77727	173083	374304	819468
		R_cfft_tw32_ci16ci16			37836	83793	186574	405715	887744
		R_cfft_sc_tw32_ci16ci16			38601	84814	189643	409808	900029
		R_cfft_x2_tw32_ci16ci16			38364	84896	188701	410145	896270
	ci32	R_DSP_FFT_ci16ci32	-	-	40489	90157	198267	433279	938765
				SC	41256	90670	201338	435328	951052
				X2	39204	86570	191094	414844	901896
			TW32	-	40510	90242	198351	433619	939104
				SC	41278	90755	201424	435669	951394
				X2	39225	86654	191179	415184	902237
		R_cfft_ci16ci32	-		40475	90143	198253	433265	938751
		R_cfft_sc_ci16ci32			41241	90655	201323	435313	951037
		R_cfft_x2_ci16ci32			39193	86559	191083	414833	901885
		R_cfft_tw32_ci16ci32			40493	90225	198334	433602	939087
		R_cfft_sc_tw32_ci16ci32			41261	90738	201407	435652	951377
		R_cfft_x2_tw32_ci16ci32			39212	86641	191166	415171	902224
ci32	ci32	R_DSP_FFT_ci32ci32	-	-	40483	90409	198773	435323	942855
				SC	40998	90155	200312	433277	946954
				X2	39203	86824	191605	416890	905991
		R_cfft_ci32ci32	-		40475	90401	198765	435315	942847
		R_cfft_sc_ci32ci32			40988	90145	200302	433267	946944
		R_cfft_x2_ci32ci32			39193	86814	191595	416880	905981
cf32	cf32	R_DSP_FFT_cf32cf32	-		44312	100382	219242	485488	1045244
		R_cfft_cf32cf32	-		44307	100377	219237	485483	1045239

(2) **Complex IFFT**

points = 16 - 256

in	out	Function name	option		points				
			TW32	Scale	16	32	64	128	256
ci16	ci16	R_DSP_IFFT_ci16ci16	-	-	658	1514	3483	7972	17960
				SC	670	1549	3471	8007	17709
				X2	655	1511	3413	7838	17427
			TW32	-	673	1561	3566	8215	18411
				SC	689	1609	3628	8404	18650
				X2	686	1591	3624	8338	18646
		R_icfft_ci16ci16	-		644	1499	3469	7958	17946
		R_icfft_sc_ci16ci16			655	1534	3456	7992	17694
		R_icfft_x2_ci16ci16			644	1499	3402	7827	17416
		R_icfft_tw32_ci16ci16			656	1543	3549	8198	18394
		R_icfft_sc_tw32_ci16ci16			672	1591	3611	8387	18633
		R_icfft_x2_tw32_ci16ci16			673	1577	3611	8325	18633
ci32		R_DSP_IFFT_ci32ci16	-	-	711	1620	3689	8387	18775
				SC	717	1647	3690	8450	18663
				X2	708	1614	3606	8221	18132
			TW32	-	725	1666	3770	8627	19224
				SC	743	1715	3838	8823	19484
				X2	737	1692	3832	8752	19478
		R_icfft_ci32ci16	-		698	1606	3676	8374	18762
		R_icfft_sc_ci32ci16			703	1633	3676	8436	18649
		R_icfft_x2_ci32ci16			699	1605	3597	8212	18123
		R_icfft_tw32_ci32ci16			709	1649	3754	8611	19208
		R_icfft_sc_tw32_ci32ci16			726	1697	3821	8806	19467
		R_icfft_x2_tw32_ci32ci16			726	1680	3821	8741	19467
	ci32	R_DSP_IFFT_ci32ci32	-	-	727	1661	3851	8727	19725
				SC	735	1709	3882	8918	19851
				X2	718	1644	3751	8532	19065
		R_icfft_ci32ci32	-		718	1651	3842	8718	19716
		R_icfft_sc_ci32ci32			725	1698	3872	8908	19841
		R_icfft_x2_ci32ci32			708	1633	3741	8522	19055
cf32	cf32	R_DSP_IFFT_cf32cf32	-		788	1781	4241	9533	21955
		R_icfft_cf32cf32	-		784	1777	4237	9529	21951

points = 512 - 8192

in	out	Function name	option		points				
			TW32	Scale	512	1024	2048	4096	8192
ci16	ci16	R_DSP_IFFT_ci16ci16	-	-	40237	88753	195118	423474	917791
				SC	39985	86967	192563	413241	901413
				X2	39192	85597	188890	406751	884684
			TW32	-	41456	91060	201009	434741	945442
				SC	42206	92004	204000	438502	957394
				X2	41948	92000	202974	438498	953296
		R_icfft_ci16ci16	-		40223	88739	195104	423460	917777
		R_icfft_sc_ci16ci16			39970	86952	192548	413226	901398
		R_icfft_x2_ci16ci16			39181	85586	188879	406740	884673
		R_icfft_tw32_ci16ci16			41439	91043	200992	434724	945425
		R_icfft_sc_tw32_ci16ci16			42189	91987	203983	438485	957377
		R_icfft_x2_tw32_ci16ci16			41935	91987	202961	438485	953283
ci32		R_DSP_IFFT_ci32ci16	-	-	41885	92001	201695	436451	944081
				SC	41899	91056	200748	430641	936221
				X2	40599	88158	194009	415968	903115
			TW32	-	43101	94306	207583	447716	971729
				SC	43873	95334	210659	451816	984021
				X2	43610	95328	209628	451810	979918
		R_icfft_ci32ci16	-		41872	91988	201682	436438	944068
		R_icfft_sc_ci32ci16			41885	91042	200734	430627	936207
		R_icfft_x2_ci32ci16			40590	88149	194000	415959	903106
		R_icfft_tw32_ci32ci16			43085	94290	207567	447700	971713
		R_icfft_sc_tw32_ci32ci16			43856	95317	210642	451799	984004
		R_icfft_x2_tw32_ci32ci16			43599	95317	209617	451799	979907
	ci32	R_DSP_IFFT_ci32ci32	-	-	43833	96895	211659	460241	992349
				SC	44599	97404	214728	462285	1004633
				X2	42534	93227	204408	441469	955146
		R_icfft_ci32ci32	-		43824	96886	211650	460232	992340
		R_icfft_sc_ci32ci32			44589	97394	214718	462275	1004623
		R_icfft_x2_ci32ci32			42524	93217	204398	441459	955136
cf32	cf32	R_DSP_IFFT_cf32cf32	-		48431	108597	235649	518279	1110803
		R_icfft_cf32cf32	-		48427	108593	235645	518275	1110799

(3) **Real FFT**

points = 16 - 256

in	out	Function name	option		points				
			TW32	Scale	16	32	64	128	256
i16	ci16	R_DSP_FFT_i16ci16	-	-	409	844	1901	4267	9557
				SC	421	854	1932	4253	9588
				X2	412	840	1895	4185	9393
			TW32	-	428	875	1978	4416	9929
				SC	438	892	2026	4477	10116
				X2	436	891	2010	4480	10057
		R_rfft_i16ci16	-		398	833	1890	4256	9546
		R_rfft_sc_i16ci16			408	841	1918	4240	9575
		R_rfft_x2_i16ci16			403	831	1885	4176	9384
		R_rfft_tw32_i16ci16			412	859	1963	4401	9914
		R_rfft_sc_tw32_i16ci16			422	875	2009	4461	10100
		R_rfft_x2_tw32_i16ci16			425	880	2000	4470	10047
	ci32	R_DSP_FFT_i16ci32	-	-	453	950	2115	4769	10574
				SC	468	962	2167	4802	10767
				X2	454	938	2097	4669	10378
			TW32	-	453	945	2103	4739	10512
				SC	466	954	2152	4769	10702
				X2	448	932	2080	4635	10310
		R_rfft_i16ci32	-		439	936	2102	4756	10561
		R_rfft_sc_i16ci32			453	947	2153	4788	10753
		R_rfft_x2_i16ci32			444	928	2087	4659	10368
		R_rfft_tw32_i16ci32			437	929	2088	4724	10497
		R_rfft_sc_tw32_i16ci32			448	936	2135	4752	10685
		R_rfft_x2_tw32_i16ci32			436	919	2067	4623	10298
i32	ci32	R_DSP_FFT_i32ci32	-	-	439	928	2078	4700	10441
				SC	455	940	2130	4746	10664
				X2	443	923	2063	4602	10245
		R_rfft_i32ci32	-		430	919	2069	4691	10432
		R_rfft_sc_i32ci32			444	929	2120	4736	10654
		R_rfft_x2_i32ci32			434	913	2053	4593	10236
f32	cf32	R_DSP_FFT_f32cf32	-		408	848	1918	4529	10126
		R_rfft_f32cf32	-		404	844	1914	4525	10122

points = 512 - 8192

in	out	Function name	option		points				
			TW32	Scale	512	1024	2048	4096	8192
i16	ci16	R_DSP_FFT_i16ci16	-	-	21129	46607	101459	220625	474325
				SC	20875	46350	99669	218064	464087
				X2	20502	45338	97695	213019	454176
			TW32	-	21854	48355	104872	228645	490026
				SC	22091	49102	105813	231632	493783
				X2	22110	48867	105896	230693	494122
		R_rfft_i16ci16	-		21118	46596	101448	220614	474314
		R_rfft_sc_i16ci16			20862	46337	99656	218051	464074
		R_rfft_x2_i16ci16			20493	45329	97686	213010	454167
		R_rfft_tw32_i16ci16			21839	48340	104857	228630	490011
		R_rfft_sc_tw32_i16ci16			22075	49086	105797	231616	493767
		R_rfft_x2_tw32_i16ci16			22100	48857	105886	230683	494112
	ci32	R_DSP_FFT_i16ci32	-	-	23427	51248	111733	241346	519623
				SC	23540	52001	112166	244339	521336
				X2	22782	49963	108143	234172	501184
			TW32	-	23300	50993	111221	240322	517574
				SC	23411	51744	111653	243314	519287
				X2	22637	49688	107551	233066	498801
		R_rfft_i16ci32	-		23414	51235	111720	241333	519610
		R_rfft_sc_i16ci32			23526	51987	112152	244325	521322
		R_rfft_x2_i16ci32			22772	49953	108133	234162	501174
		R_rfft_tw32_i16ci32			23285	50978	111206	240307	517559
		R_rfft_sc_tw32_i16ci32			23394	51727	111636	243297	519270
		R_rfft_x2_tw32_i16ci32			22625	49676	107539	233054	498789
i32	ci32	R_DSP_FFT_i32ci32	-	-	23166	50731	110704	239293	515522
				SC	23404	51738	111902	243820	521328
				X2	22508	49431	107038	232041	496752
		R_rfft_i32ci32	-		23157	50722	110695	239284	515513
		R_rfft_sc_i32ci32			23394	51728	111892	243810	521318
		R_rfft_x2_i32ci32			22499	49422	107029	232032	496743
f32	cf32	R_DSP_FFT_f32cf32	-		23155	50848	113445	245362	537719
		R_rfft_f32cf32	-		23151	50844	113441	245358	537715

(4) **Complex Conjugate Symmetric IFFT**

points = 16 - 256

in	out	Function name	option		points				
			TW32	Scale	16	32	64	128	256
ci16	i16	R_DSP_IFFT_CCS_ci16i16	-	-	435	891	1985	4418	9849
				SC	452	904	2023	4424	9921
				X2	442	890	1985	4368	9753
			TW32	-	453	917	2051	4550	10189
				SC	467	933	2100	4618	10386
				X2	455	928	2079	4611	10315
		R_irfft_ci16i16	-		421	877	1972	4405	9836
		R_irfft_sc_ci16i16			438	890	2010	4411	9908
		R_irfft_x2_ci16i16			431	879	1975	4358	9743
		R_irfft_tw32_ci16i16			435	899	2034	4533	10172
		R_irfft_sc_tw32_ci16i16			448	914	2082	4600	10368
		R_irfft_x2_tw32_ci16i16			441	914	2066	4598	10302
ci32		R_DSP_IFFT_CCS_ci32i16	-	-	465	946	2094	4643	10301
				SC	474	953	2123	4627	10331
				X2	468	942	2088	4559	10134
			TW32	-	484	972	2159	4765	10613
				SC	491	986	2204	4824	10800
				X2	485	982	2186	4819	10733
		R_irfft_ci32i16	-		450	931	2080	4629	10287
		R_irfft_sc_ci32i16			459	937	2108	4612	10316
		R_irfft_x2_ci32i16			456	930	2077	4548	10123
		R_irfft_tw32_ci32i16			466	954	2142	4748	10596
		R_irfft_sc_tw32_ci32i16			473	968	2187	4807	10783
		R_irfft_x2_tw32_ci32i16			471	968	2173	4806	10720
	i32	R_DSP_IFFT_CCS_ci32i32	-	-	466	978	2166	4881	10797
				SC	480	986	2215	4898	10959
				X2	470	967	2148	4770	10574
		R_irfft_ci32i32	-		458	969	2158	4873	10789
		R_irfft_sc_ci32i32			469	975	2205	4888	10949
		R_irfft_x2_ci32i32			459	956	2138	4760	10564
cf32	f32	R_DSP_IFFT_CCS_cf32f32	-		500	1019	2240	5155	11360
		R_irfft_cf32f32	-		496	1014	2236	5151	11356

points = 512 - 8192

in	out	Function name	option		points				
			TW32	Scale	512	1024	2048	4096	8192
ci16	i16	R_DSP_IFFT_CCS_ci16i16	-	-	21648	47635	103258	224213	480476
				SC	21542	47659	102320	223277	474674
				X2	21262	46867	100952	219605	468186
			TW32	-	22356	49367	106846	232665	497888
				SC	22632	50156	107954	235822	502324
				X2	22609	49877	107867	234711	501981
		R_irfft_ci16i16	-		21635	47622	103245	224200	480463
		R_irfft_sc_ci16i16			21529	47646	102307	223264	474661
		R_irfft_x2_ci16i16			21252	46857	100942	219595	468176
		R_irfft_tw32_ci16i16			22339	49350	106829	232648	497871
		R_irfft_sc_tw32_ci16i16			22614	50138	107936	235804	502306
		R_irfft_x2_tw32_ci16i16			22596	49864	107854	234698	501968
ci32		R_DSP_IFFT_CCS_ci32i16	-	-	22609	49559	107355	232409	497885
				SC	22353	49301	105563	229847	487645
				X2	21965	48272	103511	224722	477401
			TW32	-	23211	51055	110261	239409	511543
				SC	23446	51802	111200	242396	515298
				X2	23441	51543	111195	241369	515293
		R_irfft_ci32i16	-		22595	49545	107341	232395	497871
		R_irfft_sc_ci32i16			22338	49286	105548	229832	487630
		R_irfft_x2_ci32i16			21954	48261	103500	224711	477390
		R_irfft_tw32_ci32i16			23194	51038	110244	239392	511526
		R_irfft_sc_tw32_ci32i16			23429	51785	111183	242379	515281
		R_irfft_x2_tw32_ci32i16			23428	51530	111182	241356	515280
	i32	R_DSP_IFFT_CCS_ci32i32	-	-	23923	52255	113957	245873	529527
				SC	23924	52769	113702	247411	527480
				X2	23172	50736	109686	237250	507336
		R_irfft_ci32i32	-		23915	52247	113949	245865	529519
		R_irfft_sc_ci32i32			23914	52759	113692	247401	527470
		R_irfft_x2_ci32i32			23162	50726	109676	237240	507326
cf32	f32	R_DSP_IFFT_CCS_cf32f32	-		25605	55730	123191	264836	576649
		R_irfft_cf32f32	-		25601	55726	123187	264832	576645

Website and Support

Renesas Electronics Website

http://www.renesas.com/

Inquiries

http://www.renesas.com/contact/

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jan 21, 2019	—	First edition issued

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics Corporation
TOYOSU FORESIA, 3-2-24 Toyosu, Koto-ku, Tokyo 135-0061, Japan

Renesas Electronics America Inc.
1001 Murphy Ranch Road, Milpitas, CA 95035, U.S.A.
Tel: +1-408-432-8888, Fax: +1-408-434-5351

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709 Quantum Plaza, No.27 ZhichunLu, Haidian District, Beijing, 100191 P. R. China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, 200333 P. R. China
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL 2nd Stage, Indiranagar, Bangalore 560 038, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
17F, KAMCO Yangjae Tower, 262, Gangnam-daero, Gangnam-gu, Seoul, 06265 Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5338

Colophon 7.2