RX Family RXv2 CPU Products

An example of C-language program to use DSP instructions

R01AN3956EJ0100
Rev.1.00
Oct. 02, 2017

Introduction

This document shows definitions of RXv2 DSP instructions as assembly-language inline functions to use in C-language program.

This document is intended for users who have knowledge of digital signal processing and make digital signal processing programs in C language.

Target Device

RX Family, RXv2 CPU products

Contents

1. Assembly-language inline functions of DSP instruction

RXv2 CPU has DSP instruction which executes 16bit or 32bit fixed point multiply-add calculation, saturation and rounding in one cycle. DSP instruction operates fast and keeping precision with overflow margin for multiply-add and multiply-subtract using 72bit accumulator (ACC0, ACC1).

This document declares DSP instructions as assembly-language inline functions to use them in C-language program, and defines macro functions (DSP inline function) which are collected the same operation of assembly-language inline functions.

2. Definition of DSP inline function

This section shows DSP inline functions declared in r_dsp_inst_rxv2.h in Table 1. And it describes them by classification of operations with assembly-language inline functions. Each DSP inline function is associated with an assembly-language instruction. DSP inline functions used in “An usage example” is indicated as bold in Table 1.

Refer to “RX Family RXv2 Instruction Set Architecture User’s Manual: Software (R01US0071)” for each assembly instruction in detail.

NOTE: Specify ACC1 for DSP inline function to avoid conflict ACC0.
In case of using ACC0, confirm the two items below between calculation by DSP inline functions, with list file or something.

1. Confirm the following instructions which use ACC0 are not existed.
- EMUL
- EMULU
- FMUL
- MUL
- RMPA

2. Confirm the unintentional instructions are not existed. DSP instructions MULLO and MACLO which are used ACC0 could be created in complier parameters (“- speed” and “-save_acc”).

Table 1 List of DSP inline functions

operation classification	Function Name	Description
Multiply, Multiply-add, Multiply-subtract	**__emula**	**32bit multiply macro function**
	__emaca	**32bit multiply-add macro function**
	__emsba	32bit multiply-sub macro function
	__mulhi	16bit multiply macro function, upper 16bit x upper 16bit
	__mullh	16bit multiply macro function, 16bit x upper 16bit
	__mullo	16bit multiply macro function, 16bit x 16bit
	__machi	16bit multiply-add macro function, upper 16bit x upper 16bit
	__maclh	16bit multiply-add macro function, 16bit x upper 16bit
	__maclo	16bit multiply-add macro function, 16bit x 16bit
	__msbhi	16bit multiply-sub macro function, upper 16bit x upper 16bit
	__msblh	16bit multiply-sub macro function, 16bit x upper 16bit
	__msblo	16bit multiply-sub macro function, 16bit x 16bit
Saturation, Rounding	**__racl**	**32bit saturation and rounding macro function**
	__rdacl	32bit saturation and truncation macro function
	__racw	16bit saturation and rounding macro function
	__rdacw	16bit saturation and truncation macro function
Reading accumulator	**__mvfachi**	**Reading upper 32bit of accumulator macro functions**
	__mvfacmi	Reading middle-order 32bit of accumulator macro functions
	__mvfaclo	Reading lower 32bit of accumulator macro functions
	__mvfacgu	Reading accumulator guard bit macro functions
Writing accumulator	__mvtachi	Writing upper 32bit of accumulator macro functions
	__mvtaclo	Writing lower 32bit of accumulator macro functions
	__mvtacgu	Writing accumulator Guard bit macro functions

2.1 Multiply, Multiply-add, Multiply-subtract

__emula: 32bit multiply

Format

```
void __emula(int32_t src, int32_t src2, int adest);
```

Parameters

src:	32bit fixed point multiplicand
src2:	32bit fixed point multiplier
adest:	Assignment of an accumulator to store result (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 32bit x 32bit, then stores 64bit result to the assigned accumulator by LSB alignment.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __emula_a0(int32_t src, int32_t src2)`	`EMULA R1,R2,A0`
1	`void __emula_a1(int32_t src, int32_t src2)`	`EMULA R1,R2,A1`

__emaca: 32bit multiply-add

Format

```
void __emaca(int32_t src, int32_t src2, int adest);
```

Parameters

src:	32bit fixed point multiplicand
src2:	32bit fixed point multiplier
adest:	Assignment of an accumulator to add the product (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 32bit x 32bit, then add 64bit result to the assigned accumulator by LSB alignment.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __emaca_a0(int32_t src, int32_t src2)`	`EMACA R1,R2,A0`
1	`void __emaca_a1(int32_t src, int32_t src2)`	`EMACA R1,R2,A1`

__emsba: 32bit multiply-subtract

Format

```
void __emsba(int32_t src, int32_t src2, int adest);
```

Parameters

src:	32bit fixed point multiplicand
src2:	32bit fixed point multiplier
adest:	Assignment of an accumulator to subtract the product (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 32bit x 32bit, then subtract 64bit result from the assigned accumulator by LSB alignment.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __emsba_a0(int32_t src, int32_t src2)`	`EMSBA R1,R2,A0`
1	`void __emsba_a1(int32_t src, int32_t src2)`	`EMSBA R1,R2,A1`

__mulhi: 16bit multiply, upper 16bit x upper 16bit

Format

```
void __mulhi(int32_t src, int32_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand in upper 16bit
src2:	16bit fixed point multiplier in upper 16bit
adest:	Assignment of an accumulator to store result (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then stores 32bit result to the assigned accumulator. The multiplication targets are upper 16bit part of *src* and *src2*.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __mulhi_a0(int32_t src, int32_t src2)`	`MULHI R1,R2,A0`
1	`void __mulhi_a1(int32_t src, int32_t src2)`	`MULHI R1,R2,A1`

__mullh: 16bit multiply, 16bit x upper 16bit

Format

```
void __mullh(int16_t src, int32_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand
src2:	16bit fixed point multiplier in upper 16bit
adest:	Assignment of an accumulator to store result (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then stores 32bit result to the assigned accumulator. The multiplication target of *src2* is upper 16bit part.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __mullh_a0(int16_t src, int32_t src2)`	`MULLH  R1,R2,A0`
1	`void __mullh_a1(int16_t src, int32_t src2)`	`MULLH  R1,R2,A1`

__mullo: 16bit multiply, 16bit x 16bit

Format

```
void __mullo(int16_t src, int16_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand
src2:	16bit fixed point multiplier
adest:	Assignment of an accumulator to store result (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then stores 32bit result to assigned accumulator.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __mullo_a0(int16_t src, int16_t src2)`	`MULLO  R1,R2,A0`
1	`void __mullo_a1(int16_t src, int16_t src2)`	`MULLO  R1,R2,A1`

__machi: 16bit multiply-add, upper 16bit x upper 16bit

Format

```
void __machi(int32_t src, int32_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand in upper 16bit
src2:	16bit fixed point multiplier in upper 16bit
adest:	Assignment of an accumulator to add the product (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then add 32bit result to the assigned accumulator. The multiplication targets are upper 16bit part of *src* and *src2*.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void _machi_a0(int32_t src, int32_t src2)`	`MACHI R1,R2,A0`
1	`void _machi_a1(int32_t src, int32_t src2)`	`MACHI R1,R2,A1`

__maclh: 16bit multiply-add, 16bit x upper 16bit

Format

```
void __maclh(int16_t src, int32_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand
src2:	16bit fixed point multiplier in upper 16bit
adest:	Assignment of an accumulator to add the product (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then add 32bit result to the assigned accumulator. The multiplication target of *src2* is upper 16bit part.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __maclh_a0(int16_t src, int32_t src2)`	`MACLH R1,R2,A0`
1	`void __maclh_a1(int16_t src, int32_t src2)`	`MACLH R1,R2,A1`

__maclo: 16bit multiply-add, 16bit x 16bit

Format

```
void __maclo(int32_t src, int32_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand
src2:	16bit fixed point multiplier
adest:	Assignment of an accumulator to subtract the product (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then add 32bit result to the assigned accumulator.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __maclo_a0(int16_t src, int16_t src2)`	`MACLO  R1,R2,A0`
1	`void __maclo_a1(int16_t src, int16_t src2)`	`MACLO  R1,R2,A1`

__msbhi: 16bit multiply- subtract, upper 16bit x upper 16bit

Format

```
void __msbhi(int32_t src, int32_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand in upper 16bit
src2:	16bit fixed point multiplier in upper 16bit
adest:	Assignment of an accumulator to subtract the product (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then subtract 32bit result from the assigned accumulator. The multiplication targets are upper 16bit part of *src* and *src2*.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __msbhi_a0(int32_t src, int32_t src2)`	`MSBHI  R1,R2,A0`
1	`void __msbhi_a1(int32_t src, int32_t src2)`	`MSBHI  R1,R2,A1`

__msblh: 16bit multiply- subtract, 16bit x upper 16bit

Format

```
void __msblh(int16_t src, int32_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand
src2:	16bit fixed point multiplier in upper 16bit
adest:	Assignment of an accumulator to subtract the product (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then subtract 32bit result from the assigned accumulator. The multiplication target of *src2* is upper 16bit part.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __msblh_a0(int16_t src, int32_t src2)`	`MSBLH R1,R2,A0`
1	`void __msblh_a1(int16_t src, int32_t src2)`	`MSBLH R1,R2,A1`

__msblo: 16bit multiply-subtract, 16bit x 16bit

Format

```
void __msblo(int16_t src, int16_t src2, int adest);
```

Parameters

src:	16bit fixed point multiplicand
src2:	16bit fixed point multiplier
adest:	Assignment of an accumulator to subtract the product (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function calculates product of 16bit x 16bit, then subtract 32bit result from the assigned accumulator.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __msblo_a0(int16_t src, int16_t src2)`	`MSBLO R1,R2,A0`
1	`void __msblo_a1(int16_t src, int16_t src2)`	`MSBLO R1,R2,A1`

2.2 Saturation, Rounding

__racl: 32bit saturation and rounding

Format

```
void __racl(int shift, int asrc);
```

Parameters

shift: bit count of shift-left (1: 1bit, 2: 2bit). This parameter should be specified by immediate value.

asrc: Assignment of an accumulator to operate (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function shifts the assigned accumulator value to the left by *shift*, then stores the same accumulator by MSB alignment as 32bit value by saturation and rounding.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *shift* and *asrc* value.

shift	asrc	assembly-language inline function	corresponding assembly instruction
1	0	`void __racl_s1_a0(void)`	`RACL #1,A0`
2	0	`void __racl_s2_a0(void)`	`RACL #2,A0`
1	1	`void __racl_s1_a1(void)`	`RACL #1,A1`
2	1	`void __racl_s2_a1(void)`	`RACL #2,A1`

__rdacl: 32bit saturation and truncation

Format

```
void __rdacl(int shift, int asrc);
```

Parameters

shift: bit count of shift-left (1: 1bit, 2: 2bit). This parameter should be specified by immediate value.

asrc: Assignment of an accumulator to operate (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function shifts the assigned accumulator value to the left by *shift*, then stores the same accumulator by MSB alignment as 32bit value by saturation and truncation.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *shift* and *asrc* value.

shift	asrc	assembly-language inline function	corresponding assembly instruction
1	0	`void __rdacl_s1_a0(void)`	`RDACL #1,A0`
2	0	`void __rdacl_s2_a0(void)`	`RDACL #2,A0`
1	1	`void __rdacl_s1_a1(void)`	`RDACL #1,A1`
2	1	`void __rdacl_s2_a1(void)`	`RDACL #2,A1`

__racw: 16bit saturation and rounding

Format

```
void __racw(int shift, int asrc);
```

Parameters

shift: bit count of shift-left (1: 1bit, 2: 2bit). This parameter should be specified by immediate value.

asrc: Assignment of an accumulator to operate (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function shifts the accumulator value to the left by *shift*, then stores the same accumulator as 16bit value by saturation and rounding.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *shift* and *asrc* value.

shift	asrc	assembly-language inline function	corresponding assembly instruction
1	0	`void __racw_s1_a0(void)`	`RACW #1,A0`
2	0	`void __racw_s2_a0(void)`	`RACW #2,A0`
1	1	`void __racw_s1_a1(void)`	`RACW #1,A1`
2	1	`void __racw_s2_a1(void)`	`RACW #2,A1`

__rdacw: 16bit saturation and truncation

Format

```
void __rdacw(int shift, int asrc);
```

Parameters

shift: bit count of shift-left (1: 1bit, 2: 2bit). This parameter should be specified by immediate value.

asrc: Assignment of an accumulator to operate (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function shifts the assigned accumulator value by *shift*, then store the same accumulator as 16bit value by saturation and truncation.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *shift* and *asrc* value.

shift	asrc	assembly-language inline function	corresponding assembly instruction
1	0	`void __rdacw_s1_a0(void)`	`RDACW #1,A0`
2	0	`void __rdacw_s2_a0(void)`	`RDACW #2,A0`
1	1	`void __rdacw_s1_a1(void)`	`RDACW #1,A1`
2	1	`void __rdacw_s2_a1(void)`	`RDACW #2,A1`

2.3 Reading accumulator

__mvfachi: Reading upper 32bit of accumulator

Format

```
int32_t __mvfachi(int shift, int asrc);
```

Parameters

shift: bit count of shift-left (0: 0bit, 1: 1bit, 2: 2bit). This parameter should be specified by immediate value.

asrc: Assignment of an accumulator to read (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

Upper 32bit value of accumulator

Description

This function shifts the assigned accumulator value to the left by *shift*, then returns upper 32bit value for getting 32bit calculation result.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *shift* and *asrc* value.

shift	asrc	assembly-language inline function	corresponding assembly instruction
0	0	`int32_t __mvfachi_s0_a0 (void)`	`MVFACHI  #0,A0`
1	0	`int32_t __mvfachi_s1_a0 (void)`	`MVFACHI  #1,A0`
2	0	`int32_t __mvfachi_s2_a0 (void)`	`MVFACHI  #2,A0`
0	1	`int32_t __mvfachi_s0_a1 (void)`	`MVFACHI  #0,A1`
1	1	`int32_t __mvfachi_s1_a1 (void)`	`MVFACHI  #1,A1`
2	1	`int32_t __mvfachi_s2_a1 (void)`	`MVFACHI  #2,A1`

__mvfacmi: Reading middle-order 32bit of accumulator

Format

```
int32_t __mvfacmi(int shift, int asrc);
```

Parameters

shift: bit count of shift-left (0: 0bit, 1: 1bit, 2: 2bit). This parameter should be specified by immediate value.

asrc: Assignment of an accumulator to read (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

middle-order 32bit value of accumulator.

Description

This function operates shifts the assigned accumulator value to the left by *shift*, then returns middle-order 32bit value for getting 16bit calculation result as LSB alignment.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *shift* and *asrc* value.

shift	asrc	assembly-language inline function	corresponding assembly instruction
0	0	int32_t __mvfacmi_s0_a0 (void)	MVFACMI #0,A0
1	0	int32_t __mvfacmi_s1_a0 (void)	MVFACMI #1,A0
2	0	int32_t __mvfacmi_s2_a0 (void)	MVFACMI #2,A0
0	1	int32_t __mvfacmi_s0_a1 (void)	MVFACMI #0,A1
1	1	int32_t __mvfacmi_s1_a1 (void)	MVFACMI #1,A1
2	1	int32_t __mvfacmi_s2_a1 (void)	MVFACMI #2,A1

__mvfaclo: Reading lower 32bit of accumulator

Format

```
uint32_t __mvfaclo(int shift, int asrc);
```

Parameters

shift: bit count of shift-left (0: 0bit, 1: 1bit, 2: 2bit). This parameter should be specified by immediate value.

asrc: Assignment of an accumulator to read (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

Lower 32bit value of accumulator

Description

This function shifts the assigned accumulator value to the left by *shift*, then returns lower 32bit value for getting 32bit calculation result.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *shift* and *asrc* value.

shift	asrc	assembly-language inline function	corresponding assembly instruction
0	0	uint32_t __mvfaclo_s0_a0 (void)	MVFACLO #0,A0
1	0	uint32_t __mvfaclo_s1_a0 (void)	MVFACLO #1,A0
2	0	uint32_t __mvfaclo_s2_a0 (void)	MVFACLO #2,A0
0	1	uint32_t __mvfaclo_s0_a1 (void)	MVFACLO #0,A1
1	1	uint32_t __mvfaclo_s1_a1 (void)	MVFACLO #1,A1
2	1	uint32_t __mvfaclo_s2_a1 (void)	MVFACLO #2,A1

__mvfacgu: Reading accumulator guard bit

Format

```
uint32_t __mvfacgu(int shift, int asrc);
```

Parameters

shift: bit count of shift-left (0: 0bit, 1: 1bit, 2: 2bit). This parameter should be specified by immediate value.

asrc: Assignment of an accumulator to read (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

Accumulator guard bit value as 32bit LSB alignment.

Description

This function shifts the assigned accumulator value to the left by *shift*, then returns accumulator guard bit value as LSB aligned 32bit value.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on the *shift* and *asrc* value.

shift	asrc	assembly-language inline function	corresponding assembly instruction
0	0	uint32_t __mvfaclo_s0_a0 (void)	MVFACLO #0,A0
1	0	uint32_t __mvfaclo_s1_a0 (void)	MVFACLO #1,A0
2	0	uint32_t __mvfaclo_s2_a0 (void)	MVFACLO #2,A0
0	1	uint32_t __mvfaclo_s0_a1 (void)	MVFACLO #0,A1
1	1	uint32_t __mvfaclo_s1_a1 (void)	MVFACLO #1,A1
2	1	uint32_t __mvfaclo_s2_a1 (void)	MVFACLO #2,A1

2.4 Writing accumulator

__mvtachi: Writing upper 32bit of accumulator

Format

```
void __mvtachi(int32_t src, int adest);
```

Parameters

src: writing value to upper 32bit of accumulator.
adest: Assignment of an accumulator to write (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function writes *src* value to upper 32bit of accumulator

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __mvtachi_a0 (int32_t src)`	`MVTACHI R1,A0`
1	`void __mvtachi_a1 (int32_t src)`	`MVTACHI R1,A1`

__mvtaclo: Writing lower 32bit of accumulator

Format

```
void __mvtaclo(int32_t src, int adest);
```

Parameters

src: writing value to lower 32bit of accumulator.
adest: Assignment of an accumulator to write (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

This function writes *src* value to lower 32bit of accumulator

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __mvtaclo_a0 (uint32_t src)`	`MVTACLO R1,A0`
1	`void __mvtaclo_a1 (uint32_t src)`	`MVTACLO R1,A1`

__mvtacgu: Writing accumulator guard bit

Format

```
void __mvtacgu(uint32_t src, int adest);
```

Parameters

src:	writing value to accumulator guard bit.
adest:	Assignment of an accumulator to write (0: ACC0, 1: ACC1). This parameter should be specified by immediate value.

Return Value

none.

Description

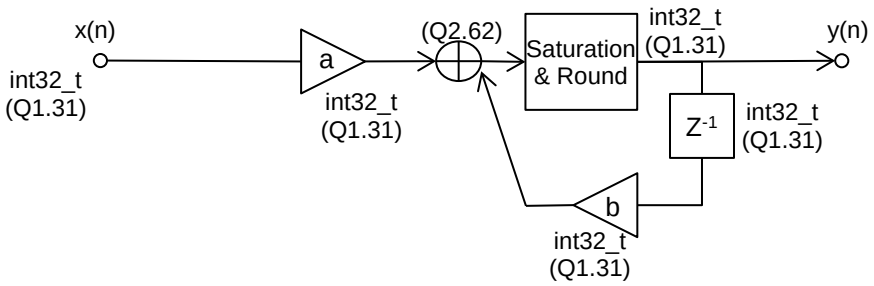
This function writes *src* value to accumulator guard bit as LSB aligned.

This function is a macro definition function, and replaced by one of following assembly-language inline functions depending on *adest* value.

adest	assembly-language inline function	corresponding assembly instruction
0	`void __mvtacgu_a0 (uint32_t src)`	`MVTACGU R1,A0`
1	`void __mvtacgu_a1 (uint32_t src)`	`MVTACGU R1,A1`

3. An usage example

An example of a program using DSP inline functions are shown below using the single pole IIR filter shown in Figure 1.



$y(n) = a_0x(n) + b_1y(n-1)$

Note: Qm.nn represents the fixed point format.
m: bit count of sign and integer part
nn: bit count of fractional part

Figure 1 Signal Flow Chart of Single-pole IIR filter

```
#include "r_dsp_inst_rxv2.h"

int32_t singlepoleiir(int32_t input, int32_t coeff[2], int32_t *delay)
{
  __emula(coeff[0], input, 1);    // acc1  = a * x(n)
  __emaca(coeff[1], *delay, 1);   // acc1 += b * y(n-1)
  __racl(1, 1);                   // saturation, rounding and MSB alignment
  *delay = __mvfachi(0, 1);       // extract filter output

  return *delay;
}
```

4. Reference documents

- RX Family RXv2 Instruction Set Architecture User's Manual: Software (R01US0071)
- CC-RX Compiler User’s Manual (R20UT3248)

The latest version can be downloaded from the Renesas Electronics website.

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	Oct. 02, 2017	-	First issue

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 disputes 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, drawing, chart, program, algorithm, 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 otherwise misappropriate 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, copy or otherwise misappropriation of Renesas Electronics products.
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; and 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.

   Renesas Electronics products are neither intended nor 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 damages (space and 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 third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the 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 radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond 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. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of 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 by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate 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. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use 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 not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
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 majority-owned subsidiaries.

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

(Rev.3.0-1 November 2016)

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

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

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

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: +44-1628-585-100, Fax: +44-1628-585-900

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, P. R. China 200333
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 II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 6.0