RX Family RXv3 CPU Products

RX DSP Library Version 5.0 Additional Information

R01AN4362EJ0100
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 RXv3 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 V3.00.00

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=rxv3	-isa=rxv3	-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=rxv3	-isa=rxv3	-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=rxv3	--isa=rxv3	-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=rxv3	-isa=rxv3	-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=rxv3	-isa=rxv3	-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=rxv3	-isa=rxv3	-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=rxv3	-isa=rxv3	-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=rxv3	-isa=rxv3	-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	570	52	601	52
	i32	int32_t	R_DSP_Mean_i32	573	52	604	52
	f32	float	R_DSP_Mean_f32	352	44	383	44
Minimum value	i16	int16_t	R_DSP_Min_i16	354	36	385	36
	i32	int32_t	R_DSP_Min_i32	345	40	376	40
	f32	float	R_DSP_Min_f32	479	20	508	20
Maximum value	i16	int16_t	R_DSP_Max_i16	354	36	385	36
	i32	int32_t	R_DSP_Max_i32	345	40	376	40
	f32	float	R_DSP_Max_f32	479	20	508	20
Minimum value with index	i16	int16_t	R_DSP_ArgMin_i16	647	48	680	48
	i32	int32_t	R_DSP_ArgMin_i32	577	48	609	48
	f32	float	R_DSP_ArgMin_f32	612	48	644	48
Maximum value with index	i16	int16_t	R_DSP_ArgMax_i16	647	48	680	48
	i32	int32_t	R_DSP_ArgMax_i32	577	48	609	48
	f32	float	R_DSP_ArgMax_f32	612	48	644	48
Mean absolute value and maximum absolute value	i16	int16_t	R_DSP_MeanAbs_i16	394	36	425	36
	i32	int32_t	R_DSP_MeanAbs_i32	572	52	601	52
	f32	float	R_DSP_MeanAbs_f32	650	168	680	168
	i16	int16_t	R_DSP_MaxAbs_i16	414	32	451	32
	i32	int32_t	R_DSP_MaxAbs_i32	521	20	557	20
	f32	float	R_DSP_MaxAbs_f32	847	164	876	164
	i16	int16_t	R_DSP_MeanMaxAbs_i16	565	40	604	40
	i32	int32_t	R_DSP_MeanMaxAbs_i32	1005	168	1039	168
	f32	float	R_DSP_MeanMaxAbs_f32	1107	176	1140	176
Mean value and variance	i16	int16_t	R_DSP_MeanVar_i16	34	72	78	72
	i32	int32_t	R_DSP_MeanVar_i32	34	72	78	72
	f32	float	R_DSP_MeanVar_f32	34	64	78	64
Variance	i16	int16_t	R_DSP_Var_GivenMean_i16	14	12	47	12
	i32	int32_t	R_DSP_Var_GivenMean_i32	14	28	47	28
	f32	float	R_DSP_Var_GivenMean_f32	14	8	47	8
Histogram	i16	uint16_t	R_DSP_Histogram_i16ui16	841	44	996	48
	i32	uint16_t	R_DSP_Histogram_i32ui16	1339	64	1354	68
	f32	uint16_t	R_DSP_Histogram_f32ui16	1181	44	1161	44
Mean value and mean absolute deviation (MAD)	i16	int16_t	R_DSP_MeanMAD_i16	34	72	78	72
	i32	int32_t	R_DSP_MeanMAD_i32	34	72	78	72
	f32	float	R_DSP_MeanMAD_f32	34	64	78	64
Mean absolute deviation (MAD)	i16	int16_t	R_DSP_MAD_GivenMean_i16	14	12	47	12
	i32	int32_t	R_DSP_MAD_GivenMean_i32	14	12	47	12
	f32	float	R_DSP_MAD_GivenMean_f32	14	8	47	8
Median functions	i16	int16_t	R_DSP_Median_InPlace_i16	70	36	109	36
	i32	int32_t	R_DSP_Median_InPlace_i32	76	36	115	36
	f32	float	R_DSP_Median_InPlace_f32	77	36	116	36
	i16	int16_t	R_DSP_Median_i16	162	36	210	36
	i32	int32_t	R_DSP_Median_i32	172	36	217	36
	f32	float	R_DSP_Median_f32	169	36	217	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	69	4	101	4
		R_DSP_FIR_i16i16	346	68	488	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	69	4	101	4
		R_DSP_FIR_i16i32	710	68	861	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	108	4	140	4
		R_DSP_FIR_ci16ci16	2371	76	2844	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	108	4	140	4
		R_DSP_FIR_ci16ci32	1599	76	1842	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	81	4	113	4
		R_DSP_FIR_i32i32	1637	76	1958	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	108	4	140	4
		R_DSP_FIR_ci32ci32	2646	76	3106	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	81	4	113	4
		R_DSP_FIR_f32f32	117	76	211	76
		R_DSP_FIR_f32f32_asm	113	36	113	36
cf32	cf32	R_DSP_FIR_Init_cf32cf32	108	4	140	4
		R_DSP_FIR_cf32cf32	226	92	320	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	17	4	41	4
		R_DSP_IIRBiquad_Init_i16i16	84	4	116	4
		R_DSP_IIRBiquad_i16i16	6479	92	8470	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	17	4	41	4
		R_DSP_IIRBiquad_Init_i16i32	84	4	116	4
		R_DSP_IIRBiquad_i16i32	10353	76	13250	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	17	4	41	4
		R_DSP_IIRBiquad_Init_ci16ci16	123	4	157	4
		R_DSP_IIRBiquad_ci16ci16	11575	108	15300	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	17	4	41	4
		R_DSP_IIRBiquad_Init_ci16ci32	123	4	157	4
		R_DSP_IIRBiquad_ci16ci32	18313	100	23792	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	17	4	41	4
		R_DSP_IIRBiquad_Init_i32i32	96	4	128	4
		R_DSP_IIRBiquad_i32i32	10529	100	14481	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	17	4	41	4
		R_DSP_IIRBiquad_Init_ci32ci32	123	4	157	4
		R_DSP_IIRBiquad_ci32ci32	21481	108	29001	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	17	4	41	4
		R_DSP_IIRBiquad_Init_f32f32	95	4	128	4
		R_DSP_IIRBiquad_f32f32	409	108	511	108
		R_DSP_IIRBiquad_f32f32_asm	405	52	405	52
cf32	cf32	R_DSP_IIRBiquad_StateSize_cf32cf32	17	4	41	4
		R_DSP_IIRBiquad_Init_cf32cf32	122	4	155	4
		R_DSP_IIRBiquad_cf32cf32	969	124	1071	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	556	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	595	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	887	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	209	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	96	88	96
		ci32	R_DSP_DFT_ci16ci32	50	96	88	96
	ci32		R_DSP_DFT_ci32ci32	50	96	88	96
	cf32	cf32	R_DSP_DFT_cf32cf32	50	108	88	108
Complex IDFT	ci16	ci16	R_DSP_IDFT_ci16ci16	50	100	88	100
	ci32		R_DSP_IDFT_ci32ci16	50	100	88	100
		ci32	R_DSP_IDFT_ci32ci32	50	100	88	100
	cf32	cf32	R_DSP_IDFT_cf32cf32	50	112	88	112
Real DFT	i16	ci16	R_DSP_DFT_i16ci16	50	92	91	92
		ci32	R_DSP_DFT_i16ci32	50	92	91	92
	i32		R_DSP_DFT_i32ci32	50	92	91	92
	f32	cf32	R_DSP_DFT_f32cf32	50	108	91	108
Complex Conjugate Symmetry IDFT	ci16	ci16	R_DSP_IDFT_CCS_ci16i16	52	100	93	100
	ci32		R_DSP_IDFT_CCS_ci32i16	52	100	93	100
		ci32	R_DSP_IDFT_CCS_ci32i32	52	100	93	100
	cf32	cf32	R_DSP_IDFT_CCS_cf32f32	52	108	93	108

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	164	12	198	12
	ci32	R_DSP_FFT_BufSize_i16ci32	164	12	198	12
i32	ci32	R_DSP_FFT_BufSize_i32ci32	156	8	190	8
f32	cf32	R_DSP_FFT_BufSize_f32cf32	156	8	175	8
ci16	i16	R_DSP_FFT_BufSize_ci16i16	164	12	198	12
	ci16	R_DSP_FFT_BufSize_ci16ci16	154	8	192	8
	ci32	R_DSP_FFT_BufSize_ci16ci32	154	8	192	8
ci32	i16	R_DSP_FFT_BufSize_ci32i16	164	12	198	12
	i32	R_DSP_FFT_BufSize_ci32i32	same as i32ci32			
	ci16	R_DSP_FFT_BufSize_ci32ci16	154	8	192	8
	ci32	R_DSP_FFT_BufSize_ci32ci32	136	4	174	4
cf32	f32	R_DSP_FFT_BufSize_cf32f32	156	8	175	8
	cf32	R_DSP_FFT_BufSize_cf32cf32	136	4	161	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	566	56	606	56
	ci32	R_DSP_FFT_Init_i16ci32	566	56	606	56
i32	ci32	R_DSP_FFT_Init_i32ci32	380	52	418	52
f32	cf32	R_DSP_FFT_Init_f32cf32	409	76	432	76
ci16	i16	R_DSP_FFT_Init_ci16i16	573	52	613	52
	ci16	R_DSP_FFT_Init_ci16ci16	378	44	411	44
	ci32	R_DSP_FFT_Init_ci16ci32	378	44	411	44
ci32	i16	R_DSP_FFT_Init_ci32i16	573	52	613	52
	i32	R_DSP_FFT_Init_ci32i32	388	52	426	52
	ci16	R_DSP_FFT_Init_ci32ci16	378	44	411	44
	ci32	R_DSP_FFT_Init_ci32ci32	310	44	342	44
cf32	f32	R_DSP_FFT_Init_cf32f32	392	76	415	76
	cf32	R_DSP_FFT_Init_cf32cf32	324	68	343	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	13102	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	10111	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	6521	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	1743	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	191	52	254	52
		i32	R_DSP_VecCplxMag_ci16i32	192	52	255	52
	ci32		R_DSP_VecCplxMag_ci32i32	173	52	236	52
	cf32	f32	R_DSP_VecCplxMag_cf32f32	173	52	236	52
	ci16	i16	R_DSP_VecCplxMag_Fast_ci16i16	191	52	254	52
		i32	R_DSP_VecCplxMag_Fast_ci16i32	192	52	255	52
	ci32		R_DSP_VecCplxMag_Fast_ci32i32	173	52	236	52
	cf32	f32	R_DSP_VecCplxMag_Fast_cf32f32	173	52	236	52
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	191	52	254	52
		i32	R_DSP_VecCplxMagSquared_ci16i32	192	52	255	52
	ci32		R_DSP_VecCplxMagSquared_ci32i32	173	52	236	52
	cf32	f32	R_DSP_VecCplxMagSquared_cf32f32	173	52	236	52
Complex number phase	ci16	i16	R_DSP_CplxPhase_ci16i16	92	28	92	28
	ci32	i32	R_DSP_CplxPhase_ci32i32	91	28	91	28
	cf32	f32	R_DSP_CplxPhase_cf32f32	61	60	61	60
	ci16	i16	R_DSP_VecCplxPhase_ci16i16	338	88	402	88
	ci32	i32	R_DSP_VecCplxPhase_ci32i32	294	88	356	88
	cf32	f32	R_DSP_VecCplxPhase_cf32f32	418	120	479	120
Complex number addition	ci16	ci16	R_DSP_ComplexAdd_ci16ci16	19	4	40	4
	ci32	ci32	R_DSP_ComplexAdd_ci32ci32	22	8	39	8
	cf32	cf32	R_DSP_ComplexAdd_cf32cf32	24	8	31	8
Complex number subtraction	ci16	ci16	R_DSP_ComplexSub_ci16ci16	19	4	40	4
	ci32	ci32	R_DSP_ComplexSub_ci32ci32	22	8	39	8
	cf32	cf32	R_DSP_ComplexSub_cf32cf32	24	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	36	8	83	8
	cf32	cf32	R_DSP_ComplexMul_cf32cf32	40	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	192	52	235	52
	ci32	ci32	R_DSP_VecCplxConjg_ci32ci32	174	52	217	52
	cf32	cf32	R_DSP_VecCplxConjg_cf32cf32	174	52	217	52

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	254	24
		i32	R_DSP_MatrixAdd_i16i32	123	20	203	20
	i32		R_DSP_MatrixAdd_i32i32	125	24	233	24
	f32	f32	R_DSP_MatrixAdd_f32f32	130	24	210	24
	ci16	ci16	R_DSP_MatrixAdd_ci16ci16	202	24	382	24
		ci32	R_DSP_MatrixAdd_ci16ci32	200	20	280	20
	ci32		R_DSP_MatrixAdd_ci32ci32	202	24	338	24
	cf32	cf32	R_DSP_MatrixAdd_cf32cf32	214	24	294	24
Matrix subtraction	i16	i16	R_DSP_MatrixSub_i16i16	125	24	254	24
		i32	R_DSP_MatrixSub_i16i32	123	20	203	20
	i32		R_DSP_MatrixSub_i32i32	125	24	233	24
	f32	f32	R_DSP_MatrixSub_f32f32	130	24	210	24
	ci16	ci16	R_DSP_MatrixSub_ci16ci16	202	24	382	24
		ci32	R_DSP_MatrixSub_ci16ci32	200	20	280	20
	ci32		R_DSP_MatrixSub_ci32ci32	202	24	338	24
	cf32	cf32	R_DSP_MatrixSub_cf32cf32	214	24	294	24
Matrix multiplication	i16	i16	R_DSP_MatrixMul_i16i16	775	76	953	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	1044	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	1402	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	274	76
			R_DSP_MatrixMul_f32f32_asm	182	36	182	36
	ci16	ci16	R_DSP_MatrixMul_ci16ci16	1204	76	1411	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	1557	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	2361	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	466	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	624	40	741	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	680	40	785	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	1275	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	184	48
			R_DSP_MatrixScale_f32f32_asm	124	20	124	20
	ci16	ci16	R_DSP_MatrixScale_ci16ci16	868	40	941	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	880	40	953	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	997	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	268	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 common measurement conditions are

- Device: RX66T Group
- Library R_DSP_FPU_LE.lib
- Code allocation: Code Flash memory
- Data alignment: Internal RAM (Data is allocated to 4-byte boundary alignment sections)
- ROM Cache: ROM cache enabled
- Wait cycle for Code flash memory access: 0 (ICLK=80MHz)

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	489	937	1833	3625	7209	14377
		NEAREST	489	936	1832	3624	7208	14376
	SATURATE	TRUNC	511	983	1927	3815	7591	15143
		NEAREST	518	998	1958	3878	7718	15398
R_DSP_FIR_i16i16_asm_nt	-		478	926	1822	3614	7198	14366
R_DSP_FIR_i16i16_asm_n2			479	926	1822	3614	7198	14366
R_DSP_FIR_i16i16_asm_st			501	973	1917	3805	7581	15133
R_DSP_FIR_i16i16_asm_s2			509	989	1949	3869	7709	15389

Taps=32, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	825	1609	3177	6313	12585	25129
		NEAREST	825	1608	3176	6312	12584	25128
	SATURATE	TRUNC	847	1655	3271	6503	12967	25895
		NEAREST	854	1670	3302	6566	13094	26150
R_DSP_FIR_i16i16_asm_nt	-		814	1598	3166	6302	12574	25118
R_DSP_FIR_i16i16_asm_n2			815	1598	3166	6302	12574	25118
R_DSP_FIR_i16i16_asm_st			837	1645	3261	6493	12957	25885
R_DSP_FIR_i16i16_asm_s2			845	1661	3293	6557	13085	26141

Taps=64, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	1497	2953	5865	11689	23337	46633
		NEAREST	1497	2952	5864	11688	23336	46632
	SATURATE	TRUNC	1519	2999	5959	11879	23719	47399
		NEAREST	1526	3014	5990	11942	23846	47654
R_DSP_FIR_i16i16_asm_nt	-		1486	2942	5854	11678	23326	46622
R_DSP_FIR_i16i16_asm_n2			1487	2942	5854	11678	23326	46622
R_DSP_FIR_i16i16_asm_st			1509	2989	5949	11869	23709	47389
R_DSP_FIR_i16i16_asm_s2			1517	3005	5981	11933	23837	47645

Taps=128, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	2841	5641	11241	22441	44841	89641
		NEAREST	2841	5640	11240	22440	44840	89640
	SATURATE	TRUNC	2863	5687	11335	22631	45223	90407
		NEAREST	2870	5702	11366	22694	45350	90662
R_DSP_FIR_i16i16_asm_nt	-		2830	5630	11230	22430	44830	89630
R_DSP_FIR_i16i16_asm_n2			2831	5630	11230	22430	44830	89630
R_DSP_FIR_i16i16_asm_st			2853	5677	11325	22621	45213	90397
R_DSP_FIR_i16i16_asm_s2			2861	5693	11357	22685	45341	90653

Taps=256, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	5529	11017	21993	43945	87849	175657
		NEAREST	5529	11016	21992	43944	87848	175656
	SATURATE	TRUNC	5551	11063	22087	44135	88231	176423
		NEAREST	5558	11078	22118	44198	88358	176678
R_DSP_FIR_i16i16_asm_nt	-		5518	11006	21982	43934	87838	175646
R_DSP_FIR_i16i16_asm_n2			5519	11006	21982	43934	87838	175646
R_DSP_FIR_i16i16_asm_st			5541	11053	22077	44125	88221	176413
R_DSP_FIR_i16i16_asm_s2			5549	11069	22109	44189	88349	176669

(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)	483	914	1778	3506	6962	13874
			UP (scale=-1)	487	927	1807	3567	7087	14127
			DOWN (scale=1)	498	945	1841	3633	7217	14385
		NEAREST	NO (scale=0)	479	911	1775	3503	6959	13871
			UP (scale=-1)	484	924	1804	3564	7084	14124
			DOWN (scale=1)	495	943	1839	3631	7215	14383
	SATURATE	TRUNC	NO (scale=0)	513	976	1904	3760	7472	14896
			UP (scale=-1)	517	989	1933	3821	7597	15149
			DOWN (scale=1)	511	974	1902	3758	7470	14894
		NEAREST	NO (scale=0)	509	972	1900	3756	7468	14892
			UP (scale=-1)	513	985	1929	3817	7593	15145
			DOWN (scale=1)	508	971	1899	3755	7467	14891
R_DSP_FIR_i16i32_asm_ntn	-		0	460	892	1756	3484	6940	13852
R_DSP_FIR_i16i32_asm_ntu			-1	468	908	1788	3548	7068	14108
R_DSP_FIR_i16i32_asm_ntd			1	476	924	1820	3612	7196	14364
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	477	925	1821	3613	7197	14365
R_DSP_FIR_i16i32_asm_stn			0	494	957	1885	3741	7453	14877
R_DSP_FIR_i16i32_asm_stu			-1	501	973	1917	3805	7581	15133
R_DSP_FIR_i16i32_asm_std			1	493	956	1884	3740	7452	14876
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	494	957	1885	3741	7453	14877

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)	818	1586	3122	6194	12338	24626
			UP (scale=-1)	823	1599	3151	6255	12463	24879
			DOWN (scale=1)	833	1617	3185	6321	12593	25137
		NEAREST	NO (scale=0)	815	1583	3119	6191	12335	24623
			UP (scale=-1)	820	1596	3148	6252	12460	24876
			DOWN (scale=1)	831	1615	3183	6319	12591	25135
	SATURATE	TRUNC	NO (scale=0)	849	1648	3248	6448	12848	25648
			UP (scale=-1)	853	1661	3277	6509	12973	25901
			DOWN (scale=1)	847	1646	3246	6446	12846	25646
		NEAREST	NO (scale=0)	845	1644	3244	6444	12844	25644
			UP (scale=-1)	849	1657	3273	6505	12969	25897
			DOWN (scale=1)	844	1643	3243	6443	12843	25643
R_DSP_FIR_i16i32_asm_ntn	-		0	796	1564	3100	6172	12316	24604
R_DSP_FIR_i16i32_asm_ntu			-1	804	1580	3132	6236	12444	24860
R_DSP_FIR_i16i32_asm_ntd			1	812	1596	3164	6300	12572	25116
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	813	1597	3165	6301	12573	25117
R_DSP_FIR_i16i32_asm_stn			0	830	1629	3229	6429	12829	25629
R_DSP_FIR_i16i32_asm_stu			-1	837	1645	3261	6493	12957	25885
R_DSP_FIR_i16i32_asm_std			1	829	1628	3228	6428	12828	25628
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	830	1629	3229	6429	12829	25629

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)	1490	2930	5810	11570	23090	46130
			UP (scale=-1)	1495	2943	5839	11631	23215	46383
			DOWN (scale=1)	1505	2961	5873	11697	23345	46641
		NEAREST	NO (scale=0)	1487	2927	5807	11567	23087	46127
			UP (scale=-1)	1492	2940	5836	11628	23212	46380
			DOWN (scale=1)	1503	2959	5871	11695	23343	46639
	SATURATE	TRUNC	NO (scale=0)	1521	2992	5936	11824	23600	47152
			UP (scale=-1)	1525	3005	5965	11885	23725	47405
			DOWN (scale=1)	1519	2990	5934	11822	23598	47150
		NEAREST	NO (scale=0)	1517	2988	5932	11820	23596	47148
			UP (scale=-1)	1521	3001	5961	11881	23721	47401
			DOWN (scale=1)	1516	2987	5931	11819	23595	47147
R_DSP_FIR_i16i32_asm_ntn	-		0	1468	2908	5788	11548	23068	46108
R_DSP_FIR_i16i32_asm_ntu			-1	1476	2924	5820	11612	23196	46364
R_DSP_FIR_i16i32_asm_ntd			1	1484	2940	5852	11676	23324	46620
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	1485	2941	5853	11677	23325	46621
R_DSP_FIR_i16i32_asm_stn			0	1502	2973	5917	11805	23581	47133
R_DSP_FIR_i16i32_asm_stu			-1	1509	2989	5949	11869	23709	47389
R_DSP_FIR_i16i32_asm_std			1	1501	2972	5916	11804	23580	47132
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	1502	2973	5917	11805	23581	47133

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)	2834	5618	11186	22322	44594	89138
			UP (scale=-1)	2839	5631	11215	22383	44719	89391
			DOWN (scale=1)	2849	5649	11249	22449	44849	89649
		NEAREST	NO (scale=0)	2831	5615	11183	22319	44591	89135
			UP (scale=-1)	2836	5628	11212	22380	44716	89388
			DOWN (scale=1)	2847	5647	11247	22447	44847	89647
	SATURATE	TRUNC	NO (scale=0)	2865	5680	11312	22576	45104	90160
			UP (scale=-1)	2869	5693	11341	22637	45229	90413
			DOWN (scale=1)	2863	5678	11310	22574	45102	90158
		NEAREST	NO (scale=0)	2861	5676	11308	22572	45100	90156
			UP (scale=-1)	2865	5689	11337	22633	45225	90409
			DOWN (scale=1)	2860	5675	11307	22571	45099	90155
R_DSP_FIR_i16i32_asm_ntn	-		0	2812	5596	11164	22300	44572	89116
R_DSP_FIR_i16i32_asm_ntu			-1	2820	5612	11196	22364	44700	89372
R_DSP_FIR_i16i32_asm_ntd			1	2828	5628	11228	22428	44828	89628
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	2829	5629	11229	22429	44829	89629
R_DSP_FIR_i16i32_asm_stn			0	2846	5661	11293	22557	45085	90141
R_DSP_FIR_i16i32_asm_stu			-1	2853	5677	11325	22621	45213	90397
R_DSP_FIR_i16i32_asm_std			1	2845	5660	11292	22556	45084	90140
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	2846	5661	11293	22557	45085	90141

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)	5522	10994	21938	43826	87602	175154
			UP (scale=-1)	5527	11007	21967	43887	87727	175407
			DOWN (scale=1)	5537	11025	22001	43953	87857	175665
		NEAREST	NO (scale=0)	5519	10991	21935	43823	87599	175151
			UP (scale=-1)	5524	11004	21964	43884	87724	175404
			DOWN (scale=1)	5535	11023	21999	43951	87855	175663
	SATURATE	TRUNC	NO (scale=0)	5553	11056	22064	44080	88112	176176
			UP (scale=-1)	5557	11069	22093	44141	88237	176429
			DOWN (scale=1)	5551	11054	22062	44078	88110	176174
		NEAREST	NO (scale=0)	5549	11052	22060	44076	88108	176172
			UP (scale=-1)	5553	11065	22089	44137	88233	176425
			DOWN (scale=1)	5548	11051	22059	44075	88107	176171
R_DSP_FIR_i16i32_asm_ntn	-		0	5500	10972	21916	43804	87580	175132
R_DSP_FIR_i16i32_asm_ntu			-1	5508	10988	21948	43868	87708	175388
R_DSP_FIR_i16i32_asm_ntd			1	5516	11004	21980	43932	87836	175644
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	5517	11005	21981	43933	87837	175645
R_DSP_FIR_i16i32_asm_stn			0	5534	11037	22045	44061	88093	176157
R_DSP_FIR_i16i32_asm_stu			-1	5541	11053	22077	44125	88221	176413
R_DSP_FIR_i16i32_asm_std			1	5533	11036	22044	44060	88092	176156
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	5534	11037	22045	44061	88093	176157

(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)	535	1015	1975	3895	7735	15415
			UP (scale=30)	552	1047	2039	4023	7991	15927
			DOWN (scale=32)	554	1048	2040	4024	7992	15928
		NEAREST	NO (scale=31)	545	1040	2032	4016	7984	15920
			UP (scale=30)	552	1056	2064	4080	8112	16176
			DOWN (scale=32)	556	1059	2067	4083	8115	16179
	SATURATE	TRUNC	NO (scale=31)	537	1022	1998	3950	7854	15662
			UP (scale=30)	576	1103	2159	4271	8495	16943
			DOWN (scale=32)	570	1089	2129	4209	8369	16689
		NEAREST	NO (scale=31)	533	1019	1995	3947	7851	15659
			UP (scale=30)	579	1115	2187	4331	8619	17195
			DOWN (scale=32)	572	1100	2156	4268	8492	16940
R_DSP_FIR_i32i32_asm_ntn	-		31	509	989	1949	3869	7709	15389
R_DSP_FIR_i32i32_asm_ntu			30	528	1024	2016	4000	7968	15904
R_DSP_FIR_i32i32_asm_ntd			32	527	1023	2015	3999	7967	15903
R_DSP_FIR_i32i32_asm_n2n			31	524	1020	2012	3996	7964	15900
R_DSP_FIR_i32i32_asm_n2u			30	535	1039	2047	4063	8095	16159
R_DSP_FIR_i32i32_asm_n2d			32	536	1040	2048	4064	8096	16160
R_DSP_FIR_i32i32_asm_stn			31	516	1003	1979	3931	7835	15643
R_DSP_FIR_i32i32_asm_stu			30	560	1087	2143	4255	8479	16927
R_DSP_FIR_i32i32_asm_std			32	551	1071	2111	4191	8351	16671
R_DSP_FIR_i32i32_asm_s2n			31	516	1004	1980	3932	7836	15644
R_DSP_FIR_i32i32_asm_s2u			30	567	1103	2175	4319	8607	17183
R_DSP_FIR_i32i32_asm_s2d			32	558	1086	2142	4254	8478	16926

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)	983	1911	3767	7479	14903	29751
			UP (scale=30)	999	1943	3831	7607	15159	30263
			DOWN (scale=32)	1000	1944	3832	7608	15160	30264
		NEAREST	NO (scale=31)	992	1936	3824	7600	15152	30256
			UP (scale=30)	1000	1952	3856	7664	15280	30512
			DOWN (scale=32)	1003	1955	3859	7667	15283	30515
	SATURATE	TRUNC	NO (scale=31)	983	1918	3790	7534	15022	29998
			UP (scale=30)	1024	1999	3951	7855	15663	31279
			DOWN (scale=32)	1017	1985	3921	7793	15537	31025
		NEAREST	NO (scale=31)	979	1915	3787	7531	15019	29995
			UP (scale=30)	1027	2011	3979	7915	15787	31531
			DOWN (scale=32)	1020	1996	3948	7852	15660	31276
R_DSP_FIR_i32i32_asm_ntn	-		31	957	1885	3741	7453	14877	29725
R_DSP_FIR_i32i32_asm_ntu			30	976	1920	3808	7584	15136	30240
R_DSP_FIR_i32i32_asm_ntd			32	975	1919	3807	7583	15135	30239
R_DSP_FIR_i32i32_asm_n2n			31	972	1916	3804	7580	15132	30236
R_DSP_FIR_i32i32_asm_n2u			30	983	1935	3839	7647	15263	30495
R_DSP_FIR_i32i32_asm_n2d			32	984	1936	3840	7648	15264	30496
R_DSP_FIR_i32i32_asm_stn			31	964	1899	3771	7515	15003	29979
R_DSP_FIR_i32i32_asm_stu			30	1008	1983	3935	7839	15647	31263
R_DSP_FIR_i32i32_asm_std			32	999	1967	3903	7775	15519	31007
R_DSP_FIR_i32i32_asm_s2n			31	964	1900	3772	7516	15004	29980
R_DSP_FIR_i32i32_asm_s2u			30	1015	1999	3967	7903	15775	31519
R_DSP_FIR_i32i32_asm_s2d			32	1006	1982	3934	7838	15646	31262

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)	1879	3703	7351	14647	29239	58423
			UP (scale=30)	1895	3735	7415	14775	29495	58935
			DOWN (scale=32)	1896	3736	7416	14776	29496	58936
		NEAREST	NO (scale=31)	1888	3728	7408	14768	29488	58928
			UP (scale=30)	1896	3744	7440	14832	29616	59184
			DOWN (scale=32)	1899	3747	7443	14835	29619	59187
	SATURATE	TRUNC	NO (scale=31)	1879	3710	7374	14702	29358	58670
			UP (scale=30)	1920	3791	7535	15023	29999	59951
			DOWN (scale=32)	1913	3777	7505	14961	29873	59697
		NEAREST	NO (scale=31)	1875	3707	7371	14699	29355	58667
			UP (scale=30)	1923	3803	7563	15083	30123	60203
			DOWN (scale=32)	1916	3788	7532	15020	29996	59948
R_DSP_FIR_i32i32_asm_ntn	-		31	1853	3677	7325	14621	29213	58397
R_DSP_FIR_i32i32_asm_ntu			30	1872	3712	7392	14752	29472	58912
R_DSP_FIR_i32i32_asm_ntd			32	1871	3711	7391	14751	29471	58911
R_DSP_FIR_i32i32_asm_n2n			31	1868	3708	7388	14748	29468	58908
R_DSP_FIR_i32i32_asm_n2u			30	1879	3727	7423	14815	29599	59167
R_DSP_FIR_i32i32_asm_n2d			32	1880	3728	7424	14816	29600	59168
R_DSP_FIR_i32i32_asm_stn			31	1860	3691	7355	14683	29339	58651
R_DSP_FIR_i32i32_asm_stu			30	1904	3775	7519	15007	29983	59935
R_DSP_FIR_i32i32_asm_std			32	1895	3759	7487	14943	29855	59679
R_DSP_FIR_i32i32_asm_s2n			31	1860	3692	7356	14684	29340	58652
R_DSP_FIR_i32i32_asm_s2u			30	1911	3791	7551	15071	30111	60191
R_DSP_FIR_i32i32_asm_s2d			32	1902	3774	7518	15006	29982	59934

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)	3671	7287	14519	28983	57911	115767
			UP (scale=30)	3687	7319	14583	29111	58167	116279
			DOWN (scale=32)	3688	7320	14584	29112	58168	116280
		NEAREST	NO (scale=31)	3680	7312	14576	29104	58160	116272
			UP (scale=30)	3688	7328	14608	29168	58288	116528
			DOWN (scale=32)	3691	7331	14611	29171	58291	116531
	SATURATE	TRUNC	NO (scale=31)	3671	7294	14542	29038	58030	116014
			UP (scale=30)	3712	7375	14703	29359	58671	117295
			DOWN (scale=32)	3705	7361	14673	29297	58545	117041
		NEAREST	NO (scale=31)	3667	7291	14539	29035	58027	116011
			UP (scale=30)	3715	7387	14731	29419	58795	117547
			DOWN (scale=32)	3708	7372	14700	29356	58668	117292
R_DSP_FIR_i32i32_asm_ntn	-		31	3645	7261	14493	28957	57885	115741
R_DSP_FIR_i32i32_asm_ntu			30	3664	7296	14560	29088	58144	116256
R_DSP_FIR_i32i32_asm_ntd			32	3663	7295	14559	29087	58143	116255
R_DSP_FIR_i32i32_asm_n2n			31	3660	7292	14556	29084	58140	116252
R_DSP_FIR_i32i32_asm_n2u			30	3671	7311	14591	29151	58271	116511
R_DSP_FIR_i32i32_asm_n2d			32	3672	7312	14592	29152	58272	116512
R_DSP_FIR_i32i32_asm_stn			31	3652	7275	14523	29019	58011	115995
R_DSP_FIR_i32i32_asm_stu			30	3696	7359	14687	29343	58655	117279
R_DSP_FIR_i32i32_asm_std			32	3687	7343	14655	29279	58527	117023
R_DSP_FIR_i32i32_asm_s2n			31	3652	7276	14524	29020	58012	115996
R_DSP_FIR_i32i32_asm_s2u			30	3703	7375	14719	29407	58783	117535
R_DSP_FIR_i32i32_asm_s2d			32	3694	7358	14686	29342	58654	117278

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)	7255	14455	28855	57655	115255	230455
			UP (scale=30)	7271	14487	28919	57783	115511	230967
			DOWN (scale=32)	7272	14488	28920	57784	115512	230968
		NEAREST	NO (scale=31)	7264	14480	28912	57776	115504	230960
			UP (scale=30)	7272	14496	28944	57840	115632	231216
			DOWN (scale=32)	7275	14499	28947	57843	115635	231219
	SATURATE	TRUNC	NO (scale=31)	7255	14462	28878	57710	115374	230702
			UP (scale=30)	7296	14543	29039	58031	116015	231983
			DOWN (scale=32)	7289	14529	29009	57969	115889	231729
		NEAREST	NO (scale=31)	7251	14459	28875	57707	115371	230699
			UP (scale=30)	7299	14555	29067	58091	116139	232235
			DOWN (scale=32)	7292	14540	29036	58028	116012	231980
R_DSP_FIR_i32i32_asm_ntn	-		31	7229	14429	28829	57629	115229	230429
R_DSP_FIR_i32i32_asm_ntu			30	7248	14464	28896	57760	115488	230944
R_DSP_FIR_i32i32_asm_ntd			32	7247	14463	28895	57759	115487	230943
R_DSP_FIR_i32i32_asm_n2n			31	7244	14460	28892	57756	115484	230940
R_DSP_FIR_i32i32_asm_n2u			30	7255	14479	28927	57823	115615	231199
R_DSP_FIR_i32i32_asm_n2d			32	7256	14480	28928	57824	115616	231200
R_DSP_FIR_i32i32_asm_stn			31	7236	14443	28859	57691	115355	230683
R_DSP_FIR_i32i32_asm_stu			30	7280	14527	29023	58015	115999	231967
R_DSP_FIR_i32i32_asm_std			32	7271	14511	28991	57951	115871	231711
R_DSP_FIR_i32i32_asm_s2n			31	7236	14444	28860	57692	115356	230684
R_DSP_FIR_i32i32_asm_s2u			30	7287	14543	29055	58079	116127	232223
R_DSP_FIR_i32i32_asm_s2d			32	7278	14526	29022	58014	115998	231966

(4) **R_DSP_FIR_f32f32**

Scale=1.0f

Function name	Taps	Samples					
		8	16	32	64	128	256
R_DSP_FIR_f32f32	16	834	1634	3234	6434	12834	25634
R_DSP_FIR_f32f32_asm		831	1631	3231	6431	12831	25631
R_DSP_FIR_f32f32	32	1602	3170	6306	12578	25122	50210
R_DSP_FIR_f32f32_asm		1599	3167	6303	12575	25119	50207
R_DSP_FIR_f32f32	64	3138	6242	12450	24866	49698	99362
R_DSP_FIR_f32f32_asm		3135	6239	12447	24863	49695	99359
R_DSP_FIR_f32f32	128	6210	12386	24738	49442	98850	197666
R_DSP_FIR_f32f32_asm		6207	12383	24735	49439	98847	197663
R_DSP_FIR_f32f32	256	12354	24674	49314	98594	197154	394274
R_DSP_FIR_f32f32_asm		12351	24671	49311	98591	197151	394271

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	82	90	108	144	216	360	648	1224	2376
			13	90	103	131	187	299	523	971	1867	3659
		NEAREST	14	78	89	111	155	243	419	771	1475	2883
			13	87	104	138	206	342	614	1158	2246	4422
	SATURATE	TRUNC	14	77	87	107	147	227	387	707	1347	2627
			13	89	106	144	220	372	676	1284	2500	4932
		NEAREST	14	77	86	106	146	226	386	706	1346	2626
			13	86	107	147	227	387	707	1347	2627	5187
R_DSP_IIRBiquad_i16i16_asm_nt1n	-		14	57	68	86	122	194	338	626	1202	2354
R_DSP_IIRBiquad_i16i16_asm_nt1u			13	64	80	108	164	276	500	948	1844	3636
R_DSP_IIRBiquad_i16i16_asm_nt1d			same as nt1u									
R_DSP_IIRBiquad_i16i16_asm_n21n			14	60	72	94	138	226	402	754	1458	2866
R_DSP_IIRBiquad_i16i16_asm_n21u			13	67	86	120	188	324	596	1140	2228	4404
R_DSP_IIRBiquad_i16i16_asm_n21d			same as n21u									
R_DSP_IIRBiquad_i16i16_asm_st1n			14	59	70	90	130	210	370	690	1330	2610
R_DSP_IIRBiquad_i16i16_asm_st1u			13	69	88	126	202	354	658	1266	2482	4914
R_DSP_IIRBiquad_i16i16_asm_st1d			same as st1u									
R_DSP_IIRBiquad_i16i16_asm_s21n			14	61	72	92	132	212	372	692	1332	2612
R_DSP_IIRBiquad_i16i16_asm_s21u			13	70	92	132	212	372	692	1332	2612	5172
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	96	116	150	218	354	626	1170	2258	4434
			13	105	130	174	262	438	790	1494	2902	5718
		NEAREST	14	97	122	166	254	430	782	1486	2894	5710
			13	106	137	193	305	529	977	1873	3665	7249
	SATURATE	TRUNC	14	95	116	154	230	382	686	1294	2510	4942
			13	106	135	191	303	527	975	1871	3663	7247
		NEAREST	14	94	115	153	229	381	685	1293	2509	4941
			13	104	136	194	310	542	1006	1934	3790	7502
R_DSP_IIRBiquad_i16i16_asm_nt1n	-		14	74	94	128	196	332	604	1148	2236	4412
R_DSP_IIRBiquad_i16i16_asm_nt1u			13	82	107	151	239	415	767	1471	2879	5695
R_DSP_IIRBiquad_i16i16_asm_nt1d			same as nt1u									
R_DSP_IIRBiquad_i16i16_asm_n21n			14	80	105	149	237	413	765	1469	2877	5693
R_DSP_IIRBiquad_i16i16_asm_n21u			13	88	119	175	287	511	959	1855	3647	7231
R_DSP_IIRBiquad_i16i16_asm_n21d			same as n21u									
R_DSP_IIRBiquad_i16i16_asm_st1n			14	78	99	137	213	365	669	1277	2493	4925
R_DSP_IIRBiquad_i16i16_asm_st1u			13	88	117	173	285	509	957	1853	3645	7229
R_DSP_IIRBiquad_i16i16_asm_st1d			same as st1u									
R_DSP_IIRBiquad_i16i16_asm_s21n			14	80	101	139	215	367	671	1279	2495	4927
R_DSP_IIRBiquad_i16i16_asm_s21u			13	89	121	179	295	527	991	1919	3775	7487
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	177	243	375	639	1167	2223	4335	8559
			13	149	190	266	418	722	1330	2546	4978	9842
		NEAREST	14	147	194	282	458	810	1514	2922	5738	11370
			13	156	209	309	509	909	1709	3309	6509	12909
	SATURATE	TRUNC	14	142	181	255	403	699	1291	2475	4843	9579
			13	152	199	291	475	843	1579	3051	5995	11883
		NEAREST	14	140	179	253	401	697	1289	2473	4841	9577
			13	150	200	294	482	858	1610	3114	6122	12138
R_DSP_IIRBiquad_i16i16_asm_nt1n	-		14	119	155	221	353	617	1145	2201	4313	8537
R_DSP_IIRBiquad_i16i16_asm_nt1u			13	126	167	243	395	699	1307	2523	4955	9819
R_DSP_IIRBiquad_i16i16_asm_nt1d			same as nt1u									
R_DSP_IIRBiquad_i16i16_asm_n21n			14	130	177	265	441	793	1497	2905	5721	11353
R_DSP_IIRBiquad_i16i16_asm_n21u			13	138	191	291	491	891	1691	3291	6491	12891
R_DSP_IIRBiquad_i16i16_asm_n21d			same as n21u									
R_DSP_IIRBiquad_i16i16_asm_st1n			14	125	164	238	386	682	1274	2458	4826	9562
R_DSP_IIRBiquad_i16i16_asm_st1u			13	134	181	273	457	825	1561	3033	5977	11865
R_DSP_IIRBiquad_i16i16_asm_st1d			same as st1u									
R_DSP_IIRBiquad_i16i16_asm_s21n			14	126	165	239	387	683	1275	2459	4827	9563
R_DSP_IIRBiquad_i16i16_asm_s21u			13	135	185	279	467	843	1595	3099	6107	12123
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	85	94	120	166	258	442	810	1546	3018
			-1	93	105	133	187	295	511	943	1807	3535
			1	94	110	146	212	344	608	1136	2192	4304
		NEAREST	0	81	94	123	177	285	501	933	1797	3525
			-1	89	103	136	198	322	570	1066	2058	4042
			1	93	112	152	230	386	698	1322	2570	5066
	SATURATE	TRUNC	0	85	95	129	195	327	591	1119	2175	4287
			-1	90	104	142	216	364	660	1252	2436	4804
			1	92	109	151	233	397	725	1381	2693	5317
		NEAREST	0	80	94	128	194	326	590	1118	2174	4286
			-1	87	103	141	215	363	659	1251	2435	4803
			1	92	109	154	240	412	756	1444	2820	5572
R_DSP_IIRBiquad_i16i32_asm_nt1n	-		0	64	75	101	147	239	423	791	1527	2999
R_DSP_IIRBiquad_i16i32_asm_nt1u			-1	71	83	111	165	273	489	921	1785	3513
R_DSP_IIRBiquad_i16i32_asm_nt1d			1	71	87	123	189	321	585	1113	2169	4281
R_DSP_IIRBiquad_i16i32_asm_n21n			0	63	77	106	160	268	484	916	1780	3508
R_DSP_IIRBiquad_i16i32_asm_n21u			-1	69	84	117	179	303	551	1047	2039	4023
R_DSP_IIRBiquad_i16i32_asm_n21d			1	73	92	132	210	366	678	1302	2550	5046
R_DSP_IIRBiquad_i16i32_asm_st1n			0	63	79	113	179	311	575	1103	2159	4271
R_DSP_IIRBiquad_i16i32_asm_st1u			-1	69	86	124	198	346	642	1234	2418	4786
R_DSP_IIRBiquad_i16i32_asm_st1d			1	71	90	132	214	378	706	1362	2674	5298
R_DSP_IIRBiquad_i16i32_asm_s21n			0	65	82	116	182	314	578	1106	2162	4274
R_DSP_IIRBiquad_i16i32_asm_s21u			-1	73	89	127	201	349	645	1237	2421	4789
R_DSP_IIRBiquad_i16i32_asm_s21d			1	74	94	139	225	397	741	1429	2805	5557

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	102	121	163	241	397	709	1333	2581	5077
			-1	109	131	175	261	433	777	1465	2841	5593
			1	111	136	188	286	482	874	1658	3226	6362
		NEAREST	0	100	126	177	275	471	863	1647	3215	6351
			-1	109	136	191	297	509	933	1781	3477	6869
			1	113	145	207	329	573	1061	2037	3989	7893
	SATURATE	TRUNC	0	99	125	177	279	483	891	1707	3339	6603
			-1	105	132	188	298	518	958	1838	3598	7118
			1	109	138	198	316	552	1024	1968	3856	7632
		NEAREST	0	95	122	174	276	480	888	1704	3336	6600
			-1	105	132	188	298	518	958	1838	3598	7118
			1	108	137	200	322	566	1054	2030	3982	7886
R_DSP_IIRBiquad_i16i32_asm_nt1n	-		0	82	102	144	222	378	690	1314	2562	5058
R_DSP_IIRBiquad_i16i32_asm_nt1u			-1	87	109	153	239	411	755	1443	2819	5571
R_DSP_IIRBiquad_i16i32_asm_nt1d			1	88	113	165	263	459	851	1635	3203	6339
R_DSP_IIRBiquad_i16i32_asm_n21n			0	83	109	160	258	454	846	1630	3198	6334
R_DSP_IIRBiquad_i16i32_asm_n21u			-1	90	117	172	278	490	914	1762	3458	6850
R_DSP_IIRBiquad_i16i32_asm_n21d			1	93	125	187	309	553	1041	2017	3969	7873
R_DSP_IIRBiquad_i16i32_asm_st1n			0	82	109	161	263	467	875	1691	3323	6587
R_DSP_IIRBiquad_i16i32_asm_st1u			-1	87	114	170	280	500	940	1820	3580	7100
R_DSP_IIRBiquad_i16i32_asm_st1d			1	90	119	179	297	533	1005	1949	3837	7613
R_DSP_IIRBiquad_i16i32_asm_s21n			0	83	110	162	264	468	876	1692	3324	6588
R_DSP_IIRBiquad_i16i32_asm_s21u			-1	91	118	174	284	504	944	1824	3584	7104
R_DSP_IIRBiquad_i16i32_asm_s21d			1	92	122	185	307	551	1039	2015	3967	7871

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	144	179	253	395	679	1247	2383	4655	9199
			-1	151	189	265	415	715	1315	2515	4915	9715
			1	153	194	278	440	764	1412	2708	5300	10484
		NEAREST	0	148	196	291	477	849	1593	3081	6057	12009
			-1	157	206	305	499	887	1663	3215	6319	12527
			1	161	215	321	531	951	1791	3471	6831	13551
	SATURATE	TRUNC	0	145	188	276	450	798	1494	2886	5670	11238
			-1	149	194	286	468	832	1560	3016	5928	11752
			1	153	200	296	486	866	1626	3146	6186	12266
		NEAREST	0	140	185	273	447	795	1491	2883	5667	11235
			-1	149	194	286	468	832	1560	3016	5928	11752
			1	152	199	298	492	880	1656	3208	6312	12520
R_DSP_IIRBiquad_i16i32_asm_nt1n	-		0	124	160	234	376	660	1228	2364	4636	9180
R_DSP_IIRBiquad_i16i32_asm_nt1u			-1	129	167	243	393	693	1293	2493	4893	9693
R_DSP_IIRBiquad_i16i32_asm_nt1d			1	130	171	255	417	741	1389	2685	5277	10461
R_DSP_IIRBiquad_i16i32_asm_n21n			0	131	179	274	460	832	1576	3064	6040	11992
R_DSP_IIRBiquad_i16i32_asm_n21u			-1	138	187	286	480	868	1644	3196	6300	12508
R_DSP_IIRBiquad_i16i32_asm_n21d			1	141	195	301	511	931	1771	3451	6811	13531
R_DSP_IIRBiquad_i16i32_asm_st1n			0	127	172	260	434	782	1478	2870	5654	11222
R_DSP_IIRBiquad_i16i32_asm_st1u			-1	131	176	268	450	814	1542	2998	5910	11734
R_DSP_IIRBiquad_i16i32_asm_st1d			1	134	181	277	467	847	1607	3127	6167	12247
R_DSP_IIRBiquad_i16i32_asm_s21n			0	128	173	261	435	783	1479	2871	5655	11223
R_DSP_IIRBiquad_i16i32_asm_s21u			-1	135	180	272	454	818	1546	3002	5914	11738
R_DSP_IIRBiquad_i16i32_asm_s21d			1	136	184	283	477	865	1641	3193	6297	12505

(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	81	87	105	141	213	357	645	1221	2373
			29	88	101	127	179	283	491	907	1739	3403
			31	86	98	122	170	266	458	842	1610	3146
		NEAREST	30	78	89	111	155	243	419	771	1475	2883
			29	86	102	134	198	326	582	1094	2118	4166
			31	89	104	134	194	314	554	1034	1994	3914
	SATURATE	TRUNC	30	76	87	107	147	227	387	707	1347	2627
			29	92	114	156	240	408	744	1416	2758	5434
			31	91	108	140	204	332	588	1100	2124	4172
		NEAREST	30	74	85	105	145	225	385	705	1345	2625
			29	91	112	156	240	408	744	1417	2764	5446
			31	88	105	139	207	343	615	1159	2247	4423
R_DSP_IIRBiquad_i32i32_asm_nt1n	-		30	58	69	87	123	195	339	627	1203	2355
R_DSP_IIRBiquad_i32i32_asm_nt1u			29	66	81	107	159	263	471	887	1719	3383
R_DSP_IIRBiquad_i32i32_asm_nt1d			31	63	77	101	149	245	437	821	1589	3125
R_DSP_IIRBiquad_i32i32_asm_n21n			30	62	74	96	140	228	404	756	1460	2868
R_DSP_IIRBiquad_i32i32_asm_n21u			29	67	85	117	181	309	565	1077	2101	4149
R_DSP_IIRBiquad_i32i32_asm_n21d			31	68	85	115	175	295	535	1015	1975	3895
R_DSP_IIRBiquad_i32i32_asm_st1n			30	60	72	92	132	212	372	692	1332	2612
R_DSP_IIRBiquad_i32i32_asm_st1u			29	74	96	138	222	390	726	1398	2740	5416
R_DSP_IIRBiquad_i32i32_asm_st1d			31	71	89	121	185	313	569	1081	2105	4153
R_DSP_IIRBiquad_i32i32_asm_s21n			30	61	72	92	132	212	372	692	1332	2612
R_DSP_IIRBiquad_i32i32_asm_s21u			29	76	97	141	225	393	729	1402	2749	5431
R_DSP_IIRBiquad_i32i32_asm_s21d			31	71	88	122	190	326	598	1142	2230	4406

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	92	112	146	214	350	622	1166	2254	4430
			29	104	128	170	254	422	758	1430	2774	5462
			31	103	126	166	246	406	726	1366	2646	5206
		NEAREST	30	95	119	163	251	427	779	1483	2891	5707
			29	105	136	190	298	514	946	1810	3538	6994
			31	107	136	188	292	500	916	1748	3412	6740
	SATURATE	TRUNC	30	93	115	153	229	381	685	1293	2509	4941
			29	111	144	204	324	564	1044	2004	3918	7738
			31	110	138	188	288	488	888	1688	3288	6488
		NEAREST	30	93	115	153	229	381	685	1293	2509	4941
			29	109	143	205	329	569	1049	2009	3930	7763
			31	106	134	186	290	498	914	1746	3410	6738
R_DSP_IIRBiquad_i32i32_asm_nt1n	-		30	74	94	128	196	332	604	1148	2236	4412
R_DSP_IIRBiquad_i32i32_asm_nt1u			29	84	108	150	234	402	738	1410	2754	5442
R_DSP_IIRBiquad_i32i32_asm_nt1d			31	82	105	145	225	385	705	1345	2625	5185
R_DSP_IIRBiquad_i32i32_asm_n21n			30	80	104	148	236	412	764	1468	2876	5692
R_DSP_IIRBiquad_i32i32_asm_n21u			29	88	119	173	281	497	929	1793	3521	6977
R_DSP_IIRBiquad_i32i32_asm_n21d			31	88	117	169	273	481	897	1729	3393	6721
R_DSP_IIRBiquad_i32i32_asm_st1n			30	78	100	138	214	366	670	1278	2494	4926
R_DSP_IIRBiquad_i32i32_asm_st1u			29	93	126	186	306	546	1026	1986	3900	7720
R_DSP_IIRBiquad_i32i32_asm_st1d			31	91	119	169	269	469	869	1669	3269	6469
R_DSP_IIRBiquad_i32i32_asm_s21n			30	80	102	140	216	368	672	1280	2496	4928
R_DSP_IIRBiquad_i32i32_asm_s21u			29	94	128	190	314	554	1034	1994	3915	7748
R_DSP_IIRBiquad_i32i32_asm_s21d			31	89	117	169	273	481	897	1729	3393	6721

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	140	176	242	374	638	1166	2222	4334	8558
			29	152	192	266	414	710	1302	2486	4854	9590
			31	150	189	261	405	693	1269	2421	4725	9333
		NEAREST	30	146	192	280	456	808	1512	2920	5736	11368
			29	156	209	307	503	895	1679	3247	6383	12655
			31	158	209	305	497	881	1649	3185	6257	12401
	SATURATE	TRUNC	30	142	182	256	404	700	1292	2476	4844	9580
			29	160	211	307	499	883	1651	3187	6253	12381
			31	160	206	292	464	808	1496	2872	5624	11128
		NEAREST	30	142	182	256	404	700	1292	2476	4844	9580
			29	158	210	308	504	895	1663	3199	6271	12414
			31	156	201	289	465	817	1521	2929	5745	11377
R_DSP_IIRBiquad_i32i32_asm_nt1n	-		30	122	158	224	356	620	1148	2204	4316	8540
R_DSP_IIRBiquad_i32i32_asm_nt1u			29	132	172	246	394	690	1282	2466	4834	9570
R_DSP_IIRBiquad_i32i32_asm_nt1d			31	129	168	240	384	672	1248	2400	4704	9312
R_DSP_IIRBiquad_i32i32_asm_n21n			30	131	177	265	441	793	1497	2905	5721	11353
R_DSP_IIRBiquad_i32i32_asm_n21u			29	139	192	290	486	878	1662	3230	6366	12638
R_DSP_IIRBiquad_i32i32_asm_n21d			31	139	190	286	478	862	1630	3166	6238	12382
R_DSP_IIRBiquad_i32i32_asm_st1n			30	127	167	241	389	685	1277	2461	4829	9565
R_DSP_IIRBiquad_i32i32_asm_st1u			29	142	193	289	481	865	1633	3169	6235	12363
R_DSP_IIRBiquad_i32i32_asm_st1d			31	141	187	273	445	789	1477	2853	5605	11109
R_DSP_IIRBiquad_i32i32_asm_s21n			30	129	169	243	391	687	1279	2463	4831	9567
R_DSP_IIRBiquad_i32i32_asm_s21u			29	143	195	293	489	880	1648	3184	6256	12399
R_DSP_IIRBiquad_i32i32_asm_s21d			31	139	184	272	448	800	1504	2912	5728	11360

(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	69	83	113	173	293	533	1013	1973	3893
R_DSP_IIRBiquad_f32f32_asm		64	80	110	170	290	530	1010	1970	3890
R_DSP_IIRBiquad_f32f32	2	89	121	179	295	527	991	1919	3775	7487
R_DSP_IIRBiquad_f32f32_asm		86	118	176	292	524	988	1916	3772	7484
R_DSP_IIRBiquad_f32f32	4	143	203	317	545	1001	1913	3737	7385	14681
R_DSP_IIRBiquad_f32f32_asm		140	200	314	542	998	1910	3734	7382	14678

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	-	-	539	1267	2979	6943	15861
				SC	549	1296	2971	6998	15709
				X2	539	1265	2913	6811	15347
			TW32	-	558	1315	3074	7214	16420
				SC	576	1364	3142	7411	16680
				X2	569	1347	3136	7340	16674
		R_cfft_ci16ci16	-		522	1251	2965	6929	15847
		R_cfft_sc_ci16ci16			529	1281	2957	6984	15695
		R_cfft_x2_ci16ci16			527	1255	2903	6801	15337
		R_cfft_tw32_ci16ci16			534	1295	3055	7195	16401
		R_cfft_sc_tw32_ci16ci16			552	1345	3123	7392	16661
		R_cfft_x2_tw32_ci16ci16			555	1332	3122	7326	16660
	ci32	R_DSP_FFT_ci16ci32	-	-	621	1433	3396	7796	17881
				SC	628	1478	3429	7988	18011
				X2	612	1412	3296	7600	17238
			TW32	-	629	1441	3398	7798	17868
				SC	636	1491	3430	7991	17996
				X2	616	1425	3302	7605	17228
		R_cfft_ci16ci32	-		606	1419	3384	7784	17869
		R_cfft_sc_ci16ci32			609	1464	3415	7974	17997
		R_cfft_x2_ci16ci32			593	1399	3285	7589	17227
		R_cfft_tw32_ci16ci32			606	1423	3381	7781	17851
		R_cfft_sc_tw32_ci16ci32			612	1469	3411	7972	17977
		R_cfft_x2_tw32_ci16ci32			600	1406	3286	7589	17212
ci32	ci32	R_DSP_FFT_ci32ci32	-	-	610	1414	3355	7723	17729
				SC	620	1463	3392	7919	17862
				X2	604	1401	3265	7538	17095
		R_cfft_ci32ci32	-		596	1401	3345	7713	17719
		R_cfft_sc_ci32ci32			606	1451	3381	7908	17851
		R_cfft_x2_ci32ci32			588	1387	3253	7526	17083
cf32	cf32	R_DSP_FFT_cf32cf32	-		636	1482	3663	8383	19685
		R_cfft_cf32cf32	-		627	1477	3659	8379	19681

points = 512 – 2048

in	out	Function name	option		points		
			TW32	Scale	512	1024	2048
ci16	ci16	R_DSP_FFT_ci16ci16	-	-	35969	80071	177491
				SC	35944	78959	176378
				X2	34941	76997	171343
			TW32	-	37424	83062	184834
				SC	38197	84090	187911
				X2	37934	84084	186880
		R_cfft_ci16ci16	-		35955	80057	177477
		R_cfft_sc_ci16ci16			35930	78945	176364
		R_cfft_x2_ci16ci16			34931	76987	171333
		R_cfft_tw32_ci16ci16			37405	83043	184815
		R_cfft_sc_tw32_ci16ci16			38178	84071	187892
		R_cfft_x2_tw32_ci16ci16			37920	84070	186866
	ci32	R_DSP_FFT_ci16ci32	-	-	40077	89490	196590
				SC	40846	90005	199664
				X2	38794	85904	189420
			TW32	-	40064	89414	196514
				SC	40833	89926	199587
				X2	38783	85830	189345
		R_cfft_ci16ci32	-		40065	89478	196578
		R_cfft_sc_ci16ci32			40832	89991	199650
		R_cfft_x2_ci16ci32			38783	85893	189409
		R_cfft_tw32_ci16ci32			40047	89397	196497
		R_cfft_sc_tw32_ci16ci32			40814	89907	199568
		R_cfft_x2_tw32_ci16ci32			38767	85814	189329
ci32	ci32	R_DSP_FFT_ci32ci32	-	-	39797	88891	195479
				SC	40569	89408	198555
				X2	38524	85313	188318
		R_cfft_ci32ci32	-		39787	88881	195469
		R_cfft_sc_ci32ci32			40558	89397	198544
		R_cfft_x2_ci32ci32			38512	85301	188306
cf32	cf32	R_DSP_FFT_cf32cf32	-		43865	99551	217403
		R_cfft_cf32cf32	-		43861	99547	217399

(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	-	-	648	1478	3396	7774	17510
				SC	652	1510	3394	7840	17380
				X2	646	1476	3330	7644	16996
			TW32	-	661	1527	3491	8048	18085
				SC	678	1574	3562	8245	18364
				X2	674	1555	3555	8175	18341
		R_icfft_ci16ci16	-		637	1467	3385	7763	17499
		R_icfft_sc_ci16ci16			637	1495	3380	7826	17366
		R_icfft_x2_ci16ci16			636	1466	3320	7634	16986
		R_icfft_tw32_ci16ci16			643	1511	3476	8033	18070
		R_icfft_sc_tw32_ci16ci16			661	1556	3545	8228	18347
		R_icfft_x2_tw32_ci16ci16			660	1540	3541	8161	18327
ci32		R_DSP_IFFT_ci32ci16	-	-	700	1582	3603	8188	18341
				SC	712	1618	3608	8261	18234
				X2	700	1580	3533	8058	17823
			TW32	-	718	1630	3703	8467	18921
				SC	732	1681	3773	8663	19199
				X2	730	1657	3762	8591	19187
		R_icfft_ci32ci16	-		690	1572	3593	8178	18331
		R_icfft_sc_ci32ci16			698	1608	3598	8251	18224
		R_icfft_x2_ci32ci16			692	1572	3525	8050	17815
		R_icfft_tw32_ci32ci16			704	1615	3689	8453	18907
		R_icfft_sc_tw32_ci32ci16			716	1662	3757	8647	19183
		R_icfft_x2_tw32_ci32ci16			718	1644	3750	8579	19175
	ci32	R_DSP_IFFT_ci32ci32	-	-	716	1631	3784	8569	19421
				SC	724	1676	3825	8787	19655
				X2	712	1614	3688	8377	18782
		R_icfft_ci32ci32	-		708	1619	3775	8560	19412
		R_icfft_sc_ci32ci32			713	1666	3816	8778	19646
		R_icfft_x2_ci32ci32			699	1602	3677	8366	18771
cf32	cf32	R_DSP_IFFT_cf32cf32	-		777	1750	4171	9371	21633
		R_icfft_cf32cf32	-		774	1744	4168	9368	21630

points = 512 - 2048

in	out	Function name	option		points		
			TW32	Scale	512	1024	2048
ci16	ci16	R_DSP_IFFT_ci16ci16	-	-	39280	86648	190722
				SC	39282	85622	189700
				X2	38254	83574	184576
			TW32	-	40754	89719	198148
				SC	41543	90830	201305
				X2	41265	90743	200195
		R_icfft_ci16ci16	-		39269	86637	190711
		R_icfft_sc_ci16ci16			39268	85608	189686
		R_icfft_x2_ci16ci16			38244	83564	184566
		R_icfft_tw32_ci16ci16			40739	89704	198133
		R_icfft_sc_tw32_ci16ci16			41526	90813	201288
		R_icfft_x2_tw32_ci16ci16			41251	90729	200181
ci32		R_DSP_IFFT_ci32ci16	-	-	40942	89975	197376
				SC	40967	89036	196441
				X2	39916	86897	191230
			TW32	-	42421	93051	204807
				SC	43209	94161	207963
				X2	42944	94148	206929
		R_icfft_ci32ci16	-		40932	89965	197366
		R_icfft_sc_ci32ci16			40957	89026	196431
		R_icfft_x2_ci32ci16			39908	86889	191222
		R_icfft_tw32_ci32ci16			42407	93037	204793
		R_icfft_sc_tw32_ci32ci16			43193	94145	207947
		R_icfft_x2_tw32_ci32ci16			42932	94136	206917
	ci32	R_DSP_IFFT_ci32ci32	-	-	43154	95638	208883
				SC	44157	96833	213407
				X2	41875	92056	201717
		R_icfft_ci32ci32	-		43145	95629	208874
		R_icfft_sc_ci32ci32			44148	96824	213398
		R_icfft_x2_ci32ci32			41864	92045	201706
cf32	cf32	R_DSP_IFFT_cf32cf32	-		47733	107259	232791
		R_icfft_cf32cf32	-		47730	107256	232788

(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	-	-	410	837	1866	4184	9361
				SC	423	843	1899	4182	9426
				X2	416	833	1863	4118	9232
			TW32	-	429	872	1950	4351	9768
				SC	442	885	1996	4415	9962
				X2	434	879	1974	4408	9890
		R_rfft_i16ci16	-		393	821	1853	4171	9348
		R_rfft_sc_i16ci16			406	829	1886	4169	9413
		R_rfft_x2_i16ci16			400	821	1851	4107	9221
		R_rfft_tw32_i16ci16			409	853	1932	4333	9750
		R_rfft_sc_tw32_i16ci16			420	866	1978	4397	9944
		R_rfft_x2_tw32_i16ci16			415	861	1956	4391	9873
	ci32	R_DSP_FFT_i16ci32	-	-	445	934	2078	4692	10404
				SC	456	943	2124	4725	10595
				X2	448	922	2057	4592	10208
			TW32	-	447	931	2066	4664	10345
				SC	461	941	2117	4701	10541
				X2	455	924	2051	4572	10157
		R_rfft_i16ci32	-		432	924	2068	4682	10394
		R_rfft_sc_i16ci32			443	932	2112	4714	10584
		R_rfft_x2_i16ci32			434	914	2048	4584	10200
		R_rfft_tw32_i16ci32			430	917	2053	4651	10332
		R_rfft_sc_tw32_i16ci32			445	925	2102	4686	10526
		R_rfft_x2_tw32_i16ci32			436	912	2040	4561	10146
i32	ci32	R_DSP_FFT_i32ci32	-	-	439	921	2047	4635	10280
				SC	448	924	2090	4660	10467
				X2	449	912	2029	4536	10087
		R_rfft_i32ci32	-		428	913	2039	4627	10272
		R_rfft_sc_i32ci32			435	914	2079	4650	10457
		R_rfft_x2_i32ci32			428	899	2017	4524	10075
f32	cf32	R_DSP_FFT_f32cf32	-		403	832	1873	4438	9925
		R_rfft_f32cf32	-		394	828	1869	4435	9922

points = 512 – 2048

in	out	Function name	option		points		
			TW32	Scale	512	1024	2048
i16	ci16	R_DSP_FFT_i16ci16	-	-	20698	45667	99436
				SC	20569	45668	98412
				X2	20200	44658	96442
			TW32	-	21537	47658	103539
				SC	21808	48443	104641
				X2	21786	48164	104556
		R_rfft_i16ci16	-		20685	45654	99423
		R_rfft_sc_i16ci16			20556	45655	98399
		R_rfft_x2_i16ci16			20189	44647	96431
		R_rfft_tw32_i16ci16			21519	47640	103521
		R_rfft_sc_tw32_i16ci16			21790	48425	104623
		R_rfft_x2_tw32_i16ci16			21769	48147	104539
	ci32	R_DSP_FFT_i16ci32	-	-	23098	50541	110387
				SC	23227	51309	110901
				X2	22454	49258	106800
			TW32	-	22974	50291	109880
				SC	23107	51063	110397
				X2	22353	49030	106378
		R_rfft_i16ci32	-		23088	50531	110377
		R_rfft_sc_i16ci32			23216	51298	110890
		R_rfft_x2_i16ci32			22446	49250	106792
		R_rfft_tw32_i16ci32			22961	50278	109867
		R_rfft_sc_tw32_i16ci32			23092	51048	110382
		R_rfft_x2_tw32_i16ci32			22342	49019	106367
i32	ci32	R_DSP_FFT_i32ci32	-	-	22864	50049	109449
				SC	22970	50797	109876
				X2	22221	48768	105862
		R_rfft_i32ci32	-		22856	50041	109441
		R_rfft_sc_i32ci32			22960	50787	109866
		R_rfft_x2_i32ci32			22209	48756	105850
f32	cf32	R_DSP_FFT_f32cf32	-		22764	50015	111846
		R_rfft_f32cf32	-		22761	50012	111843

(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	-	-	433	877	1940	4318	9626
				SC	441	880	1969	4317	9689
				X2	432	871	1932	4250	9491
			TW32	-	445	896	2000	4451	9967
				SC	460	915	2052	4515	10160
				X2	453	913	2031	4517	10096
		R_irfft_ci16i16	-		419	864	1930	4308	9616
		R_irfft_sc_ci16i16			426	866	1956	4304	9676
		R_irfft_x2_ci16i16			423	861	1923	4241	9482
		R_irfft_tw32_ci16i16			431	882	1987	4438	9954
		R_irfft_sc_tw32_ci16i16			445	899	2037	4500	10145
		R_irfft_x2_tw32_ci16i16			439	900	2019	4505	10084
ci32		R_DSP_IFFT_CCS_ci32i16	-	-	462	927	2043	4527	10041
				SC	472	940	2078	4531	10108
				X2	465	925	2042	4460	9912
			TW32	-	476	956	2111	4664	10387
				SC	488	968	2155	4726	10577
				X2	481	964	2136	4720	10508
		R_irfft_ci32i16	-		447	914	2033	4517	10031
		R_irfft_sc_ci32i16			458	926	2065	4518	10095
		R_irfft_x2_ci32i16			454	915	2033	4451	9903
		R_irfft_tw32_ci32i16			460	941	2097	4650	10373
		R_irfft_sc_tw32_ci32i16			469	949	2135	4708	10557
		R_irfft_x2_tw32_ci32i16			466	950	2123	4707	10495
	i32	R_DSP_IFFT_CCS_ci32i32	-	-	462	957	2115	4762	10537
				SC	474	966	2164	4799	10735
				X2	467	952	2102	4669	10349
		R_irfft_ci32i32	-		451	947	2106	4753	10528
		R_irfft_sc_ci32i32			460	955	2154	4789	10725
		R_irfft_x2_ci32i32			452	940	2091	4658	10338
cf32	f32	R_DSP_IFFT_CCS_cf32f32	-		495	997	2183	5035	11099
		R_irfft_cf32f32	-		484	988	2177	5029	11093

points = 512 – 2048

in	out	Function name	option		points		
			TW32	Scale	512	1024	2048
ci16	i16	R_DSP_IFFT_CCS_ci16i16	-	-	21216	46701	101491
				SC	21087	46699	100465
				X2	20700	45669	98414
			TW32	-	21940	48448	105157
				SC	22181	49202	106103
				X2	22199	48963	106186
		R_irfft_ci16i16	-		21206	46691	101481
		R_irfft_sc_ci16i16			21074	46686	100452
		R_irfft_x2_ci16i16			20691	45660	98405
		R_irfft_tw32_ci16i16			21927	48435	105144
		R_irfft_sc_tw32_ci16i16			22166	49187	106088
		R_irfft_x2_tw32_ci16i16			22187	48951	106174
ci32		R_DSP_IFFT_CCS_ci32i16	-	-	22049	48363	104819
				SC	21925	48366	103799
				X2	21534	47338	101744
			TW32	-	22762	50101	108412
				SC	23031	50882	109512
				X2	23010	50606	109428
		R_irfft_ci32i16	-		22039	48353	104809
		R_irfft_sc_ci32i16			21912	48353	103786
		R_irfft_x2_ci32i16			21525	47329	101735
		R_irfft_tw32_ci32i16			22748	50087	108398
		R_irfft_sc_tw32_ci32i16			23013	50862	109494
		R_irfft_x2_tw32_ci32i16			22997	50593	109415
	i32	R_DSP_IFFT_CCS_ci32i32	-	-	23360	51059	111418
				SC	23509	51849	112015
				X2	22723	49783	107837
		R_irfft_ci32i32	-		23351	51050	111409
		R_irfft_sc_ci32i32			23499	51839	112005
		R_irfft_x2_ci32i32			22712	49772	107826
cf32	f32	R_DSP_IFFT_CCS_cf32f32	-		25089	54645	121083
		R_irfft_cf32f32	-		25083	54639	121077

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