RX ファミリ RXv2 CPU 搭載製品
C 言語で DSP 機能命令を使用するプログラム例

R01AN3956JJ0100
Rev.1.00
2017.10.02

要旨

本アプリケーションノートでは、RXv2 の DSP 機能命令を C 言語で使用できるようにアセンブラ埋め込みインライン関数として定義します。

本アプリケーションノートは、デジタル信号処理の知識を持ち、C 言語でデジタル信号処理プログラムを作成するユーザを対象とします。

対象デバイス

RX ファミリ RXv2 CPU 搭載製品

目次

1. DSP 機能命令のアセンブラ埋め込みインライン関数

RXv2 CPU には 16bit および 32bit の固定小数点積和演算と演算結果の飽和・丸め処理を 1 サイクルで実行する DSP 機能命令があります。DSP 機能命令は 72bit アキュムレータ(ACC0、ACC1)を使用して、乗算結果である 32bit や 64bit に対して、積和・積差演算のオーバーフローマージンを確保することで、精度を維持しながら高速に演算します。

本アプリケーションノートでは、C 言語から DSP 機能命令を利用するために、アセンブリ命令に対応するアセンブラ埋め込みインライン関数を定義し、アセンブラ埋め込みインライン関数をアセンブリ命令ごとにまとめてマクロ定義関数（以降 DSP インライン関数）として定義します。

2. DSP インライン関数定義

本章では、サンプルファイルの r_dsp_inst_rxv2.h で定義する DSP インライン関数の一覧を表 1 に示し、以降機能毎にアセンブラ埋め込みインライン関数と共に説明します。表 1 では、コーディング例に使用する関数を太字で示します。

各アセンブリ命令の詳細は「RX ファミリ RXv2 命令セットアーキテクチャ ユーザーズマニュアル ソフトウェア編 (R01US0071)」を参照ください。

【注】 DSP インライン関数は、ACC0 の競合を避けるために ACC1 を使用してください。
ACC0 を使用する場合、DSP インライン関数を使用する演算の開始から終了までの間に下記 2 点をリストファイル等で確認してください。

1. DSP 機能命令以外の次の ACC0 を使用する命令がないこと。
- EMUL
- EMULU
- FMUL
- MUL
- RMPA

2. コンパイル条件(-speed かつ -save_acc)により、16bit の乗算、積和演算に対して、ACC0 を使用する DSP 機能命令の MULLO, MACLO が生成される場合がある。左記 2 命令がないこと。

表 1 DSP インライン関数一覧

機能	関数名	機能
乗算・積和・積差	**__emula**	**32bit 乗算マクロ定義関数**
	__emaca	**32bit 積和マクロ定義関数**
	__emsba	32bit 積差マクロ定義関数
	__mulhi	16bit 乗算マクロ定義関数、上位 16bit x 上位 16bit
	__mullh	16bit 乗算マクロ定義関数、16bit x 上位 16bit
	__mullo	16bit 乗算マクロ定義関数、16bit x 16bit
	__machi	16bit 積和マクロ定義関数、上位 16bit x 上位 16bit
	__maclh	16bit 積和マクロ定義関数、16bit x 上位 16bit
	__maclo	16bit 積和マクロ定義関数、16bit x 16bit
	__msbhi	16bit 積差マクロ定義関数、上位 16bit x 上位 16bit
	__msblh	16bit 積差マクロ定義関数、16bit x 上位 16bit
	__msblo	16bit 積差マクロ定義関数、16bit x 16bit
飽和・丸め	**__racl**	**32bit 飽和と丸めマクロ定義関数**
	__rdacl	32bit 飽和と切り捨てマクロ定義関数
	__racw	16bit 飽和と丸めマクロ定義関数
	__rdacw	16bit 飽和と切り捨てマクロ定義関数
アキュムレータ読み出し	**__mvfachi**	**アキュムレータ上位 32bit 読み出しマクロ定義関数**
	__mvfacmi	アキュムレータ中央 32bit 読み出しマクロ定義関数
	__mvfaclo	アキュムレータ下位 32bit 読み出しマクロ定義関数
	__mvfacgu	アキュムレータガードビット読み出しマクロ定義関数
アキュムレータ書き込み	__mvtachi	アキュムレータ上位 32bit 書き込みマクロ定義関数
	__mvtaclo	アキュムレータ下位 32bit 書き込みマクロ定義関数
	__mvtacgu	アキュムレータガードビット書き込みマクロ定義関数

2.1 乗算・積和・積差

__emula: 32bit 乗算

Format

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

Parameters

src:	32bit 固定小数点の被乗数
src2:	32bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、32bit x 32bit の乗算を行い、64bit の結果を、指定したアキュムレータに LSB 詰めで格納します。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 積和

Format

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

Parameters

src:	32bit 固定小数点の被乗数
src2:	32bit 固定小数点の乗数
adest:	結果を加算するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、32bit x 32bit の乗算を行い、64bit の結果を、指定したアキュムレータに LSB 詰めで加算します。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 積差

Format

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

Parameters

src:	32bit 固定小数点の被乗数
src2:	32bit 固定小数点の乗数
adest:	結果を減算するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、32bit x 32bit の乗算を行い、64bit の結果を、指定したアキュムレータから LSB 詰めで減算します。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 乗算 上位 16bit x 上位 16bit

Format

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

Parameters

src:	上位 16bit に 16bit 固定小数点の被乗数
src2:	上位 16bit に 16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータに格納します。src および src2 の上位 16bit を乗算対象とします。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 乗算 16bit x 上位 16bit

Format

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

Parameters

src:	16bit 固定小数点の被乗数
src2:	上位 16bit に 16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータに格納します。src2 は上位 16bit を乗算対象とします。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 乗算 16bit x 16bit

Format

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

Parameters

src:	16bit 固定小数点の被乗数
src2:	16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータに格納します。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 積和 上位 16bit x 上位 16bit

Format

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

Parameters

src:	上位 16bit に 16bit 固定小数点の被乗数
src2:	上位 16bit に 16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータに加算します。src および src2 の上位 16bit を乗算対象とします。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 積和 16bit x 上位 16bit

Format

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

Parameters

src:	16bit 固定小数点の被乗数
src2:	上位 16bit に 16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータに加算します。src2 は上位 16bit を乗算対象とします。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 積和 16bit x 16bit

Format

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

Parameters

src:	16bit 固定小数点の被乗数
src2:	16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータに加算します。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 積差 上位 16bit x 上位 16bit

Format

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

Parameters

src:	上位 16bit に 16bit 固定小数点の被乗数
src2:	上位 16bit に 16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータから減算します。src および src2 の上位 16bit を乗算対象とします。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 積差 16bit x 上位 16bit

Format

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

Parameters

src:	16bit 固定小数点の被乗数
src2:	上位 16bit に 16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータから減算します。src2 は上位 16bit を乗算対象とします。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 積差 16bit x 16bit

Format

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

Parameters

src:	16bit 固定小数点の被乗数
src2:	16bit 固定小数点の乗数
adest:	結果を格納するアキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、16bit x 16bit の乗算を行い、32bit の結果を、指定したアキュムレータから減算します。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 飽和・丸め

__racl: 32bit 飽和と丸め

Format

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

Parameters

shift: アキュムレータ値の左シフト量指定 (1: 1bit, 2: 2bit)。本パラメータは即値で指定してください。

asrc: 操作対象アキュムレータ指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、アキュムレータの値を shift で指定した左シフトを行い、飽和処理と丸めにより 32bit 値にして MSB 詰めでアキュムレータに格納します。

本関数はマクロ定義関数です。shift と asrc の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

shift	asrc	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 飽和と切り捨て

Format

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

Parameters

shift: アキュムレータ値の左シフト量指定 (1: 1bit, 2: 2bit)。本パラメータは即値で指定してください。

asrc: 操作対象アキュムレータ指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、アキュムレータの値を shift で指定した左シフトを行い、飽和処理と切り捨てにより 32bit 値にして MSB 詰めでアキュムレータに格納します。

本関数はマクロ定義関数です。shift と asrc の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

shift	asrc	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 飽和と丸め

Format

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

Parameters

shift: アキュムレータ値の左シフト量指定 (1: 1bit, 2: 2bit)。本パラメータは即値で指定してください。

asrc: 操作対象アキュムレータ指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、アキュムレータの値を shift で指定した左シフトを行い、飽和処理と丸めにより 16bit 値にしてアキュムレータに格納します。

本関数はマクロ定義関数です。shift と asrc の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

shift	asrc	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 飽和と切り捨て

Format

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

Parameters

shift: アキュムレータ値の左シフト量指定 (1: 1bit, 2: 2bit)。本パラメータは即値で指定してください。

asrc: 操作対象アキュムレータ指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、アキュムレータの値を shift で指定した左シフトを行い、飽和処理と切り捨てにより 16bit 値にしてアキュムレータに格納します。

本関数はマクロ定義関数です。shift と asrc の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

shift	asrc	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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 アキュムレータ読み出し

__mvfachi: アキュムレータ上位 32 ビットの読み出し

Format

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

Parameters

shift: アキュムレータ値の左シフト量指定 (0: 0bit, 1: 1bit, 2: 2bit)。本パラメータは即値で指定してください。

asrc: 操作対象アキュムレータ指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

アキュムレータの上位 32bit 値。

Description

本関数は、アキュムレータの値を shift で指定した左シフト結果の上位 32bit を返します。32bit 演算の結果を取得するのに使用します。

本関数はマクロ定義関数です。shift と asrc の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

shift	asrc	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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: アキュムレータ中央 32bit の読み出し

Format

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

Parameters

shift: アキュムレータ値の左シフト量の指定 (0: 0bit, 1: 1bit, 2: 2bit)。本パラメータは即値で指定してください。

asrc: 操作対象アキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

アキュムレータの中央 32bit 値。

Description

本関数は、アキュムレータの値を shift で指定した左シフト結果の中央 32bit を返します。16bit 演算の結果を LSB 詰めで取得するのに使用します。

本関数はマクロ定義関数です。shift と asrc の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

shift	asrc	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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: アキュムレータ下位 32bit の読み出し

Format

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

Parameters

shift: アキュムレータ値の左シフト量の指定 (0: 0bit, 1: 1bit, 2: 2bit)。本パラメータは即値で指定してください。

asrc: 操作対象アキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

アキュムレータの下位 32bit 値。

Description

本関数は、アキュムレータの値を shift で指定した左シフト結果の下位 32bit を返します。32bit 演算の結果の下位 32bit を取得するのに使用します。

本関数はマクロ定義関数です。shift と asrc の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

shift	asrc	アセンブラ埋め込みインライン関数	対応するアセンブリコード
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: アキュムレータガードビット読み出し

Format

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

Parameters

shift: アキュムレータ値の左シフト量の指定 (0: 0bit, 1: 1bit, 2: 2bit)。本パラメータは即値で指定してください。

asrc: 操作対象アキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

アキュムレータガードビットを LSB 詰めで 32bit 値。

Description

本関数は、アキュムレータガードビットの値を指定の量の左シフトした結果の符号情報 LSB 詰め 32bit で返します。

本関数はマクロ定義関数です。shift と asrc の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

shift	asrc	アセンブラ埋め込みインライン関数	対応するアセンブリコード
0	0	`uint32_t __mvfacgu_s0_a0 (void)`	`MVFACGU #0,A0`
1	0	`uint32_t __mvfacgu_s1_a0 (void)`	`MVFACGU #1,A0`
2	0	`uint32_t __mvfacgu_s2_a0 (void)`	`MVFACGU #2,A0`
0	1	`uint32_t __mvfacgu_s0_a1 (void)`	`MVFACGU #0,A1`
1	1	`uint32_t __mvfacgu_s1_a1 (void)`	`MVFACGU #1,A1`
2	1	`uint32_t __mvfacgu_s2_a1 (void)`	`MVFACGU #2,A1`

2.4 アキュムレータ書き込み

__mvtachi: アキュムレータ上位 32bit 書き込み

Format

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

Parameters

src:	アキュムレータの上位 32bit に書き込む値。
adest:	書き込み対象アキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、アキュムレータの上位 32bit に src の値を書き込みます。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
0	`void __mvtachi_a0 (int32_t src)`	`MVTACHI R1,A0`
1	`void __mvtachi_a1 (int32_t src)`	`MVTACHI R1,A1`

__mvtaclo: アキュムレータ下位 32bit 書き込み

Format

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

Parameters

src:	アキュムレータの下位 32bit に書き込む値。
adest:	書き込み対象アキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

本関数は、アキュムレータの下位 32bit に src の値を書き込みます。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
0	`void __mvtaclo_a0 (uint32_t src)`	`MVTACLO R1,A0`
1	`void __mvtaclo_a1 (uint32_t src)`	`MVTACLO R1,A1`

__mvtacgu: アキュムレータガードビット書き込み

Format

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

Parameters

src:	アキュムレータガードビットに書き込む値。
adest:	書き込み対象アキュムレータの指定 (0: ACC0, 1: ACC1)。本パラメータは即値で指定してください。

Return Value

なし。

Description

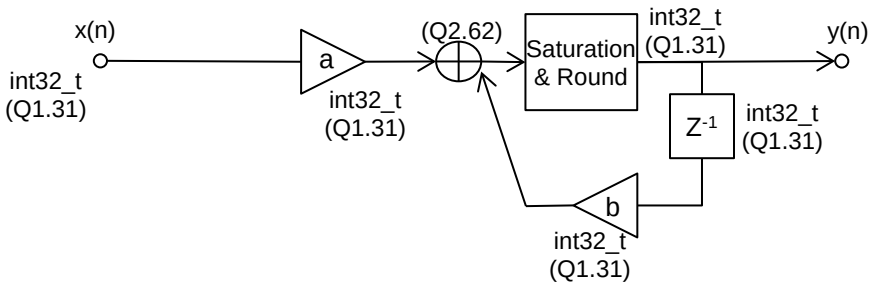
本関数は、アキュムレータガードビットに src の値を LSB 詰めで書き込みます。

本関数はマクロ定義関数です。adest の指定により次のいずれかのアセンブラ埋め込みインライン関数に置換されます。

adest	アセンブラ埋め込みインライン関数	対応するアセンブリコード
0	`void __mvtacgu_a0 (uint32_t src)`	`MVTACGU R1,A0`
1	`void __mvtacgu_a1 (uint32_t src)`	`MVTACGU R1,A1`

3. コーディング例

図 1 に示す単極 IIR フィルタを例に DSP インライン関数を使用したプログラム例を示します。



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

【注】Qm.nn は固定小数点の形式を示します。
m: 符号および整数部のビット数
nn: 小数部のビット数

図 1 単極 IIR フィルタ シグナルフロー

```
#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. 参考資料

- RX ファミリ RXv2 命令セットアーキテクチャ ユーザーズマニュアル　ソフトウェア編 (R01US0071)
- CC-RX コンパイラ ユーザーズマニュアル (R20UT3248)

最新版をルネサス エレクトロニクスホームページから入手してください。

ホームページとサポート窓口

ルネサス エレクトロニクスホームページ
http://japan.renesas.com/

お問合せ先
http://japan.renesas.com/contact/

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	2017.10.02	-	初版

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 未使用端子の処理
   【注意】未使用端子は、本文の「未使用端子の処理」に従って処理してください。
   CMOS製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI周辺のノイズが印加され、LSI内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。未使用端子は、本文「未使用端子の処理」で説明する指示に従い処理してください。
2. 電源投入時の処置
   【注意】電源投入時は，製品の状態は不定です。
   電源投入時には、LSIの内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。
   外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。
   同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. リザーブアドレス（予約領域）のアクセス禁止
   【注意】リザーブアドレス（予約領域）のアクセスを禁止します。
   アドレス領域には、将来の機能拡張用に割り付けられているリザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
4. クロックについて
   【注意】リセット時は、クロックが安定した後、リセットを解除してください。
   プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。
   リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子
   （または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
5. 製品間の相違について
   【注意】型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。
   同じグループのマイコンでも型名が違うと、内部ROM、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ輻射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器・システムの設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品、本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を、全部または一部を問わず、改造、改変、複製、その他の不適切に使用しないでください。かかる改造、改変、複製等により生じた損害に関し、当社は、一切その責任を負いません。
5. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA機器、通信機器、計測機器、AV機器、
   家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：　輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、
   金融端末基幹システム、各種安全制御装置等

   当社製品は、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することはできません。たとえ、意図しない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
6. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
8. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制するRoHS指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
9. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。また、当社製品および技術を、(1)核兵器、化学兵器、生物兵器等の大量破壊兵器およびこれらを運搬することができるミサイル（無人航空機を含みます。）の開発、設計、製造、使用もしくは貯蔵等の目的、(2)通常兵器の開発、設計、製造または使用の目的、または(3)その他の国際的な平和および安全の維持の妨げとなる目的で、自ら使用せず、かつ、第三者に使用、販売、譲渡、輸出、賃貸もしくは使用許諾しないでください。
   当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
10. お客様の転売、貸与等により、本書（本ご注意書きを含みます。）記載の諸条件に抵触して当社製品が使用され、その使用から損害が生じた場合、当社は一切その責任を負わず、お客様にかかる使用に基づく当社への請求につき当社を免責いただきます。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
12. 本資料に記載された情報または当社製品に関し、ご不明点がある場合には、当社営業にお問い合わせください。

注1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注2. 本資料において使用されている「当社製品」とは、注1において定義された当社の開発、製造製品をいいます。

(Rev.3.0-1　2016.11)

ルネサス エレクトロニクス株式会社

■営業お問合せ窓口　http://www.renesas.com

※営業お問合せ窓口の住所は変更になることがあります。最新情報につきましては、弊社ホームページをご覧ください。

ルネサス エレクトロニクス株式会社　〒135-0061　東京都江東区豊洲3-2-24（豊洲フォレシア）

■技術的なお問合せおよび資料のご請求は下記へどうぞ。
総合お問合せ窓口：https://www.renesas.com/contact/