RX Family RXv1 CPU Products

RX DSP Library Version 5.0 Additional Information

R01AN4360EJ0100
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 RXv1 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	No	RX_DSP_NOFPU_LE.lib
		Yes	RX_DSP_NOFPU_LE_Check.lib
	Big-endian	No	RX_DSP_NOFPU_BE.lib
		Yes	RX_DSP_NOFPU_BE_Check.lib
Supported	Little-endian	No	RX_DSP_FPU_LE.lib
		Yes	RX_DSP_FPU_LE_Check.lib
	Big-endian	No	RX_DSP_FPU_BE.lib
		Yes	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 V1.02.01

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, No Error Checking

C/C++ Compiler options	Assembly options	Linker options
-cpu=rx200 -define=R_DSP_PARAMETER_CHECK=0 -define=__RXV1=1 -optimize=max -speed -nofpu -nologo	-cpu=rx200 -define=R_DSP_OVERFLOW_CHECK=0 -define=__RXV1=1 -nologo -chkfpu	-noprelink -form=library=u -nomessage -nologo -output="RX_DSP_NOFPU_LE.lib"

(2) RX_DSP_NOFPU_LE_Check

FPU Not Supported, Little Endian, Error Checking

C/C++ Compiler options	Assembly options	Linker options
-cpu=rx200 -define=R_DSP_PARAMETER_CHECK=1 -define=__RXV1=1 -optimize=max -speed -nofpu -nologo	-cpu=rx200 -define=R_DSP_OVERFLOW_CHECK=1 -define=__RXV1=1 -nologo -chkfpu	-noprelink -form=library=u -nomessage -nologo -output="RX_DSP_NOFPU_LE_Check.lib"

(3) RX_DSP_NOFPU_BE

FPU Not Supported, Big Endian, No Error Checking

C/C++ Compiler options	Assembly options	Linker options
-cpu=rx200 -define=R_DSP_PARAMETER_CHECK=0 -define=__RXV1=1 -optimize=max -speed -nofpu -endian=big -nologo	-cpu=rx200 -define=R_DSP_OVERFLOW_CHECK=0 -define=__RXV1=1 -nologo -chkfpu -endian=big	-noprelink -form=library=u -nomessage -nologo -output="RX_DSP_NOFPU_BE.lib"

(4) RX_DSP_NOFPU_BE_Check

FPU Not Supported, Big Endian, Error Checking

C/C++ Compiler options	Assembly options	Linker options
-cpu=rx200 -define=R_DSP_PARAMETER_CHECK=1 -define=__RXV1=1 -optimize=max -speed -nofpu -endian=big -nologo	-cpu=rx200 -define=R_DSP_OVERFLOW_CHECK=1 -define=__RXV1=1 -nologo -chkfpu -endian=big	-noprelink -form=library=u -nomessage -nologo -output="RX_DSP_NOFPU_BE_Check.lib"

(5) RX_DSP_FPU_LE

FPU Supported, Little Endian, No Error Checking

C/C++ Compiler options	Assembly options	Linker options
-cpu=rx600 -define=R_DSP_PARAMETER_CHECK=0 -define=__RXV1=1 -optimize=max -speed -fpu -nologo	-cpu=rx600 -define=R_DSP_OVERFLOW_CHECK=0 -define=__FPU=1 -define=__RXV1=1 -nologo	-noprelink -form=library=u -nomessage -nologo -output="RX_DSP_FPU_LE.lib"

(6) RX_DSP_FPU_LE_Check

FPU Supported, Little Endian, Error Checking

C/C++ Compiler options	Assembly options	Linker options
-cpu=rx600 -define=R_DSP_PARAMETER_CHECK=1 -define=__RXV1=1 -optimize=max -speed -fpu -nologo	-cpu=rx600 -define=R_DSP_OVERFLOW_CHECK=1 -define=__FPU=1 -define=__RXV1=1 -nologo	-noprelink -form=library=u -nomessage -nologo -output="RX_DSP_FPU_LE_Check.lib"

(7) RX_DSP_FPU_BE

FPU Supported, Big Endian, No Error Checking

C/C++ Compiler options	Assembly options	Linker options
-cpu=rx600 -define=R_DSP_PARAMETER_CHECK=0 -define=__RXV1=1 -optimize=max -speed -fpu -endian=big -nologo	-cpu=rx600 -define=R_DSP_OVERFLOW_CHECK=0 -define=__FPU=1 -define=__RXV1=1 -nologo -endian=big	-noprelink -form=library=u -nomessage -nologo -output="RX_DSP_FPU_BE.lib"

(8) RX_DSP_FPU_BE_Check

FPU Supported, Big Endian, Error Checking

C/C++ Compiler options	Assembly options	Linker options
-cpu=rx600 -define=R_DSP_PARAMETER_CHECK=1 -define=__RXV1=1 -optimize=max -speed -fpu -endian=big -nologo	-cpu=rx600 -define=R_DSP_OVERFLOW_CHECK=1 -define=__FPU=1 -define=__RXV1=1 -nologo -endian=big	-noprelink -form=library=u -nomessage -nologo -output="RX_DSP_FPU_BE_Check.lib"

2. Internal Functions

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

Show the internal functions which can be called by the user program.

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 of the structure are referred to. For details, see the User's Manual. 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	
R_DSP_FIR_i16i16	NO SATURATE	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)
R_DSP_FIR_ci16ci16	NO SATURATE	TRUNC	R_DSP_FIR_ci16ci16_asm_nt(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_ci16ci16_asm_n2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_FIR_ci16ci16_asm_st(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_ci16ci16_asm_s2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	R_DSP_FIR_i16i32_asm_nt(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_i16i32_asm_n2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_FIR_i16i32_asm_st(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_i16i32_asm_s2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
R_DSP_FIR_ci16ci32	NO SATURATE	TRUNC	R_DSP_FIR_ci16ci32_asm_nt(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_ci16ci32_asm_n2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_FIR_ci16ci32_asm_st(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_ci16ci32_asm_s2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	R_DSP_FIR_i32i32_asm_nt(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_i32i32_asm_n2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_FIR_i32i32_asm_st(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_i32i32_asm_s2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
R_DSP_FIR_ci32ci32	NO SATURATE	TRUNC	R_DSP_FIR_ci32ci32_asm_nt(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_ci32ci32_asm_n2(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_FIR_ci32ci32_asm_st(const r_dsp_firfilter_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_FIR_ci32ci32_asm_s2(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 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 functions of the library with "_Check").

Public Functions list

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_IIRBiquad_i16i16	NO SATURATE	TRUNC	R_DSP_IIRBiquad_i16i16_asm_nt(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_i16i16_asm_n2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRBiquad_i16i16_asm_st(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_i16i16_asm_s2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
R_DSP_IIRBiquad_ci16ci16	NO SATURATE	TRUNC	R_DSP_IIRBiquad_ci16ci16_asm_nt(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_ci16ci16_asm_n2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRBiquad_ci16ci16_asm_st(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_ci16ci16_asm_s2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)

public function	<option>		internal function
	SATURATE	ROUNDING	
R_DSP_IIRBiquad_i16i32	NO SATURATE	TRUNC	R_DSP_IIRBiquad_i16i32_asm_nt(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_i16i32_asm_n2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRBiquad_i16i32_asm_st(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_i16i32_asm_s2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
R_DSP_IIRBiquad_ci16ci32	NO SATURATE	TRUNC	R_DSP_IIRBiquad_ci16ci32_asm_nt(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_ci16ci32_asm_n2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRBiquad_ci16ci32_asm_st(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_ci16ci32_asm_s2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
R_DSP_IIRBiquad_i32i32	NO SATURATE	TRUNC	R_DSP_IIRBiquad_i32i32_asm_nt(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_i32i32_asm_n2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRBiquad_i32i32_asm_st(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_i32i32_asm_s2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
R_DSP_IIRBiquad_ci32ci32	NO SATURATE	TRUNC	R_DSP_IIRBiquad_ci32ci32_asm_nt(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_ci32ci32_asm_n2(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
	SATURATE	TRUNC	R_DSP_IIRBiquad_ci32ci32_asm_st(const r_dsp_iirbiquad_t * handle, const vector_t * input, vector_t * output)
		NEAREST	R_DSP_IIRBiquad_ci32ci32_asm_s2(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 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 functions of the 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)
		NEARES T	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)
		NEARES T	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 beginning 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 beginning 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 beginning 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)
		NEAREST	R_DSP_MatrixMul_i16i16_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixMul_i16i16_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixMul_i16i16_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
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)
		NEAREST	R_DSP_MatrixMul_ci16ci16_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixMul_ci16ci16_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixMul_ci16ci16_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
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)
		NEAREST	R_DSP_MatrixMul_i16i32_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixMul_i16i32_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixMul_i16i32_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
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)
		NEAREST	R_DSP_MatrixMul_ci16ci32_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixMul_ci16ci32_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixMul_ci16ci32_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
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)
		NEAREST	R_DSP_MatrixMul_i32i32_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixMul_i32i32_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixMul_i32i32_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)

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)
		NEAREST	R_DSP_MatrixMul_ci32ci32_asm_n2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixMul_ci32ci32_asm_st(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixMul_ci32ci32_asm_s2(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
R_DSP_MatrixMul_f32f32	-	-	R_DSP_MatrixMul_f32f32_asm(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)
R_DSP_MatrixMul_cf32cf32	-	-	R_DSP_MatrixMul_cf32cf32_asm(const matrix_t * inputA, const matrix_t * inputB, matrix_t * output, scale_t shift)

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 beginning 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 beginning 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)
		NEAREST	R_DSP_MatrixScale_i16i16_asm_n2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixScale_i16i16_asm_st(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixScale_i16i16_asm_s2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
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)
		NEAREST	R_DSP_MatrixScale_ci16ci16_asm_n2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixScale_ci16ci16_asm_st(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixScale_ci16ci16_asm_s2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
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)
		NEAREST	R_DSP_MatrixScale_i16i32_asm_n2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixScale_i16i32_asm_st(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixScale_i16i32_asm_s2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
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)
		NEAREST	R_DSP_MatrixScale_ci16ci32_asm_n2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixScale_ci16ci32_asm_st(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixScale_ci16ci32_asm_s2(const matrix_t * input, const int16_t scalar, matrix_t * output, scale_t shift)
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)
		NEAREST	R_DSP_MatrixScale_i32i32_asm_n2(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixScale_i32i32_asm_st(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixScale_i32i32_asm_s2(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift)

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)
		NEAREST	R_DSP_MatrixScale_ci32ci32_asm_n2(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift)
	SATURATE	TRUNC	R_DSP_MatrixScale_ci32ci32_asm_st(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift)
		NEAREST	R_DSP_MatrixScale_ci32ci32_asm_s2(const matrix_t * input, const int32_t scalar, matrix_t * output, scale_t shift)
R_DSP_MatrixScale_f32f32	-	-	R_DSP_MatrixScale_f32f32_asm(const matrix_t * input, const float scalar, matrix_t * output, scale_t shift)
R_DSP_MatrixScale_cf32cf32	-	-	R_DSP_MatrixScale_cf32cf32_asm(const matrix_t * input, const float scalar, matrix_t * output, scale_t shift)

2.3 Internal Function of Linear Transform API

This section describes the internal functions of the following filter 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 point floating point complex FFT functions remain 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 where the calculation result is stored.

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_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 (in case of fixed-point functions of the 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. The storage order of the output data is as follows. (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 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	814	164	871	176
	i32	int32_t	R_DSP_Mean_i32	794	168	836	168
	f32	float	R_DSP_Mean_f32	384	24	452	24
Minimum value	i16	int16_t	R_DSP_Min_i16	124	24	177	28
	i32	int32_t	R_DSP_Min_i32	114	24	164	28
	f32	float	R_DSP_Min_f32	528	24	594	24
Maximum value	i16	int16_t	R_DSP_Max_i16	124	24	177	28
	i32	int32_t	R_DSP_Max_i32	114	24	164	28
	f32	float	R_DSP_Max_f32	528	24	594	24
Minimum value with index	i16	int16_t	R_DSP_ArgMin_i16	923	52	1137	128
	i32	int32_t	R_DSP_ArgMin_i32	837	52	941	72
	f32	float	R_DSP_ArgMin_f32	185	48	245	48
Maximum value with index	i16	int16_t	R_DSP_ArgMax_i16	923	52	1137	128
	i32	int32_t	R_DSP_ArgMax_i32	837	52	941	72
	f32	float	R_DSP_ArgMax_f32	185	48	245	48
Mean absolute value and maximum absolute value	i16	int16_t	R_DSP_MeanAbs_i16	107	24	156	24
	i32	int32_t	R_DSP_MeanAbs_i32	177	48	221	44
	f32	float	R_DSP_MeanAbs_f32	176	56	217	56
	i16	int16_t	R_DSP_MaxAbs_i16	130	24	187	28
	i32	int32_t	R_DSP_MaxAbs_i32	126	16	185	16
	f32	float	R_DSP_MaxAbs_f32	160	48	208	48
	i16	int16_t	R_DSP_MeanMaxAbs_i16	170	36	230	40
	i32	int32_t	R_DSP_MeanMaxAbs_i32	244	64	313	68
	f32	float	R_DSP_MeanMaxAbs_f32	243	68	296	68
Mean value and variance	i16	int16_t	R_DSP_MeanVar_i16	35	184	87	196
	i32	int32_t	R_DSP_MeanVar_i32	35	188	87	188
	f32	float	R_DSP_MeanVar_f32	35	44	87	44
Variance	i16	int16_t	R_DSP_Var_GivenMean_i16	14	20	51	20
	i32	int32_t	R_DSP_Var_GivenMean_i32	14	28	51	28
	f32	float	R_DSP_Var_GivenMean_f32	14	8	51	8
Histogram	i16	uint16_t	R_DSP_Histogram_i16ui16	260	36	376	44
	i32	uint16_t	R_DSP_Histogram_i32ui16	343	44	437	52
	f32	uint16_t	R_DSP_Histogram_f32ui16	1261	36	424	36
Mean value and mean absolute deviation (MAD)	i16	int16_t	R_DSP_MeanMAD_i16	35	184	87	196
	i32	int32_t	R_DSP_MeanMAD_i32	35	188	87	188
	f32	float	R_DSP_MeanMAD_f32	35	44	87	44
Mean absolute deviation (MAD)	i16	int16_t	R_DSP_MAD_GivenMean_i16	14	12	51	12
	i32	int32_t	R_DSP_MAD_GivenMean_i32	14	12	51	12
	f32	float	R_DSP_MAD_GivenMean_f32	14	8	51	8
Median functions	i16	int16_t	R_DSP_Median_InPlace_i16	72	36	119	36
	i32	int32_t	R_DSP_Median_InPlace_i32	72	36	119	36
	f32	float	R_DSP_Median_InPlace_f32	73	36	120	36
	i16	int16_t	R_DSP_Median_i16	197	40	280	40
	i32	int32_t	R_DSP_Median_i32	197	40	280	40
	f32	float	R_DSP_Median_f32	198	40	281	40

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	85	4	115	4
		R_DSP_FIR_i16i16	346	68	485	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	85	4	115	4
		R_DSP_FIR_i16i32	446	68	612	68
		R_DSP_FIR_i16i32_asm_nt	87	32	111	32
		R_DSP_FIR_i16i32_asm_n2	92	32	120	32
		R_DSP_FIR_i16i32_asm_st	112	32	118	32
		R_DSP_FIR_i16i32_asm_s2	119	32	125	32
ci16	ci16	R_DSP_FIR_Init_ci16ci16	97	4	127	4
		R_DSP_FIR_ci16ci16	1316	116	1509	124
		R_DSP_FIR_ci16ci16_asm_nt	297	56	330	60
		R_DSP_FIR_ci16ci16_asm_n2	305	56	338	60
		R_DSP_FIR_ci16ci16_asm_st	335	56	348	60
		R_DSP_FIR_ci16ci16_asm_s2	343	56	356	60
	ci32	R_DSP_FIR_Init_ci16ci32	97	4	127	4
		R_DSP_FIR_ci16ci32	1520	116	1776	124
		R_DSP_FIR_ci16ci32_asm_nt	336	56	395	60
		R_DSP_FIR_ci16ci32_asm_n2	346	56	415	60
		R_DSP_FIR_ci16ci32_asm_st	394	56	407	60
		R_DSP_FIR_ci16ci32_asm_s2	408	56	421	60
i32	i32	R_DSP_FIR_Init_i32i32	97	4	127	4
		R_DSP_FIR_i32i32	592	76	825	76
		R_DSP_FIR_i32i32_asm_nt	108	36	162	36
		R_DSP_FIR_i32i32_asm_n2	114	36	180	36
		R_DSP_FIR_i32i32_asm_st	161	36	167	36
		R_DSP_FIR_i32i32_asm_s2	173	36	179	36
ci32	ci32	R_DSP_FIR_Init_ci32ci32	124	4	158	4
		R_DSP_FIR_ci32ci32	2188	140	2573	148
		R_DSP_FIR_ci32ci32_asm_nt	477	68	596	72
		R_DSP_FIR_ci32ci32_asm_n2	489	68	628	72
		R_DSP_FIR_ci32ci32_asm_st	583	68	596	72
		R_DSP_FIR_ci32ci32_asm_s2	603	68	616	72
f32	f32	R_DSP_FIR_Init_f32f32	97	4	127	4
		R_DSP_FIR_f32f32	117	76	217	76
		R_DSP_FIR_f32f32_asm	113	36	113	36
cf32	cf32	R_DSP_FIR_Init_cf32cf32	124	4	158	4
		R_DSP_FIR_cf32cf32	236	92	336	92
		R_DSP_FIR_cf32cf32_asm	232	44	232	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	15	4	40	4
		R_DSP_IIRBiquad_Init_i16i16	93	4	125	4
		R_DSP_IIRBiquad_i16i16	2695	108	3062	116
		R_DSP_IIRBiquad_i16i16_asm_nt	598	52	681	56
		R_DSP_IIRBiquad_i16i16_asm_n2	626	52	711	56
		R_DSP_IIRBiquad_i16i16_asm_st	703	52	742	56
		R_DSP_IIRBiquad_i16i16_asm_s2	732	52	770	56
	i32	R_DSP_IIRBiquad_StateSize_i16i32	15	4	40	4
		R_DSP_IIRBiquad_Init_i16i32	93	4	125	4
		R_DSP_IIRBiquad_i16i32	3017	108	3483	116
		R_DSP_IIRBiquad_i16i32_asm_nt	663	52	778	56
		R_DSP_IIRBiquad_i16i32_asm_n2	691	52	842	56
		R_DSP_IIRBiquad_i16i32_asm_st	792	52	831	56
		R_DSP_IIRBiquad_i16i32_asm_s2	835	52	873	56
ci16	ci16	R_DSP_IIRBiquad_StateSize_ci16ci16	15	4	40	4
		R_DSP_IIRBiquad_Init_ci16ci16	105	4	138	4
		R_DSP_IIRBiquad_ci16ci16	3920	164	4394	172
		R_DSP_IIRBiquad_ci16ci16_asm_nt	878	80	994	84
		R_DSP_IIRBiquad_ci16ci16_asm_n2	918	80	1034	84
		R_DSP_IIRBiquad_ci16ci16_asm_st	1024	80	1084	84
		R_DSP_IIRBiquad_ci16ci16_asm_s2	1064	80	1124	84
	ci32	R_DSP_IIRBiquad_StateSize_ci16ci32	15	4	40	4
		R_DSP_IIRBiquad_Init_ci16ci32	105	4	138	4
		R_DSP_IIRBiquad_ci16ci32	4528	156	5167	164
		R_DSP_IIRBiquad_ci16ci32_asm_nt	1001	76	1177	80
		R_DSP_IIRBiquad_ci16ci32_asm_n2	1041	76	1261	80
		R_DSP_IIRBiquad_ci16ci32_asm_st	1195	76	1255	80
		R_DSP_IIRBiquad_ci16ci32_asm_s2	1255	76	1315	80
i32	i32	R_DSP_IIRBiquad_StateSize_i32i32	15	4	40	4
		R_DSP_IIRBiquad_Init_i32i32	105	4	138	4
		R_DSP_IIRBiquad_i32i32	3722	124	4382	132
		R_DSP_IIRBiquad_i32i32_asm_nt	791	60	994	64
		R_DSP_IIRBiquad_i32i32_asm_n2	826	60	1084	64
		R_DSP_IIRBiquad_i32i32_asm_st	1006	60	1045	64
		R_DSP_IIRBiquad_i32i32_asm_s2	1063	60	1101	64
ci32	ci32	R_DSP_IIRBiquad_StateSize_ci32ci32	15	4	40	4
		R_DSP_IIRBiquad_Init_ci32ci32	135	4	172	4
		R_DSP_IIRBiquad_ci32ci32	5876	188	6834	204
		R_DSP_IIRBiquad_ci32ci32_asm_nt	1259	92	1576	100
		R_DSP_IIRBiquad_ci32ci32_asm_n2	1311	92	1712	100
		R_DSP_IIRBiquad_ci32ci32_asm_st	1591	92	1650	100
		R_DSP_IIRBiquad_ci32ci32_asm_s2	1679	92	1738	100
f32	f32	R_DSP_IIRBiquad_StateSize_f32f32	15	4	40	4
		R_DSP_IIRBiquad_Init_f32f32	105	4	138	4
		R_DSP_IIRBiquad_f32f32	435	108	545	108
		R_DSP_IIRBiquad_f32f32_asm	431	52	431	52

in	out	Function name	No Checked		Checked	
			Code [bytes]	Stack [bytes]	Code [bytes]	Stack [bytes]
cf32	cf32	R_DSP_IIRBiquad_StateSize_cf32cf32	15	4	40	4
		R_DSP_IIRBiquad_Init_cf32cf32	135	4	172	4
		R_DSP_IIRBiquad_cf32cf32	1055	124	1165	124
		R_DSP_IIRBiquad_cf32cf32_asm	1051	60	1051	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	417	36	558	44
		R_DSP_IIRSinglePole_i16i16_asm_nt	83	16	92	20
		R_DSP_IIRSinglePole_i16i16_asm_n2	91	16	100	20
		R_DSP_IIRSinglePole_i16i16_asm_st	101	16	107	20
		R_DSP_IIRSinglePole_i16i16_asm_s2	106	16	112	20
i16	i32	R_DSP_IIRSinglePole_i16i32	460	44	597	52
		R_DSP_IIRSinglePole_i16i32_asm_nt	96	20	103	24
		R_DSP_IIRSinglePole_i16i32_asm_n2	105	20	111	24
		R_DSP_IIRSinglePole_i16i32_asm_st	109	20	115	24
		R_DSP_IIRSinglePole_i16i32_asm_s2	114	20	120	24
i32	i32	R_DSP_IIRSinglePole_i32i32	709	60	971	84
		R_DSP_IIRSinglePole_i32i32_asm_nt	152	28	169	32
		R_DSP_IIRSinglePole_i32i32_asm_n2	166	28	186	32
		R_DSP_IIRSinglePole_i32i32_asm_st	172	28	186	32
		R_DSP_IIRSinglePole_i32i32_asm_s2	183	28	203	32
f32	f32	R_DSP_IIRSinglePole_f32f32	90	44	222	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	47	100	105	100
		ci32	R_DSP_DFT_ci16ci32	47	100	105	100
	ci32		R_DSP_DFT_ci32ci32	47	100	105	100
	cf32	cf32	R_DSP_DFT_cf32cf32	47	108	105	108
Complex IDFT	ci16	ci16	R_DSP_IDFT_ci16ci16	48	112	107	112
	ci32		R_DSP_IDFT_ci32ci16	48	112	107	112
		ci32	R_DSP_IDFT_ci32ci32	48	112	107	112
	cf32	cf32	R_DSP_IDFT_cf32cf32	48	116	107	116
Real DFT	i16	ci16	R_DSP_DFT_i16ci16	47	96	108	96
		ci32	R_DSP_DFT_i16ci32	47	96	108	96
	i32		R_DSP_DFT_i32ci32	47	96	108	96
	f32	cf32	R_DSP_DFT_f32cf32	47	108	108	108
Complex Conjugate Symmetry IDFT	ci16	ci16	R_DSP_IDFT_CCS_ci16i16	50	100	114	100
	ci32		R_DSP_IDFT_CCS_ci32i16	50	100	114	100
		ci32	R_DSP_IDFT_CCS_ci32i32	50	100	114	100
	cf32	cf32	R_DSP_IDFT_CCS_cf32f32	50	112	114	112

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	175	20	227	16
	ci32	R_DSP_FFT_BufSize_i16ci32	175	20	227	16
i32	ci32	R_DSP_FFT_BufSize_i32ci32	161	40	203	16
f32	cf32	R_DSP_FFT_BufSize_f32cf32	161	16	189	16
ci16	i16	R_DSP_FFT_BufSize_ci16i16	175	20	227	16
	ci16	R_DSP_FFT_BufSize_ci16ci16	160	12	202	12
	ci32	R_DSP_FFT_BufSize_ci16ci32	160	12	202	12
ci32	i16	R_DSP_FFT_BufSize_ci32i16	175	20	227	16
	i32	R_DSP_FFT_BufSize_ci32i32	same as i32ci32			
	ci16	R_DSP_FFT_BufSize_ci32ci16	160	12	202	12
	ci32	R_DSP_FFT_BufSize_ci32ci32	151	12	193	12
cf32	f32	R_DSP_FFT_BufSize_cf32f32	161	16	189	16
	cf32	R_DSP_FFT_BufSize_cf32cf32	151	12	179	12

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	582	52	615	64
	ci32	R_DSP_FFT_Init_i16ci32	582	52	615	64
i32	ci32	R_DSP_FFT_Init_i32ci32	408	52	452	56
f32	cf32	R_DSP_FFT_Init_f32cf32	454	80	489	84
ci16	i16	R_DSP_FFT_Init_ci16i16	576	52	633	60
	ci16	R_DSP_FFT_Init_ci16ci16	404	44	459	52
	ci32	R_DSP_FFT_Init_ci16ci32	404	44	459	52
ci32	i16	R_DSP_FFT_Init_ci32i16	576	52	633	60
	i32	R_DSP_FFT_Init_ci32i32	403	52	448	56
	ci16	R_DSP_FFT_Init_ci32ci16	404	44	459	52
	ci32	R_DSP_FFT_Init_ci32ci32	308	44	353	44
cf32	f32	R_DSP_FFT_Init_cf32f32	433	80	468	84
	cf32	R_DSP_FFT_Init_cf32cf32	354	72	384	72

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	9627	116	11720	120
			R_cfft_ci16ci16	1365	56	1682	56
			R_cfft_sc_ci16ci16	1385	56	1638	56
			R_cfft_x2_ci16ci16	1365	56	1618	56
			R_cfft_tw32_ci16ci16	1802	56	2168	56
			R_cfft_sc_tw32_ci16ci16	1822	56	2198	56
			R_cfft_x2_tw32_ci16ci16	1818	56	2184	56
		ci32	R_DSP_FFT_ci16ci32	9555	116	11241	120
			R_cfft_ci16ci32	1567	56	1802	56
			R_cfft_sc_ci16ci32	1607	56	1803	56
			R_cfft_x2_ci16ci32	1567	56	1762	56
			R_cfft_tw32_ci16ci32	1568	56	1894	56
			R_cfft_sc_tw32_ci16ci32	1608	56	1894	56
			R_cfft_x2_tw32_ci16ci32	1568	56	1854	56
	ci32	ci32	R_DSP_FFT_ci32ci32	5655	116	6715	120
			R_cfft_ci32ci32	1860	56	2188	56
			R_cfft_sc_ci32ci32	1900	56	2188	56
			R_cfft_x2_ci32ci32	1860	56	2148	56
	cf32	cf32	R_DSP_FFT_cf32cf32	1360	116	1491	120
			R_cfft_cf32cf32	1352	56	1352	56
icFFT	ci16	ci16	R_DSP_IFFT_ci16ci16	9627	116	11720	120
			R_icfft_ci16ci16	1365	56	1682	56
			R_icfft_sc_ci16ci16	1385	56	1638	56
			R_icfft_x2_ci16ci16	1365	56	1618	56
			R_icfft_tw32_ci16ci16	1802	56	2168	56
			R_icfft_sc_tw32_ci16ci16	1822	56	2198	56
			R_icfft_x2_tw32_ci16ci16	1818	56	2184	56
	ci32		R_DSP_IFFT_ci32ci16	8573	116	10227	120
			R_icfft_ci32ci16	1176	56	1491	56
			R_icfft_sc_ci32ci16	1196	56	1447	56
			R_icfft_x2_ci32ci16	1176	56	1427	56
			R_icfft_tw32_ci32ci16	1645	56	1897	56
			R_icfft_sc_tw32_ci32ci16	1665	56	1836	56
			R_icfft_x2_tw32_ci32ci16	1645	56	1897	56
		ci32	R_DSP_IFFT_ci32ci32	5655	116	6715	120
			R_icfft_ci32ci32	1860	56	2188	56
			R_icfft_sc_ci32ci32	1900	56	2188	56
			R_icfft_x2_ci32ci32	1860	56	2148	56
	cf32	cf32	R_DSP_IFFT_cf32cf32	1360	116	1491	120
			R_icfft_cf32cf32	1352	56	1352	56

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
rFFT	i16	ci16	R_DSP_FFT_i16ci16	13185	116	15790	124
			R_rfft_i16ci16	1839	56	2241	56
			R_rfft_sc_i16ci16	1859	56	2197	56
			R_rfft_x2_i16ci16	1839	56	2177	56
			R_rfft_tw32_i16ci16	2514	56	2958	56
			R_rfft_sc_tw32_i16ci16	2534	56	2988	56
			R_rfft_x2_tw32_i16ci16	2530	56	2974	56
		ci32	R_DSP_FFT_i16ci32	12072	116	14165	124
			R_rfft_i16ci32	1988	56	2257	56
			R_rfft_sc_i16ci32	2028	56	2258	56
			R_rfft_x2_i16ci32	1988	56	2217	56
			R_rfft_tw32_i16ci32	1986	56	2406	56
			R_rfft_sc_tw32_i16ci32	2026	56	2406	56
			R_rfft_x2_tw32_i16ci32	1986	56	2366	56
	i32	ci32	R_DSP_FFT_i32ci32	7827	116	9318	124
			R_rfft_i32ci32	2584	56	3052	56
			R_rfft_sc_i32ci32	2624	56	3052	56
			R_rfft_x2_i32ci32	2584	56	3012	56
	f32	cf32	R_DSP_FFT_f32cf32	1762	116	1898	124
			R_rfft_f32cf32	1754	56	1754	56
irFFT	ci16	i16	R_DSP_IFFT_CCS_ci16i16	13185	116	15787	124
			R_irfft_ci16i16	1839	56	2241	56
			R_irfft_sc_ci16i16	1859	56	2197	56
			R_irfft_x2_ci16i16	1839	56	2177	56
			R_irfft_tw32_ci16i16	2514	56	2958	56
			R_irfft_sc_tw32_ci16i16	2534	56	2988	56
			R_irfft_x2_tw32_ci16i16	2530	56	2974	56
	ci32		R_DSP_IFFT_CCS_ci32i16	10016	116	11903	124
			R_irfft_ci32i16	1368	56	1720	56
			R_irfft_sc_ci32i16	1388	56	1676	56
			R_irfft_x2_ci32i16	1368	56	1656	56
			R_irfft_tw32_ci32i16	1934	56	2220	56
			R_irfft_sc_tw32_ci32i16	1954	56	2159	56
			R_irfft_x2_tw32_ci32i16	1934	56	2220	56
		i32	R_DSP_IFFT_CCS_ci32i32	7827	116	9322	124
			R_irfft_ci32i32	2584	56	3052	56
			R_irfft_sc_ci32i32	2624	56	3052	56
			R_irfft_x2_ci32i32	2584	56	3012	56
	cf32	f32	R_DSP_IFFT_CCS_cf32f32	1762	116	1902	124
			R_irfft_cf32f32	1754	56	1754	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	22	4	22	4
	cf32	f32	R_DSP_CplxMag_cf32f32	14	4	14	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	91	4	91	4
	cf32	f32	R_DSP_CplxMag_Fast_cf32f32	53	4	53	4
	ci16	i16	R_DSP_VecCplxMag_ci16i16	131	48	246	48
		i32	R_DSP_VecCplxMag_ci16i32	131	48	246	48
	ci32		R_DSP_VecCplxMag_ci32i32	217	72	337	72
	cf32	f32	R_DSP_VecCplxMag_cf32f32	217	72	337	72
	ci16	i16	R_DSP_VecCplxMag_Fast_ci16i16	131	48	246	48
		i32	R_DSP_VecCplxMag_Fast_ci16i32	131	48	246	48
	ci32		R_DSP_VecCplxMag_Fast_ci32i32	217	72	337	72
	cf32	f32	R_DSP_VecCplxMag_Fast_cf32f32	217	72	337	72
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	17	4	17	4
	cf32	f32	R_DSP_CplxMagSquared_cf32f32	10	4	10	4
	ci16	i16	R_DSP_VecCplxMagSquared_ci16i16	131	48	246	48
		i32	R_DSP_VecCplxMagSquared_ci16i32	131	48	246	48
	ci32		R_DSP_VecCplxMagSquared_ci32i32	217	72	337	72
	cf32	f32	R_DSP_VecCplxMagSquared_cf32f32	217	72	337	72
Complex number phase	ci16	i16	R_DSP_CplxPhase_ci16i16	76	24	76	24
	ci32	i32	R_DSP_CplxPhase_ci32i32	90	28	90	28
	cf32	f32	R_DSP_CplxPhase_cf32f32	59	60	59	60
	ci16	i16	R_DSP_VecCplxPhase_ci16i16	250	76	366	76
	ci32	i32	R_DSP_VecCplxPhase_ci32i32	348	108	468	108
	cf32	f32	R_DSP_VecCplxPhase_cf32f32	460	140	580	140
Complex number addition	ci16	ci16	R_DSP_ComplexAdd_ci16ci16	19	4	41	4
	ci32	ci32	R_DSP_ComplexAdd_ci32ci32	19	8	38	8
	cf32	cf32	R_DSP_ComplexAdd_cf32cf32	21	8	30	8
Complex number subtraction	ci16	ci16	R_DSP_ComplexSub_ci16ci16	19	4	41	4
	ci32	ci32	R_DSP_ComplexSub_ci32ci32	19	8	38	8
	cf32	cf32	R_DSP_ComplexSub_cf32cf32	21	8	30	8
Complex number multiplication	ci16	ci16	R_DSP_ComplexMul_ci16ci16	67	4	99	4
	ci32		R_DSP_ComplexMul_ci16ci32	37	4	111	4
		ci32	R_DSP_ComplexMul_ci32ci32	62	8	140	8
	cf32	cf32	R_DSP_ComplexMul_cf32cf32	37	8	46	8
Complex conjugate	ci16	ci16	R_DSP_ComplexConjg_ci16ci16	14	4	30	4
	ci32	ci32	R_DSP_ComplexConjg_ci32ci32	9	4	23	4
	cf32	cf32	R_DSP_ComplexConjg_cf32cf32	10	4	18	4
	ci16	ci16	R_DSP_VecCplxConjg_ci16ci16	133	48	194	48
	ci32	ci32	R_DSP_VecCplxConjg_ci32ci32	219	72	281	72
	cf32	cf32	R_DSP_VecCplxConjg_cf32cf32	219	72	281	72

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	307	28
		i32	R_DSP_MatrixAdd_i16i32	123	20	256	24
	i32		R_DSP_MatrixAdd_i32i32	125	24	286	28
	f32	f32	R_DSP_MatrixAdd_f32f32	130	24	263	28
	ci16	ci16	R_DSP_MatrixAdd_ci16ci16	202	24	435	28
		ci32	R_DSP_MatrixAdd_ci16ci32	200	20	333	24
	ci32		R_DSP_MatrixAdd_ci32ci32	202	24	391	28
	cf32	cf32	R_DSP_MatrixAdd_cf32cf32	214	24	347	28
Matrix subtraction	i16	i16	R_DSP_MatrixSub_i16i16	125	24	307	28
		i32	R_DSP_MatrixSub_i16i32	123	20	256	24
	i32		R_DSP_MatrixSub_i32i32	125	24	286	28
	f32	f32	R_DSP_MatrixSub_f32f32	130	24	263	28
	ci16	ci16	R_DSP_MatrixSub_ci16ci16	202	24	435	28
		ci32	R_DSP_MatrixSub_ci16ci32	200	20	333	24
	ci32		R_DSP_MatrixSub_ci32ci32	202	24	391	28
	cf32	cf32	R_DSP_MatrixSub_cf32cf32	214	24	347	28
Matrix multiplication	i16	i16	R_DSP_MatrixMul_i16i16	825	76	1032	88
			R_DSP_MatrixMul_i16i16_asm_nt	182	36	202	40
			R_DSP_MatrixMul_i16i16_asm_n2	186	36	206	40
			R_DSP_MatrixMul_i16i16_asm_st	203	36	214	40
			R_DSP_MatrixMul_i16i16_asm_s2	207	36	218	40
		i32	R_DSP_MatrixMul_i16i32	909	76	1122	88
			R_DSP_MatrixMul_i16i32_asm_nt	204	36	224	40
			R_DSP_MatrixMul_i16i32_asm_n2	208	36	233	40
			R_DSP_MatrixMul_i16i32_asm_st	222	36	233	40
			R_DSP_MatrixMul_i16i32_asm_s2	228	36	239	40
	i32	i32	R_DSP_MatrixMul_i32i32	1431	116	1719	128
			R_DSP_MatrixMul_i32i32_asm_nt	316	56	372	60
			R_DSP_MatrixMul_i32i32_asm_n2	322	56	393	60
			R_DSP_MatrixMul_i32i32_asm_st	367	56	375	60
			R_DSP_MatrixMul_i32i32_asm_s2	379	56	387	60
	f32	f32	R_DSP_MatrixMul_f32f32	186	76	292	76
			R_DSP_MatrixMul_f32f32_asm	182	36	182	36
	ci16	ci16	R_DSP_MatrixMul_ci16ci16	2375	84	2604	96
			R_DSP_MatrixMul_ci16ci16_asm_nt	561	40	590	44
			R_DSP_MatrixMul_ci16ci16_asm_n2	569	40	598	44
			R_DSP_MatrixMul_ci16ci16_asm_st	595	40	608	44
			R_DSP_MatrixMul_ci16ci16_asm_s2	603	40	616	44
		ci32	R_DSP_MatrixMul_ci16ci32	2591	84	2891	96
			R_DSP_MatrixMul_ci16ci32_asm_nt	602	40	661	44
			R_DSP_MatrixMul_ci16ci32_asm_n2	610	40	679	44
			R_DSP_MatrixMul_ci16ci32_asm_st	660	40	673	44
			R_DSP_MatrixMul_ci16ci32_asm_s2	672	40	685	44

API	in	out	function name	No Checked		Checked	
				ROM [bytes]	Stack [bytes]	ROM [bytes]	Stack [bytes]
Matrix multiplication	ci32	ci32	R_DSP_MatrixMul_ci32ci32	3575	100	3994	112
			R_DSP_MatrixMul_ci32ci32_asm_nt	822	48	931	52
			R_DSP_MatrixMul_ci32ci32_asm_n2	834	48	973	52
			R_DSP_MatrixMul_ci32ci32_asm_st	924	48	937	52
			R_DSP_MatrixMul_ci32ci32_asm_s2	948	48	961	52
	cf32	cf32	R_DSP_MatrixMul_cf32cf32	392	100	498	100
			R_DSP_MatrixMul_cf32cf32_asm	388	48	388	48
Matrix transposition	i16	i16	R_DSP_MatrixTrans_i16i16	104	36	155	36
		i32	R_DSP_MatrixTrans_i16i32	105	36	156	36
	i32		R_DSP_MatrixTrans_i32i32	104	36	155	36
	f32	f32	R_DSP_MatrixTrans_f32f32	104	36	155	36
	ci16	ci16	R_DSP_MatrixTrans_ci16ci16	150	36	201	36
		ci32	R_DSP_MatrixTrans_ci16ci32	150	36	201	36
	ci32		R_DSP_MatrixTrans_ci32ci32	150	36	201	36
	cf32	cf32	R_DSP_MatrixTrans_cf32cf32	150	36	201	36
Matrix real number multiplication	i16	i16	R_DSP_MatrixScale_i16i16	637	40	753	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	693	40	798	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	1077	72	1285	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	192	48
			R_DSP_MatrixScale_f32f32_asm	124	20	124	20
	ci16	ci16	R_DSP_MatrixScale_ci16ci16	881	40	953	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	893	40	966	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	935	72	1007	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	276	48
			R_DSP_MatrixScale_cf32cf32_asm	208	20	208	20

4. Execution Cycle Count

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

The measurement conditions are

- Device: RX63N Group
- Library: R_DSP_FPU_LE.lib
- Code allocation: Flash memory
- Data allocation: Internal RAM (Data is allocated to 4-byte boundary alignment sections)

Target functions

Filter operation APIs

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

Transform kernels

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

4.1 Filter operation API

4.1.1 Generic FIR Filter

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	548	1052	2060	4076	8108	16172
		NEAREST	561	1081	2121	4201	8361	16681
	SATURATE	TRUNC	586	1130	2218	4394	8746	17450
		NEAREST	593	1145	2249	4457	8873	17705
R_DSP_FIR_i16i16_asm_nt	-		534	1038	2046	4062	8094	16158
R_DSP_FIR_i16i16_asm_n2			548	1068	2108	4188	8348	16668
R_DSP_FIR_i16i16_asm_st			573	1117	2205	4381	8733	17437
R_DSP_FIR_i16i16_asm_s2			581	1133	2237	4445	8861	17693

Taps=32, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	884	1724	3404	6764	13484	26924
		NEAREST	897	1753	3465	6889	13737	27433
	SATURATE	TRUNC	922	1802	3562	7082	14122	28202
		NEAREST	929	1817	3593	7145	14249	28457
R_DSP_FIR_i16i16_asm_nt	-		870	1710	3390	6750	13470	26910
R_DSP_FIR_i16i16_asm_n2			884	1740	3452	6876	13724	27420
R_DSP_FIR_i16i16_asm_st			909	1789	3549	7069	14109	28189
R_DSP_FIR_i16i16_asm_s2			917	1805	3581	7133	14237	28445

Taps=64, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	1556	3068	6092	12140	24236	48428
		NEAREST	1569	3097	6153	12265	24489	48937
	SATURATE	TRUNC	1594	3146	6250	12458	24874	49706
		NEAREST	1601	3161	6281	12521	25001	49961
R_DSP_FIR_i16i16_asm_nt	-		1542	3054	6078	12126	24222	48414
R_DSP_FIR_i16i16_asm_n2			1556	3084	6140	12252	24476	48924
R_DSP_FIR_i16i16_asm_st			1581	3133	6237	12445	24861	49693
R_DSP_FIR_i16i16_asm_s2			1589	3149	6269	12509	24989	49949

Taps=128, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	2900	5756	11468	22892	45740	91436
		NEAREST	2913	5785	11529	23017	45993	91945
	SATURATE	TRUNC	2938	5834	11626	23210	46378	92714
		NEAREST	2945	5849	11657	23273	46505	92969
R_DSP_FIR_i16i16_asm_nt	-		2886	5742	11454	22878	45726	91422
R_DSP_FIR_i16i16_asm_n2			2900	5772	11516	23004	45980	91932
R_DSP_FIR_i16i16_asm_st			2925	5821	11613	23197	46365	92701
R_DSP_FIR_i16i16_asm_s2			2933	5837	11645	23261	46493	92957

Taps=256, Scale=15

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i16	NO SATURATE	TRUNC	5588	11132	22220	44396	88748	177452
		NEAREST	5601	11161	22281	44521	89001	177961
	SATURATE	TRUNC	5626	11210	22378	44714	89386	178730
		NEAREST	5633	11225	22409	44777	89513	178985
R_DSP_FIR_i16i16_asm_nt	-		5574	11118	22206	44382	88734	177438
R_DSP_FIR_i16i16_asm_n2			5588	11148	22268	44508	88988	177948
R_DSP_FIR_i16i16_asm_st			5613	11197	22365	44701	89373	178717
R_DSP_FIR_i16i16_asm_s2			5621	11213	22397	44765	89501	178973

(2) **R_DSP_FIR_i16i32**

Taps=16, Scale=0

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	568	1088	2128	4208	8368	16688
		NEAREST	590	1134	2222	4398	8750	17454
	SATURATE	TRUNC	623	1199	2351	4655	9263	18479
		NEAREST	622	1198	2350	4654	9262	18478
R_DSP_FIR_i16i32_asm_nt	-		553	1073	2113	4193	8353	16673
R_DSP_FIR_i16i32_asm_n2			576	1120	2208	4384	8736	17440
R_DSP_FIR_i16i32_asm_st			609	1185	2337	4641	9249	18465
R_DSP_FIR_i16i32_asm_s2			609	1185	2337	4641	9249	18465

Taps=32, Scale=0

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	904	1760	3472	6896	13744	27440
		NEAREST	926	1806	3566	7086	14126	28206
	SATURATE	TRUNC	959	1871	3695	7343	14639	29231
		NEAREST	958	1870	3694	7342	14638	29230
R_DSP_FIR_i16i32_asm_nt	-		889	1745	3457	6881	13729	27425
R_DSP_FIR_i16i32_asm_n2			912	1792	3552	7072	14112	28192
R_DSP_FIR_i16i32_asm_st			945	1857	3681	7329	14625	29217
R_DSP_FIR_i16i32_asm_s2			945	1857	3681	7329	14625	29217

Taps=64, Scale=0

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	1576	3104	6160	12272	24496	48944
		NEAREST	1598	3150	6254	12462	24878	49710
	SATURATE	TRUNC	1631	3215	6383	12719	25391	50735
		NEAREST	1630	3214	6382	12718	25390	50734
R_DSP_FIR_i16i32_asm_nt	-		1561	3089	6145	12257	24481	48929
R_DSP_FIR_i16i32_asm_n2			1584	3136	6240	12448	24864	49696
R_DSP_FIR_i16i32_asm_st			1617	3201	6369	12705	25377	50721
R_DSP_FIR_i16i32_asm_s2			1617	3201	6369	12705	25377	50721

Taps=128, Scale=0

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	2920	5792	11536	23024	46000	91952
		NEAREST	2942	5838	11630	23214	46382	92718
	SATURATE	TRUNC	2975	5903	11759	23471	46895	93743
		NEAREST	2974	5902	11758	23470	46894	93742
R_DSP_FIR_i16i32_asm_nt	-		2905	5777	11521	23009	45985	91937
R_DSP_FIR_i16i32_asm_n2			2928	5824	11616	23200	46368	92704
R_DSP_FIR_i16i32_asm_st			2961	5889	11745	23457	46881	93729
R_DSP_FIR_i16i32_asm_s2			2961	5889	11745	23457	46881	93729

Taps=256, Scale=0

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i16i32	NO SATURATE	TRUNC	5608	11168	22288	44528	89008	177968
		NEAREST	5630	11214	22382	44718	89390	178734
	SATURATE	TRUNC	5663	11279	22511	44975	89903	179759
		NEAREST	5662	11278	22510	44974	89902	179758
R_DSP_FIR_i16i32_asm_nt	-		5593	11153	22273	44513	88993	177953
R_DSP_FIR_i16i32_asm_n2			5616	11200	22368	44704	89376	178720
R_DSP_FIR_i16i32_asm_st			5649	11265	22497	44961	89889	179745
R_DSP_FIR_i16i32_asm_s2			5649	11265	22497	44961	89889	179745

(3) **R_DSP_FIR_i32i32**

Taps=16, Scale=31

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	784	1520	2992	5936	11824	23600
		NEAREST	791	1535	3023	5999	11951	23855
	SATURATE	TRUNC	847	1647	3247	6447	12847	25647
		NEAREST	862	1678	3310	6574	13102	26158
R_DSP_FIR_i32i32_asm_nt	-		770	1506	2978	5922	11810	23586
R_DSP_FIR_i32i32_asm_n2			778	1522	3010	5986	11938	23842
R_DSP_FIR_i32i32_asm_st			834	1634	3234	6434	12834	25634
R_DSP_FIR_i32i32_asm_s2			850	1666	3298	6562	13090	26146

Taps=32, Scale=31

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	1296	2544	5040	10032	20016	39984
		NEAREST	1303	2559	5071	10095	20143	40239
	SATURATE	TRUNC	1359	2671	5295	10543	21039	42031
		NEAREST	1374	2702	5358	10670	21294	42542
R_DSP_FIR_i32i32_asm_nt	-		1282	2530	5026	10018	20002	39970
R_DSP_FIR_i32i32_asm_n2			1290	2546	5058	10082	20130	40226
R_DSP_FIR_i32i32_asm_st			1346	2658	5282	10530	21026	42018
R_DSP_FIR_i32i32_asm_s2			1362	2690	5346	10658	21282	42530

Taps=64, Scale=31

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	2320	4592	9136	18224	36400	72752
		NEAREST	2327	4607	9167	18287	36527	73007
	SATURATE	TRUNC	2383	4719	9391	18735	37423	74799
		NEAREST	2398	4750	9454	18862	37678	75310
R_DSP_FIR_i32i32_asm_nt	-		2306	4578	9122	18210	36386	72738
R_DSP_FIR_i32i32_asm_n2			2314	4594	9154	18274	36514	72994
R_DSP_FIR_i32i32_asm_st			2370	4706	9378	18722	37410	74786
R_DSP_FIR_i32i32_asm_s2			2386	4738	9442	18850	37666	75298

Taps=128, Scale=31

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	4368	8688	17328	34608	69168	138288
		NEAREST	4375	8703	17359	34671	69295	138543
	SATURATE	TRUNC	4431	8815	17583	35119	70191	140335
		NEAREST	4446	8846	17646	35246	70446	140846
R_DSP_FIR_i32i32_asm_nt	-		4354	8674	17314	34594	69154	138274
R_DSP_FIR_i32i32_asm_n2			4362	8690	17346	34658	69282	138530
R_DSP_FIR_i32i32_asm_st			4418	8802	17570	35106	70178	140322
R_DSP_FIR_i32i32_asm_s2			4434	8834	17634	35234	70434	140834

Taps=256, Scale=31

Function name	option		Samples					
	SATURATE	ROUNDING	8	16	32	64	128	256
R_DSP_FIR_i32i32	NO SATURATE	TRUNC	8464	16880	33712	67376	134704	269360
		NEAREST	8471	16895	33743	67439	134831	269615
	SATURATE	TRUNC	8527	17007	33967	67887	135727	271407
		NEAREST	8542	17038	34030	68014	135982	271918
R_DSP_FIR_i32i32_asm_nt	-		8450	16866	33698	67362	134690	269346
R_DSP_FIR_i32i32_asm_n2			8458	16882	33730	67426	134818	269602
R_DSP_FIR_i32i32_asm_st			8514	16994	33954	67874	135714	271394
R_DSP_FIR_i32i32_asm_s2			8530	17026	34018	68002	135970	271906

(4) **R_DSP_FIR_f32f32**

Scale=1.0f

Function name	Taps	Samples					
		8	16	32	64	128	256
R_DSP_FIR_f32f32	16	1514	2994	5954	11874	23714	47394
R_DSP_FIR_f32f32_asm		1511	2991	5951	11871	23711	47391
R_DSP_FIR_f32f32	32	2922	5810	11586	23138	46242	92450
R_DSP_FIR_f32f32_asm		2919	5807	11583	23135	46239	92447
R_DSP_FIR_f32f32	64	5738	11442	22850	45666	91298	182562
R_DSP_FIR_f32f32_asm		5735	11439	22847	45663	91295	182559
R_DSP_FIR_f32f32	128	11370	22706	45378	90722	181410	362786
R_DSP_FIR_f32f32_asm		11367	22703	45375	90719	181407	362783
R_DSP_FIR_f32f32	256	22634	45234	90434	180834	361634	723234
R_DSP_FIR_f32f32_asm		22631	45231	90431	180831	361631	723231

4.1.2 Biquad IIR Filter

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, Scale=14

Function name	option		Samples								
	SATURATE	ROUNDING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i16	NO SATURATE	TRUNC	127	175	271	463	847	1615	3151	6223	12367
		NEAREST	129	179	279	479	879	1679	3279	6479	12879
	SATURATE	TRUNC	138	195	309	537	993	1905	3729	7377	14673
		NEAREST	138	197	315	551	1023	1967	3855	7631	15183
R_DSP_IIRBiquad_i16i16_asm_nt	-		113	161	257	449	833	1601	3137	6209	12353
R_DSP_IIRBiquad_i16i16_asm_n2			116	166	266	466	866	1666	3266	6466	12866
R_DSP_IIRBiquad_i16i16_asm_st			125	182	296	524	980	1892	3716	7364	14660
R_DSP_IIRBiquad_i16i16_asm_s2			126	185	303	539	1011	1955	3843	7619	15171

stages=2, Scale=14

Function name	option		Samples								
	SATURATE	ROUNDING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i16	NO SATURATE	TRUNC	179	268	446	802	1514	2938	5786	11482	22874
		NEAREST	181	273	457	825	1561	3033	5977	11865	23641
	SATURATE	TRUNC	193	295	499	907	1723	3355	6619	13147	26203
		NEAREST	194	299	509	929	1769	3449	6809	13529	26969
R_DSP_IIRBiquad_i16i16_asm_nt	-		165	254	432	788	1500	2924	5772	11468	22860
R_DSP_IIRBiquad_i16i16_asm_n2			168	260	444	812	1548	3020	5964	11852	23628
R_DSP_IIRBiquad_i16i16_asm_st			180	282	486	894	1710	3342	6606	13134	26190
R_DSP_IIRBiquad_i16i16_asm_s2			182	287	497	917	1757	3437	6797	13517	26957

stages=4, Scale=14

Function name	option		Samples								
	SATURATE	ROUNDING	1	2	4	8	16	32	64	128	256
R_DSP_IIRBiquad_i16i16	NO SATURATE	TRUNC	289	460	802	1486	2854	5590	11062	22006	43894
		NEAREST	293	469	821	1525	2933	5749	11381	22645	45173
	SATURATE	TRUNC	311	503	887	1655	3191	6263	12407	24695	49271
		NEAREST	316	513	907	1695	3271	6423	12727	25335	50551
R_DSP_IIRBiquad_i16i16_asm_nt	-		275	446	788	1472	2840	5576	11048	21992	43880
R_DSP_IIRBiquad_i16i16_asm_n2			280	456	808	1512	2920	5736	11368	22632	45160
R_DSP_IIRBiquad_i16i16_asm_st			298	490	874	1642	3178	6250	12394	24682	49258
R_DSP_IIRBiquad_i16i16_asm_s2			304	501	895	1683	3259	6411	12715	25323	50539

(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	142	193	301	517	949	1813	3541	6997	13909
			-1	143	196	308	532	980	1876	3668	7252	14420
			1	145	199	313	541	997	1909	3733	7381	14677
		NEAREST	0	142	194	304	524	964	1844	3604	7124	14164
			-1	145	201	319	555	1027	1971	3859	7635	15187
			1	146	202	320	556	1028	1972	3860	7636	15188
	SATURATE	TRUNC	0	152	212	339	593	1101	2117	4149	8213	16341
			-1	155	219	354	624	1164	2244	4404	8724	17364
			1	153	214	343	601	1117	2149	4213	8341	16597
		NEAREST	0	152	214	344	604	1124	2164	4244	8404	16724
			-1	156	223	363	643	1203	2323	4563	9043	18003
			1	155	220	356	628	1172	2260	4436	8788	17492
R_DSP_IIRBiquad_i16i32_asm_nt	-		0	128	179	287	503	935	1799	3527	6983	13895
R_DSP_IIRBiquad_i16i32_asm_n2			0	129	181	291	511	951	1831	3591	7111	14151
R_DSP_IIRBiquad_i16i32_asm_st			0	139	199	326	580	1088	2104	4136	8200	16328
R_DSP_IIRBiquad_i16i32_asm_s2			0	140	202	332	592	1112	2152	4232	8392	16712

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	193	286	478	862	1630	3166	6238	12382	24670
			-1	194	289	485	877	1661	3229	6365	12637	25181
			1	196	292	490	886	1678	3262	6430	12766	25438
		NEAREST	0	194	288	482	870	1646	3198	6302	12510	24926
			-1	197	295	497	901	1709	3325	6557	13021	25949
			1	198	296	498	902	1710	3326	6558	13022	25950
	SATURATE	TRUNC	0	206	311	528	962	1830	3566	7038	13982	27870
			-1	209	318	543	993	1893	3693	7293	14493	28893
			1	207	313	532	970	1846	3598	7102	14110	28126
		NEAREST	0	209	318	542	990	1886	3678	7262	14430	28766
			-1	213	327	561	1029	1965	3837	7581	15069	30045
			1	212	324	554	1014	1934	3774	7454	14814	29534
R_DSP_IIRBiquad_i16i32_asm_nt	-		0	179	272	464	848	1616	3152	6224	12368	24656
R_DSP_IIRBiquad_i16i32_asm_n2			0	181	275	469	857	1633	3185	6289	12497	24913
R_DSP_IIRBiquad_i16i32_asm_st			0	193	298	515	949	1817	3553	7025	13969	27857
R_DSP_IIRBiquad_i16i32_asm_s2			0	197	306	530	978	1874	3666	7250	14418	28754

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	301	478	838	1558	2998	5878	11638	23158	46198
			-1	302	481	845	1573	3029	5941	11765	23413	46709
			1	304	484	850	1582	3046	5974	11830	23542	46966
		NEAREST	0	304	482	844	1568	3016	5912	11704	23288	46456
			-1	307	489	859	1599	3079	6039	11959	23799	47479
			1	308	490	860	1600	3080	6040	11960	23800	47480
	SATURATE	TRUNC	0	322	517	914	1708	3296	6472	12824	25528	50936
			-1	325	524	929	1739	3359	6599	13079	26039	51959
			1	323	519	918	1716	3312	6504	12888	25656	51192
		NEAREST	0	329	532	944	1768	3416	6712	13304	26488	52856
			-1	333	541	963	1807	3495	6871	13623	27127	54135
			1	332	538	956	1792	3464	6808	13496	26872	53624
R_DSP_IIRBiquad_i16i32_asm_nt	-		0	287	464	824	1544	2984	5864	11624	23144	46184
R_DSP_IIRBiquad_i16i32_asm_n2			0	291	469	831	1555	3003	5899	11691	23275	46443
R_DSP_IIRBiquad_i16i32_asm_st			0	309	504	901	1695	3283	6459	12811	25515	50923
R_DSP_IIRBiquad_i16i32_asm_s2			0	317	520	932	1756	3404	6700	13292	26476	52844

(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	149	212	338	590	1094	2102	4118	8150	16214
			29	149	212	338	590	1094	2102	4118	8150	16214
			31	149	212	338	590	1094	2102	4118	8150	16214
		NEAREST	30	150	215	345	605	1125	2165	4245	8405	16725
			29	150	215	345	605	1125	2165	4245	8405	16725
			31	150	215	345	605	1125	2165	4245	8405	16725
	SATURATE	TRUNC	30	165	247	411	739	1395	2707	5331	10579	21075
			29	165	247	411	739	1395	2707	5331	10579	21075
			31	165	247	411	739	1395	2707	5331	10579	21075
		NEAREST	30	167	252	422	762	1442	2802	5522	10954	21810
			29	167	252	422	746	1394	2690	5286	10486	20866
			31	167	252	422	762	1442	2802	5522	10958	21826
R_DSP_IIRBiquad_i32i32_asm_nt	-		30	135	198	324	576	1080	2088	4104	8136	16200
R_DSP_IIRBiquad_i32i32_asm_n2			30	137	202	332	592	1112	2152	4232	8392	16712
R_DSP_IIRBiquad_i32i32_asm_st			30	152	234	398	726	1382	2694	5318	10566	21062
R_DSP_IIRBiquad_i32i32_asm_s2			30	155	240	410	750	1430	2790	5510	10942	21798

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	207	317	537	977	1857	3617	7137	14177	28257
			29	207	317	537	977	1857	3617	7137	14177	28257
			31	207	317	537	977	1857	3617	7137	14177	28257
		NEAREST	30	208	321	547	999	1903	3711	7327	14559	29023
			29	208	321	547	999	1903	3711	7327	14559	29023
			31	208	321	547	999	1903	3711	7327	14559	29023
	SATURATE	TRUNC	30	231	368	642	1190	2286	4478	8862	17630	35166
			29	231	368	642	1190	2286	4478	8862	17630	35166
			31	231	368	642	1190	2286	4478	8862	17630	35166
		NEAREST	30	235	377	661	1229	2365	4637	9181	18249	36373
			29	235	377	661	1229	2333	4541	8957	17805	35489
			31	235	377	661	1229	2365	4637	9181	18257	36401
R_DSP_IIRBiquad_i32i32_asm_nt	-		30	193	303	523	963	1843	3603	7123	14163	28243
R_DSP_IIRBiquad_i32i32_asm_n2			30	195	308	534	986	1890	3698	7314	14546	29010
R_DSP_IIRBiquad_i32i32_asm_st			30	218	355	629	1177	2273	4465	8849	17617	35153
R_DSP_IIRBiquad_i32i32_asm_s2			30	223	365	649	1217	2353	4625	9169	18237	36361

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	331	535	943	1759	3391	6655	13183	26239	52351
			29	331	535	943	1759	3391	6655	13183	26239	52351
			31	331	535	943	1759	3391	6655	13183	26239	52351
		NEAREST	30	332	541	959	1795	3467	6811	13499	26875	53627
			29	332	541	959	1795	3467	6811	13499	26875	53627
			31	332	541	959	1795	3467	6811	13499	26875	53627
	SATURATE	TRUNC	30	369	616	1110	2098	4074	8026	15930	31738	63354
			29	369	616	1110	2098	4074	8026	15930	31738	63354
			31	369	616	1110	2098	4074	8026	15930	31738	63354
		NEAREST	30	377	633	1145	2169	4217	8313	16505	32853	65541
			29	377	633	1145	2169	4213	8245	16309	32433	64685
			31	377	633	1145	2169	4217	8313	16505	32861	65565
R_DSP_IIRBiquad_i32i32_asm_nt	-		30	317	521	929	1745	3377	6641	13169	26225	52337
R_DSP_IIRBiquad_i32i32_asm_n2			30	319	528	946	1782	3454	6798	13486	26862	53614
R_DSP_IIRBiquad_i32i32_asm_st			30	356	603	1097	2085	4061	8013	15917	31725	63341
R_DSP_IIRBiquad_i32i32_asm_s2			30	365	621	1133	2157	4205	8301	16493	32841	65529

(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	105	151	243	427	795	1531	3003	5947	11835
R_DSP_IIRBiquad_f32f32_asm		102	148	240	424	792	1528	3000	5944	11832
R_DSP_IIRBiquad_f32f32	2	160	249	427	783	1495	2919	5767	11463	22855
R_DSP_IIRBiquad_f32f32_asm		157	246	424	780	1492	2916	5764	11460	22852
R_DSP_IIRBiquad_f32f32	4	278	453	803	1503	2903	5703	11303	22503	44903
R_DSP_IIRBiquad_f32f32_asm		275	450	800	1500	2900	5700	11300	22500	44900

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	-	-	838	2075	4928	11701	26690
				SC	865	2163	5039	12071	27187
				X2	848	2114	4913	11737	26388
			TW32	-	1303	3466	8673	21038	49269
				SC	1336	3549	8803	21356	49746
				X2	1339	3529	8827	21285	49851
		R_cfft_ci16ci16	-		823	2060	4913	11686	26675
		R_cfft_sc_ci16ci16			850	2148	5024	12056	27172
		R_cfft_x2_ci16ci16			837	2103	4902	11726	26377
		R_cfft_tw32_ci16ci16			1285	3448	8655	21020	49251
		R_cfft_sc_tw32_ci16ci16			1320	3533	8787	21340	49730
		R_cfft_x2_tw32_ci16ci16			1326	3516	8814	21272	49838
	ci32	R_DSP_FFT_ci16ci32	-	-	1450	3761	9463	22612	53178
				SC	1504	3922	9768	23426	54728
				X2	1476	3805	9528	22689	53208
			TW32	-	1466	3780	9534	22696	53470
				SC	1513	3939	9789	23463	54749
				X2	1478	3802	9518	22662	53150
		R_cfft_ci16ci32	-		1436	3747	9449	22598	53164
		R_cfft_sc_ci16ci32			1490	3908	9754	23412	54714
		R_cfft_x2_ci16ci32			1466	3795	9518	22679	53198
		R_cfft_tw32_ci16ci32			1450	3764	9518	22680	53454
		R_cfft_sc_tw32_ci16ci32			1497	3923	9773	23447	54733
		R_cfft_x2_tw32_ci16ci32			1466	3790	9506	22650	53138
ci32	ci32	R_DSP_FFT_ci32ci32	-	-	1417	3700	9355	22408	52815
				SC	1466	3845	9602	23087	53992
				X2	1432	3731	9354	22407	52558
		R_cfft_ci32ci32	-		1406	3689	9344	22397	52804
		R_cfft_sc_ci32ci32			1456	3835	9592	23077	53982
		R_cfft_x2_ci32ci32			1421	3720	9343	22396	52547
cf32	cf32	R_DSP_FFT_cf32cf32	-		1467	3700	8942	21189	48675
		R_cfft_cf32cf32	-		1462	3695	8937	21184	48670

points = 512 - 8192

in	out	Function name	option		points				
			TW32	Scale	512	1024	2048	4096	8192
ci16	ci16	R_DSP_FFT_ci16ci16	-	-	61087	135468	302169	657510	1440435
				SC	62679	137699	309023	667435	1469847
				X2	60728	133251	298687	644554	1418294
			TW32	-	113630	256229	573222	1262381	2768286
				SC	114803	257913	577450	1268144	2783233
				X2	114525	258339	576341	1269851	2778797
		R_cfft_ci16ci16	-		61072	135453	302154	657495	1440420
		R_cfft_sc_ci16ci16			62664	137684	309008	667420	1469832
		R_cfft_x2_ci16ci16			60717	133240	298676	644543	1418283
		R_cfft_tw32_ci16ci16			113612	256211	573204	1262363	2768268
		R_cfft_sc_tw32_ci16ci16			114787	257897	577434	1268128	2783217
		R_cfft_x2_tw32_ci16ci16			114512	258326	576328	1269838	2778784
	ci32	R_DSP_FFT_ci16ci32	-	-	121411	274761	610122	1347920	2938689
				SC	125322	282256	628354	1383048	3021914
				X2	121241	273888	607441	1340376	2919849
			TW32	-	121760	275942	611544	1352670	2944432
				SC	125343	282085	627927	1381341	3018159
				X2	121118	273638	606934	1339358	2917806
		R_cfft_ci16ci32	-		121397	274747	610108	1347906	2938675
		R_cfft_sc_ci16ci32			125308	282242	628340	1383034	3021900
		R_cfft_x2_ci16ci32			121231	273878	607431	1340366	2919839
		R_cfft_tw32_ci16ci32			121744	275926	611528	1352654	2944416
		R_cfft_sc_tw32_ci16ci32			125327	282069	627911	1381325	3018143
		R_cfft_x2_tw32_ci16ci32			121106	273626	606922	1339346	2917794
ci32	ci32	R_DSP_FFT_ci32ci32	-	-	120728	273567	607904	1344167	2931864
				SC	123825	279034	621819	1369092	2993653
				X2	120215	271518	603807	1331878	2907287
		R_cfft_ci32ci32	-		120717	273556	607893	1344156	2931853
		R_cfft_sc_ci32ci32			123815	279024	621809	1369082	2993643
		R_cfft_x2_ci32ci32			120204	271507	603796	1331867	2907276
cf32	cf32	R_DSP_FFT_cf32cf32	-		110886	247028	548591	1198077	2614760
		R_cfft_cf32cf32	-		110881	247023	548586	1198072	2614755

(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	-	-	983	2359	5481	12803	28877
				SC	1018	2461	5625	13218	29454
				X2	1004	2407	5522	12891	28806
			TW32	-	1454	3745	9245	22106	51474
				SC	1487	3846	9385	22518	52045
				X2	1478	3797	9353	22326	51918
		R_icfft_ci16ci16	-		969	2345	5467	12789	28863
		R_icfft_sc_ci16ci16			1004	2447	5611	13204	29440
		R_icfft_x2_ci16ci16			994	2397	5512	12881	28796
		R_icfft_tw32_ci16ci16			1438	3729	9229	22090	51458
		R_icfft_sc_tw32_ci16ci16			1471	3830	9369	22502	52029
		R_icfft_x2_tw32_ci16ci16			1466	3785	9341	22314	51906
ci32		R_DSP_IFFT_ci32ci16	-	-	1067	2529	5829	13517	30329
				SC	1093	2615	5939	13867	30775
				X2	1082	2563	5842	13527	30098
			TW32	-	1580	4001	9892	23405	54596
				SC	1602	4072	9964	23672	54864
				X2	1573	3988	9801	23222	53967
		R_icfft_ci32ci16	-		1053	2515	5815	13503	30315
		R_icfft_sc_ci32ci16			1080	2602	5926	13854	30762
		R_icfft_x2_ci32ci16			1071	2552	5831	13516	30087
		R_icfft_tw32_ci32ci16			1564	3985	9876	23389	54580
		R_icfft_sc_tw32_ci32ci16			1587	4057	9949	23657	54849
		R_icfft_x2_tw32_ci32ci16			1560	3975	9788	23209	53954
	ci32	R_DSP_IFFT_ci32ci32	-	-	1569	3980	9929	23478	55023
				SC	1620	4167	10222	24395	56530
				X2	1584	4011	9928	23477	54766
		R_icfft_ci32ci32	-		1558	3969	9918	23467	55012
		R_icfft_sc_ci32ci32			1610	4157	10212	24385	56520
		R_icfft_x2_ci32ci32			1573	4000	9917	23466	54755
cf32	cf32	R_DSP_IFFT_cf32cf32	-		1699	4129	9797	22813	51977
		R_icfft_cf32cf32	-		1693	4123	9791	22807	51971

points = 512 - 8192

in	out	Function name	option		points				
			TW32	Scale	512	1024	2048	4096	8192
ci16	ci16	R_DSP_IFFT_ci16ci16	-	-	65459	144189	319611	692357	1510131
				SC	67123	146495	326268	701576	1536757
				X2	65323	142902	317043	683134	1491691
			TW32	-	117755	264771	589188	1295500	2830077
				SC	119414	267069	595838	1304709	2856694
				X2	118647	266559	592768	1302664	2844409
		R_icfft_ci16ci16	-		65445	144175	319597	692343	1510117
		R_icfft_sc_ci16ci16			67109	146481	326254	701562	1536743
		R_icfft_x2_ci16ci16			65313	142892	317033	683124	1491681
		R_icfft_tw32_ci16ci16			117739	264755	589172	1295484	2830061
		R_icfft_sc_tw32_ci16ci16			119398	267053	595822	1304693	2856678
		R_icfft_x2_tw32_ci16ci16			118635	266547	592756	1302652	2844397
ci32		R_DSP_IFFT_ci32ci16	-	-	68445	150249	332069	717617	1562013
				SC	69851	152039	337699	724783	1584539
				X2	67887	148090	327335	703922	1532927
			TW32	-	124005	279340	618333	1362020	2963125
				SC	125048	280384	622464	1366152	2979576
				X2	122744	275777	611202	1343627	2926332
		R_icfft_ci32ci16	-		68431	150235	332055	717603	1561999
		R_icfft_sc_ci32ci16			69838	152026	337686	724770	1584526
		R_icfft_x2_ci32ci16			67876	148079	327324	703911	1532916
		R_icfft_tw32_ci32ci16			123989	279324	618317	1362004	2963109
		R_icfft_sc_tw32_ci32ci16			125033	280369	622449	1366137	2979561
		R_icfft_x2_tw32_ci32ci16			122731	275764	611189	1343614	2926319
	ci32	R_DSP_IFFT_ci32ci32	-	-	124856	282113	623874	1377291	2993660
				SC	129179	289442	643747	1411754	3083419
				X2	124343	280064	619777	1365002	2969083
		R_icfft_ci32ci32	-		124845	282102	623863	1377280	2993649
		R_icfft_sc_ci32ci32			129169	289432	643737	1411744	3083409
		R_icfft_x2_ci32ci32			124332	280053	619766	1364991	2969072
cf32	cf32	R_DSP_IFFT_cf32cf32	-		117197	259929	573269	1248609	2711373
		R_icfft_cf32cf32	-		117191	259923	573263	1248603	2711367

(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	-	-	587	1252	2915	6615	15088
				SC	613	1292	3028	6786	15558
				X2	601	1267	2950	6625	15114
			TW32	-	971	2317	5572	13000	29717
				SC	1002	2359	5705	13185	30285
				X2	985	2356	5656	13180	30084
		R_rfft_i16ci16	-		573	1238	2901	6601	15074
		R_rfft_sc_i16ci16			600	1279	3015	6773	15545
		R_rfft_x2_i16ci16			590	1256	2939	6614	15103
		R_rfft_tw32_i16ci16			955	2301	5556	12984	29701
		R_rfft_sc_tw32_i16ci16			987	2344	5690	13170	30270
		R_rfft_x2_tw32_i16ci16			972	2343	5643	13167	30071
	ci32	R_DSP_FFT_i16ci32	-	-	1021	2467	5889	13816	31412
				SC	1051	2520	6042	14087	32147
				X2	1032	2485	5911	13828	31376
			TW32	-	1020	2470	5876	13817	31349
				SC	1051	2514	6035	14071	32130
				X2	1032	2479	5904	13812	31359
		R_rfft_i16ci32	-		1005	2451	5873	13800	31396
		R_rfft_sc_i16ci32			1037	2506	6028	14073	32133
		R_rfft_x2_i16ci32			1021	2474	5900	13817	31365
		R_rfft_tw32_i16ci32			1001	2451	5857	13798	31330
		R_rfft_sc_tw32_i16ci32			1034	2497	6018	14054	32113
		R_rfft_x2_tw32_i16ci32			1017	2464	5889	13797	31344
i32	ci32	R_DSP_FFT_i32ci32	-	-	1002	2439	5841	13737	31269
				SC	1033	2486	6000	13992	32036
				X2	1017	2454	5872	13736	31268
		R_rfft_i32ci32	-		991	2428	5830	13726	31258
		R_rfft_sc_i32ci32			1022	2475	5989	13981	32025
		R_rfft_x2_i32ci32			1007	2444	5862	13726	31258
f32	cf32	R_DSP_FFT_f32cf32	-		904	2073	4947	11469	26277
		R_rfft_f32cf32	-		899	2068	4942	11464	26272

points = 512 - 8192

in	out	Function name	option		points				
			TW32	Scale	512	1024	2048	4096	8192
i16	ci16	R_DSP_FFT_i16ci16	-	-	33468	74657	162605	356458	766070
				SC	34164	76560	165406	364106	777304
				X2	33269	74266	160805	352866	754797
			TW32	-	66797	148502	326366	712479	1542183
				SC	67605	150909	329861	722629	1557197
				X2	67548	150012	329412	718580	1554428
		R_rfft_i16ci16	-		33454	74643	162591	356444	766056
		R_rfft_sc_i16ci16			34151	76547	165393	364093	777291
		R_rfft_x2_i16ci16			33258	74255	160794	352855	754786
		R_rfft_tw32_i16ci16			66781	148486	326350	712463	1542167
		R_rfft_sc_tw32_i16ci16			67590	150894	329846	722614	1557182
		R_rfft_x2_tw32_i16ci16			67535	149999	329399	718567	1554415
	ci32	R_DSP_FFT_i16ci32	-	-	70875	156899	345834	752362	1632497
				SC	72204	160340	352157	768157	1661862
				X2	70665	156241	343962	747674	1620899
			TW32	-	70862	156630	345759	751263	1632168
				SC	72154	160289	351977	767976	1661168
				X2	70615	156190	343782	747493	1620205
		R_rfft_i16ci32	-		70859	156883	345818	752346	1632481
		R_rfft_sc_i16ci32			72190	160326	352143	768143	1661848
		R_rfft_x2_i16ci32			70654	156230	343951	747663	1620888
		R_rfft_tw32_i16ci32			70843	156611	345740	751244	1632149
		R_rfft_sc_tw32_i16ci32			72137	160272	351960	767959	1661151
		R_rfft_x2_tw32_i16ci32			70600	156175	343767	747478	1620190
i32	ci32	R_DSP_FFT_i32ci32	-	-	70637	156469	345149	751165	1630789
				SC	71916	160052	351292	767548	1659460
				X2	70380	155956	343100	747068	1618500
		R_rfft_i32ci32	-		70626	156458	345138	751154	1630778
		R_rfft_sc_i32ci32			71905	160041	351281	767537	1659449
		R_rfft_x2_i32ci32			70370	155946	343090	747058	1618490
f32	cf32	R_DSP_FFT_f32cf32	-		58883	131335	287957	630481	1361887
		R_rfft_f32cf32	-		58878	131330	287952	630476	1361882

(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	-	-	647	1368	3159	7079	16074
				SC	668	1386	3211	7144	16287
				X2	656	1371	3149	7018	15890
			TW32	-	945	2229	5359	12542	28762
				SC	968	2266	5470	12707	29227
				X2	955	2253	5411	12648	28980
		R_irfft_ci16i16	-		631	1352	3143	7063	16058
		R_irfft_sc_ci16i16			654	1372	3197	7130	16273
		R_irfft_x2_ci16i16			645	1360	3138	7007	15879
		R_irfft_tw32_ci16i16			927	2211	5341	12524	28744
		R_irfft_sc_tw32_ci16i16			952	2250	5454	12691	29211
		R_irfft_x2_tw32_ci16i16			942	2240	5398	12635	28967
ci32		R_DSP_IFFT_CCS_ci32i16	-	-	688	1445	3310	7374	16661
				SC	708	1469	3390	7470	16981
				X2	696	1454	3316	7352	16560
			TW32	-	990	2347	5613	13157	30065
				SC	1009	2376	5686	13258	30326
				X2	999	2362	5630	13154	29994
		R_irfft_ci32i16	-		673	1430	3295	7359	16646
		R_irfft_sc_ci32i16			693	1454	3375	7455	16966
		R_irfft_x2_ci32i16			685	1443	3305	7341	16549
		R_irfft_tw32_ci32i16			972	2329	5595	13139	30047
		R_irfft_sc_tw32_ci32i16			991	2358	5668	13240	30308
		R_irfft_x2_tw32_ci32i16			985	2348	5616	13140	29980
	i32	R_DSP_IFFT_CCS_ci32i32	-	-	977	2365	5645	13338	30370
				SC	1008	2412	5804	13593	31137
				X2	993	2381	5677	13338	30370
		R_irfft_ci32i32	-		966	2354	5634	13327	30359
		R_irfft_sc_ci32i32			998	2402	5794	13583	31127
		R_irfft_x2_ci32i32			981	2369	5665	13326	30358
cf32	f32	R_DSP_IFFT_CCS_cf32f32	-		1042	2345	5462	12507	28274
		R_irfft_cf32f32	-		1038	2341	5458	12503	28270

points = 512 - 8192

in	out	Function name	option		points				
			TW32	Scale	512	1024	2048	4096	8192
ci16	i16	R_DSP_IFFT_CCS_ci16i16	-	-	35334	78625	170093	372376	796068
				SC	35628	79567	171484	376599	802596
				X2	34815	77347	166960	365164	779385
			TW32	-	64865	144585	318544	696720	1510807
				SC	65540	146468	321261	704285	1521702
				X2	65293	145461	320254	700222	1517639
		R_irfft_ci16i16	-		35318	78609	170077	372360	796052
		R_irfft_sc_ci16i16			35614	79553	171470	376585	802582
		R_irfft_x2_ci16i16			34804	77336	166949	365153	779374
		R_irfft_tw32_ci16i16			64847	144567	318526	696702	1510789
		R_irfft_sc_tw32_ci16i16			65524	146452	321245	704269	1521686
		R_irfft_x2_tw32_ci16i16			65280	145448	320241	700209	1517626
ci32		R_DSP_IFFT_CCS_ci32i16	-	-	36497	80948	174720	381627	814535
				SC	36881	82228	176256	386747	820679
				X2	36156	80032	172332	375912	800884
			TW32	-	67897	150945	332969	726761	1577713
				SC	68238	151862	334078	729918	1581126
				X2	67586	150050	330474	720666	1562658
		R_irfft_ci32i16	-		36482	80933	174705	381612	814520
		R_irfft_sc_ci32i16			36866	82213	176241	386732	820664
		R_irfft_x2_ci32i16			36145	80021	172321	375901	800873
		R_irfft_tw32_ci32i16			67879	150927	332951	726743	1577695
		R_irfft_sc_tw32_ci32i16			68220	151844	334060	729900	1581108
		R_irfft_x2_tw32_ci32i16			67572	150036	330460	720652	1562644
	i32	R_DSP_IFFT_CCS_ci32i32	-	-	68891	152667	337828	735380	1600413
				SC	70170	156250	343971	751763	1629084
				X2	68635	152155	335780	731284	1588125
		R_irfft_ci32i32	-		68880	152656	337817	735369	1600402
		R_irfft_sc_ci32i32			70160	156240	343961	751753	1629074
		R_irfft_x2_ci32i32			68623	152143	335768	731272	1588113
cf32	f32	R_DSP_IFFT_CCS_cf32f32	-		62943	139170	303919	661290	1424695
		R_irfft_cf32f32	-		62939	139166	303915	661286	1424691

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	Dec 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