

本説明書は V850ES/Kx1、V850ES/Kx1+用の浮動小数点演算のサンプルソフトウェアの動作を説明したものです。

ご注意

本ソフトウェアはあくまで参考用のソフトであり、当社がこの動作を保証するものではありません。

本ソフトウェアを使用する場合、お客様のセット上で十分な評価の上ご使用いただきますようお願いいたします。

目次

浮動小数点演算機能関数説明	3
フローチャート	18
付録	27

関数一覧は以下のように構成されています。

テーマ (ハードウェア略号)

【関 数 名】	サンプル関数の名前
【引 数】	引数の型と概要
【処 理 内 容】	サンプル関数の処理内容
【使用 S F R】	レジスタ名
【c a l l 関 数】	呼び出し関数の名前と機能
【変 数】	サンプル関数での使用変数の型、名前、概要
【フ ァ イ ル 名】	関数本体が記述されているサンプルプログラム・ファイル名
【注 意 事 項】	関数使用上の注意

割り込み関数

【関 数 名】	サンプル関数の名前
【処 理 内 容】	サンプル関数の処理内容
【要 因】	要因名称と用途
【c a l l 関 数】	呼び出し関数の名前と機能
【変 数】	変数名、機能
【フ ァ イ ル 名】	対応するサンプルプログラム・ファイル名
【注 意 事 項】	関数使用上の注意

浮動小数点演算機能 ()

【関 数 名】	FloatingPointAddition
【引 数】	無し
【処 理 内 容】	浮動小数点加算を行います。
【使用 S F R】	無し
【c a l l 関 数】	AddSub_MainCalculation 加減算の共通処理部
【変 数】	<i>signed char</i> scRwSign RegW 符号部データ格納エリア <i>signed char</i> scRySign RegY 符号部データ格納エリア
【フ ァ イ ル 名】	V850_fp_main.c
【注 意 事 項】	無し

【関 数 名】	FloatingPointSubtract
【引 数】	無し
【処 理 内 容】	浮動小数点減算を行います。
【使用 S F R】	無し
【c a l l 関 数】	AddSub_MainCalculation 加減算の共通処理部
【変 数】	<i>signed char</i> scRwSign RegW 符号部データ格納エリア <i>signed char</i> scRySign RegY 符号部データ格納エリア
【フ ァ イ ル 名】	V850_fp_main.c
【注 意 事 項】	無し

【関 数 名】	AddSub_MainCalculation
【引 数】	無し
【処 理 内 容】	浮動小数点加算/減算の共通処理部で RegX RegX+RegY を行います。
【使用 S F R】	無し
【c a l l 関 数】	ShiftDownRegY RegY 仮数部シフトダウン ShiftDownRegX RegX 仮数部シフトダウン ExchangeRegX_W RegX、RegW 仮数部交換 AdditionBCD RegX+RegY 十進補正加算 SubtractBCD RegX-ucReg 十進補正減算 ClearRegW RegW 仮数部ゼロクリア Normalize 正規化
【変 数】	<i>signed char</i> scHoldData 指数部計算用作業エリア <i>unsigned char</i> ucHoldData 仮数部計算用作業エリア <i>unsigned char</i> ucBorrow ボローデータ格納エリア <i>signed char</i> scRwExp RegW 指数部格納エリア <i>signed char</i> scRxExp RegX 指数部格納エリア <i>signed char</i> scRyExp RegY 指数部格納エリア <i>unsigned char</i> ucRx[] RegX 仮数部格納エリア <i>unsigned char</i> ucRy[] RegY 仮数部格納エリア <i>signed char</i> scRxSign RegX 符号部格納エリア <i>signed char</i> scRwSign RegW 符号部格納エリア <i>unsigned char</i> ucFExChg レジスタ交換記録用フラグ <i>unsigned char</i> ucFOver オーバフロー・エラーフラグ
【フ ァ イ ル 名】	V850_fp_main.c
【注 意 事 項】	無し

【関 数 名】	FloatingPointMulti	
【引 数】	無し	
【処 理 内 容】	浮動小数点乗算を行います。	
【使用 S F R】	無し	
【c a l l 関 数】	ExchangeRegX_W	RegX、RegW 仮数部交換
	ClearRegX	RegX 符号部、指数部、仮数部ゼロクリア
	ShiftDownRegX	RegX 仮数部シフトダウン
	ShiftDownRegW	RegW 仮数部シフトダウン
	AdditionBCD	RegX+RegY 十進補正加算
	Normalize	正規化
【変 数】	<i>signed char</i> scRwSign	RegW 符号部格納エリア
	<i>signed char</i> scRxSign	RegX 符号部格納エリア
	<i>signed char</i> scRySign	RegY 符号部格納エリア
	<i>signed char</i> scRwExp	RegW 指数部格納エリア
	<i>signed char</i> scRxExp	RegX 指数部格納エリア
	<i>signed char</i> scRyExp	RegY 指数部格納エリア
	<i>signed char</i> scRdigCn	桁カウンタ
	<i>unsigned char</i> ucFOver	オーバフロー フラグ
	<i>unsigned char</i> ucFUnder	アンダフロー フラグ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

【関 数 名】	FloatingPointDivide	
【引 数】	無し	
【処 理 内 容】	浮動小数点除算を行います。	
【使用 S F R】	無し	
【c a l l 関 数】	ClearRegW	RegW 仮数部ゼロクリア
	ShiftUpRegX	RegX 仮数部シフトアップ
	ShiftUpRegW	RegW 仮数部シフトアップ
	AdditionBCD	RegX+RegY 十進補正加算
	SubtractBCD	RegX-ucReg 十進補正減算
	ExchangeRegX_W	RegX、RegW 仮数部交換
	Normalize	正規化
【変 数】	<i>unsigned char</i> ucBorrow	ボローデータ格納エリア
	<i>unsigned char</i> ucRw[]	RegW 仮数部格納エリア
	<i>unsigned char</i> ucRy[]	RegY 仮数部格納エリア
	<i>signed char</i> scRxExp	RegX 指数部格納エリア
	<i>signed char</i> scRyExp	RegY 指数部格納エリア
	<i>signed char</i> scRwSign	RegW 符号部格納エリア
	<i>signed char</i> scRxSign	RegX 符号部格納エリア
	<i>signed char</i> scRySign	RegY 符号部格納エリア
	<i>unsigned char</i> ucFDivErr	除数ゼロ・エラーフラグ
	<i>unsigned char</i> ucFOver	オーバフロー フラグ
	<i>unsigned char</i> ucFUnder	アンダフロー フラグ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

【関 数 名】	AdditionBCD	
【引 数】	無し	
【処 理 内 容】	RegX+RegY の十進補正加算処理を行います。	
【使用 S F R】	無し	
【c a l l 関 数】	無し	
【変 数】	<i>unsigned char</i> ucCarry	加算用キャリイ格納エリア
	<i>unsigned char</i> ucLowBit	加算用下位桁格納エリア
	<i>unsigned char</i> ucHighBit	加算用上位桁格納エリア
	<i>unsigned char</i> ucRx[]	RegX 仮数部格納エリア
	<i>unsigned char</i> ucRxWorkReg	RegX ワークレジスタ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

【関数名】	SubtractBCD	
【引数】	<i>unsigned char</i> ucReg[]	仮数部格納エリア
【処理内容】	RegX ucReg の十進補正減算を行います。	
【使用 S F R】	無し	
【call 関数】	無し	
【変数】	<i>unsigned char</i> ucCarry	減算用キャリイ格納エリア
	<i>unsigned char</i> ucLowBit	減算用下位桁格納エリア
	<i>unsigned char</i> ucHighBit	減算用上位桁格納エリア
	<i>unsigned char</i> ucRx[]	RegX 仮数部格納エリア
	<i>unsigned char</i> ucRxWorkReg	RegX ワークレジスタ
【ファイル名】	V850_fp_main.c	
【注意事項】	無し	

【関数名】	ShiftUpRegW	
【引数】	無し	
【処理内容】	RegW の仮数部に対して 1 桁アップ・シフトを行います。	
【使用 S F R】	無し	
【call 関数】	RegisterUpShiftMain	レジスタ・アップ・シフト
【変数】	<i>unsigned char</i> ucRwWorkReg	RegW ワークレジスタ
【ファイル名】	V850_fp_main.c	
【注意事項】	無し	

【関数名】	ShiftUpRegX	
【引数】	無し	
【処理内容】	RegX の仮数部に対して 1 桁アップ・シフトを行います。	
【使用 S F R】	無し	
【call 関数】	RegisterUpShiftMain	レジスタ・アップ・シフト
【変数】	<i>unsigned char</i> ucRxWorkReg	RegX ワークレジスタ
【ファイル名】	V850_fp_main.c	
【注意事項】	無し	

【関 数 名】	RegisterUpShiftMain
【引 数】	<i>unsigned char</i> ucReg[] レジスタ仮数部
【処 理 内 容】	各レジスタ・アップ・シフト処理の共通処理部です。
【使用 S F R】	無し
【c a l l 関 数】	無し
【変 数】	<i>unsigned char</i> ucBit 次バイト下位桁用退避エリア <i>unsigned char</i> ucLowBit アップ・シフト用下位桁退避エリア <i>unsigned char</i> ucHoldData 作業用データ退避エリア
【フ ァ イ ル 名】	V850_fp_main.c
【注 意 事 項】	無し

【関 数 名】	ShiftDownRegW
【引 数】	無し
【処 理 内 容】	RegW の仮数部に対して1桁ダウン・シフトを行います。
【使用 S F R】	無し
【c a l l 関 数】	RegisterDownShiftMain レジスタ・ダウン・シフト
【変 数】	<i>signed char</i> scRdigCn 加算回数カウンタ <i>unsigned char</i> ucRwWorkReg RegW ワークレジスタ
【フ ァ イ ル 名】	V850_fp_main.c
【注 意 事 項】	無し

【関 数 名】	ShiftDownRegY
【引 数】	無し
【処 理 内 容】	RegY の仮数部に対して1桁ダウン・シフトを行います。
【使用 S F R】	無し
【c a l l 関 数】	RegisterDownShiftMain レジスタ・ダウン・シフト
【変 数】	<i>unsigned char</i> ucRyWorkReg RegY ワークレジスタ
【フ ァ イ ル 名】	V850_fp_main.c
【注 意 事 項】	無し

【関 数 名】	ShiftDownRegX	
【引 数】	無し	
【処 理 内 容】	RegX の仮数部に対して 1 桁ダウン・シフトを行います。	
【使用 S F R】	無し	
【c a l l 関 数】	RegisterDownShiftMain	レジスタ・ダウン・シフト
【変 数】	<i>unsigned char</i> ucRxWorkReg	RegX ワークレジスタ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

【関 数 名】	RegisterDownShiftMain	
【引 数】	<i>unsigned char</i> ucReg[]	レジスタ仮数部
	<i>unsigned char</i> ucWorkReg	レジスタ・オーバフロー格納エリア
【処 理 内 容】	各レジスタ・ダウン・シフト処理の共通処理部です。	
【使用 S F R】	無し	
【c a l l 関 数】	無し	
【変 数】	<i>unsigned char</i> ucBit	次バイト上位桁用退避エリア
	<i>unsigned char</i> ucHighBit	ダウン・シフト用上位桁退避エリア
	<i>unsigned char</i> ucHoldData	作業用データ退避エリア
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

【関 数 名】	ExchangeRegX_W	
【引 数】	無し	
【処 理 内 容】	RegX と RegW の仮数部とワークを交換します。	
【使用 S F R】	無し	
【c a l l 関 数】	無し	
【変 数】	<i>unsigned char</i> ucHoldData	オペランド・データ交換用作業エリア
	<i>unsigned char</i> ucRx[]	RegX 仮数部格納エリア
	<i>unsigned char</i> ucRw[]	RegW 仮数部格納エリア
	<i>unsigned char</i> ucRxWorkReg	RegX ワークレジスタ
	<i>unsigned char</i> ucRwWorkReg	RegW ワークレジスタ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

【関 数 名】	Regularization	
【引 数】	無し	
【処 理 内 容】	RegX に対して正規化を行います。	
【使用 S F R】	無し	
【c a l l 関 数】	ShiftUpRegX	RegX レジスタ 1 桁アップ・シフト
	ClearRegX	RegX レジスタ 0 クリア
【変 数】	<i>unsigned char</i> ucRX[]	RegX 仮数部格納エリア
	<i>signed char</i> scRXExp	RegX 指数部格納エリア
	<i>unsigned char</i> ucFZero	演算結果ゼロフラグ
	<i>unsigned char</i> ucFUnder	アンダフロー フラグ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

【関 数 名】	ClearRegX	
【引 数】	無し	
【処 理 内 容】	RegX(符号部、指数部、仮数部)エリアを全て0クリアします。	
【使用 S F R】	無し	
【c a l l 関 数】	無し	
【変 数】	<i>signed short</i> scRxSign	RegX 符号部格納エリア
	<i>signed short</i> scRxExp	RegX 指数部格納エリア
	<i>unsigned char</i> ucRx[]	RegX 仮数部格納エリア
	<i>unsigned char</i> ucRxWorkReg	RegX ワークレジスタ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

【関 数 名】	ClearRegW	
【引 数】	無し	
【処 理 内 容】	RegW(仮数部)エリアを全て0クリアします。	
【使用 S F R】	無し	
【c a l l 関 数】	無し	
【変 数】	<i>unsigned char</i> ucRw[]	RegW 仮数部格納エリア
	<i>unsigned char</i> ucRwWorkReg	RegW ワークレジスタ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

浮動小数点演算

本サンプルプログラムでは、加減乗除算でそれぞれ使用する被演算データ、演算データ、演算結果データを下図に示すような浮動小数点付き10進数で扱います。各演算データは、正負の符号を表す符号部、小数点位置を表す指数部、符号も小数点もないデータの数値を表す仮数部から構成され、かつ管理されます。



符号部 : 1ビット(1バイトの最下位ビット)

指数部(16進2桁) : 2桁×4ビット

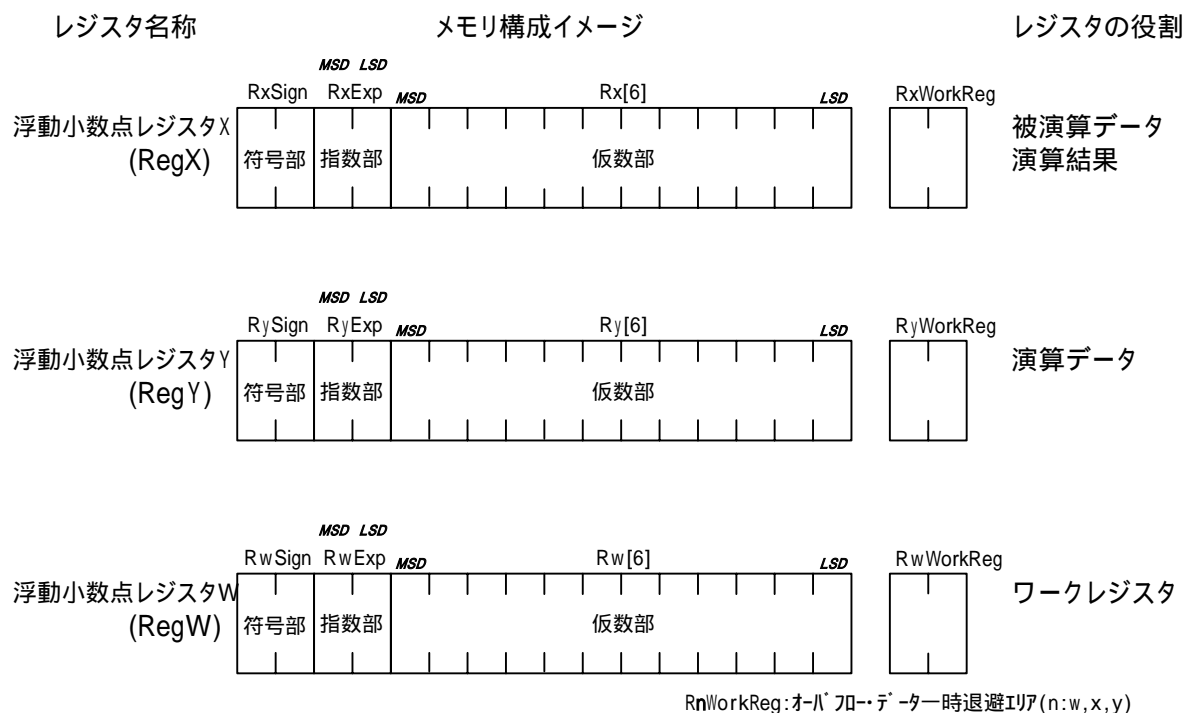
仮数部(10進12桁) : 12桁×4ビット(整数部1桁、小数部11桁)

この数値の管理構成を数式で表すと次のようになります。

$$(-1 \times (\text{符号部値})) \times (\text{仮数部値}) \times 10^{(\text{指数部値})}$$

[浮動小数点演算用レジスタ構成と説明]

このサンプルプログラムでは、浮動小数点演算データを扱うレジスタとして、被演算(演算結果)データ、演算データ、作業用の3つのレジスタ群を用意しています。



符号部

数値の符号は1ビットで表します。

正(+)の場合：0 (bit0)

負(-)の場合：1 (bit1)

注)空きビットは“0”固定です。

指数部

指数部は、表現する数値の底10の指数を2桁の16進数で構成しています。負数は2の補数表現になります。

指数値...	10 ⁷	10 ⁶ ...	10 ²	10 ¹	10 ⁰	10 ⁻¹	10 ⁻² ...	10 ⁻⁵	10 ⁻⁶
指数部...	07	06	02	01	00	FF	FE ...	FB	FA

備考) 2の補数：2ⁿからその数を引いた残りをいう。

仮数部

仮数部は有効数字の部分を、整数部 1 桁と小数部 1 1 桁の合計 1 2 桁で表現します。仮数部には正規化された 1 0 進数の絶対値として格納します。

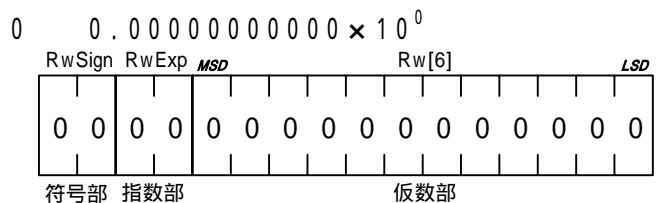
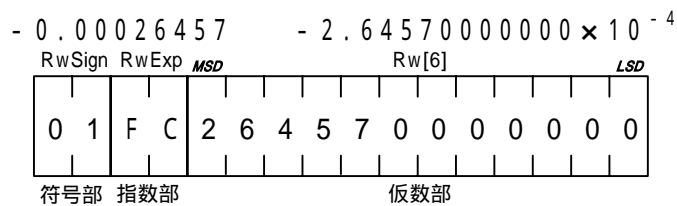
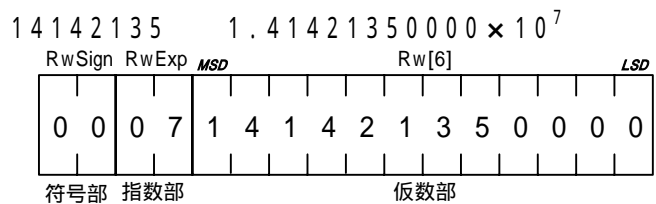
正規化とは、仮数部をあらかじめ定められた範囲内に収めるために、指数部と仮数部を調整することです。このサンプルプログラムでは、数値が 0 の場合を除いて仮数部の範囲を 1 仮数部(整数部) < 1 0 としていますので、正規化を行うと整数部に 0 以外の数値が格納されるように指数部が調整されます。

正規化の例を下表に示します。

正規化前の値	正規化後の値
1 7 3 2 . 0 5 0 8	1 . 7 3 2 0 5 0 8 × 1 0 ⁻³
0 . 2 2 3 6 0 6 7 9	2 . 2 3 6 0 6 7 9 × 1 0 ⁻¹
0 . 0 0 0 5 2 9 1 5	5 . 2 9 1 5 × 1 0 ⁻⁴

[浮動小数点レジスタへの格納例]

本サンプルプログラムで使用する浮動小数点レジスタへの格納例を示します。



[浮動小数点演算方法]

浮動小数点加算 (関数名: SFPADD)

計算式 : $Reg X = Reg X + Reg Y$

処理内容

Reg XとReg Yの浮動小数点数値を10進加算します。そして演算結果に対して正規化を行い、Reg Xに格納します。

入力設定

Reg X : 正規化された被加数値

Reg Y : 正規化された加数値

出力結果

Reg X : 正規化された演算結果

浮動小数点減算 (関数名: SFPSUB)

計算式 : $Reg X = Reg X - Reg Y$

処理内容

Reg XとReg Yの浮動小数点数値を10進減算します。そして演算結果に対して正規化を行い、Reg Xに格納します。

入力設定

Reg X : 正規化された被減数値

Reg Y : 正規化された減数値

出力結果

Reg X : 正規化された演算結果

浮動小数点乗算 (関数名: SFPMULT)

計算式 : $Reg X = Reg X \times Reg Y$

処理内容

Reg XとReg Yの浮動小数点数値を10進乗算します。そして演算結果に対して正規化を行い、Reg Xに格納します。

入力設定

Reg X : 正規化された乗数値

Reg Y : 正規化された被乗数値

出力結果

Reg X : 正規化された演算結果

浮動小数点除算（関数名：SFPDIV）

計算式： $Reg X = Reg X \div Reg Y$

処理内容

Reg XとReg Yの浮動小数点数値を10進除算します。そして演算結果に対して正規化を行い、Reg Xに格納します。

入力設定

Reg X：正規化された被除数値

Reg Y：正規化された除数値

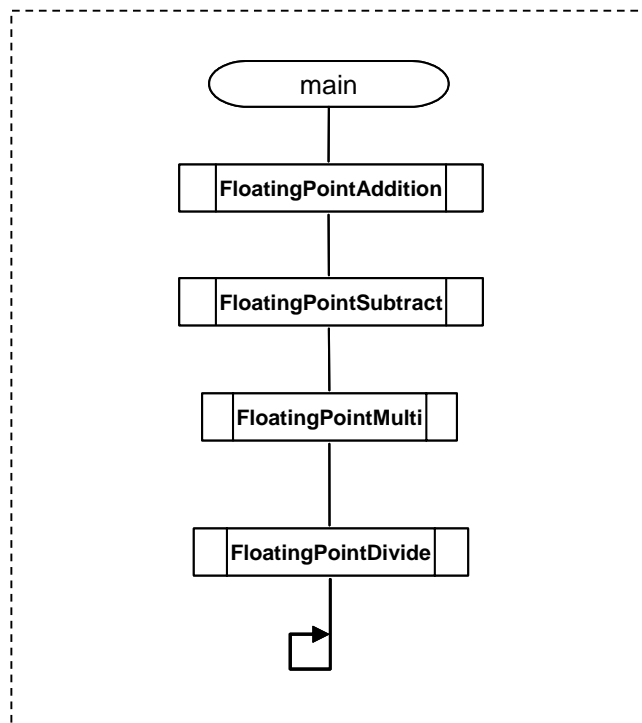
出力結果

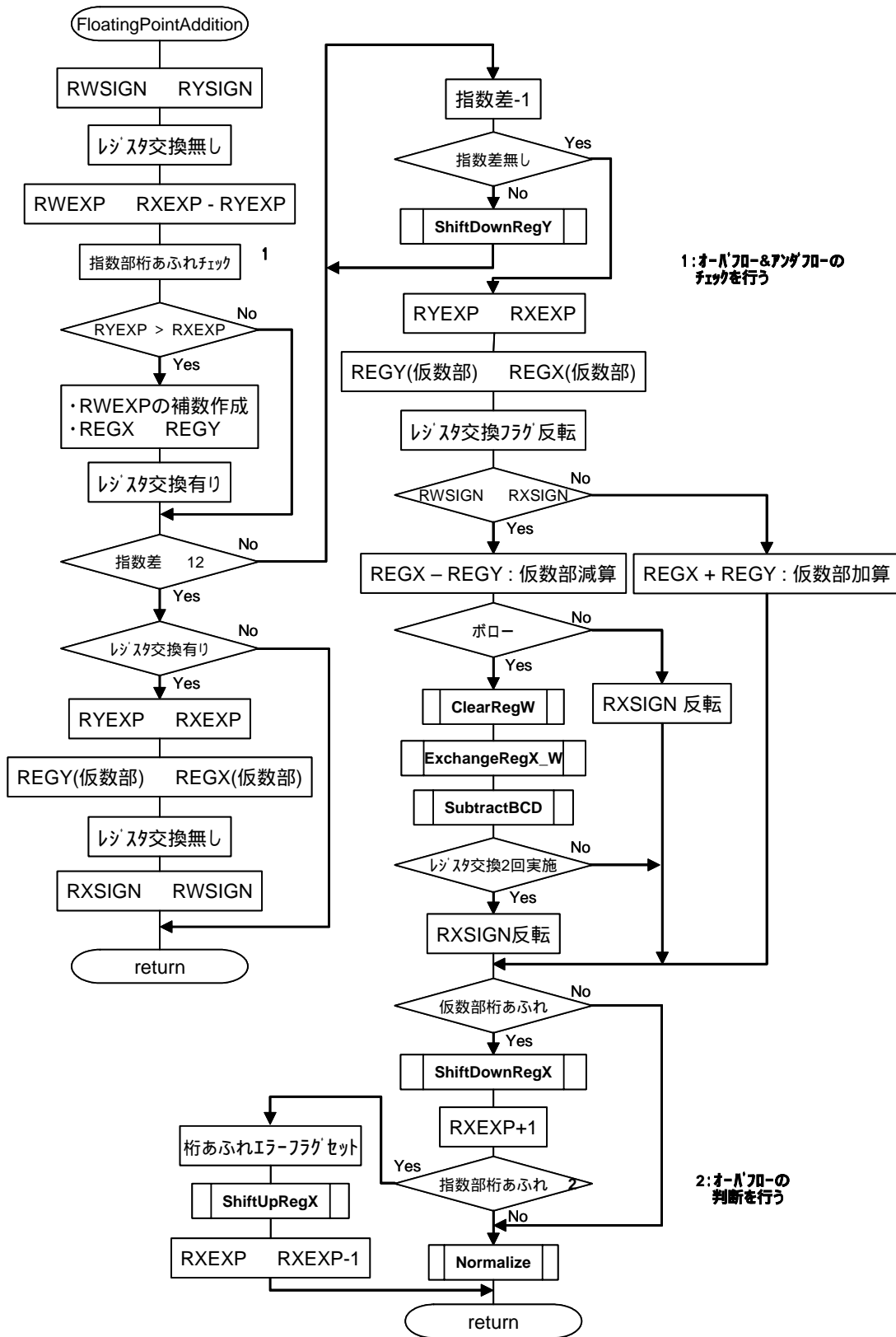
Reg X：正規化された演算結果

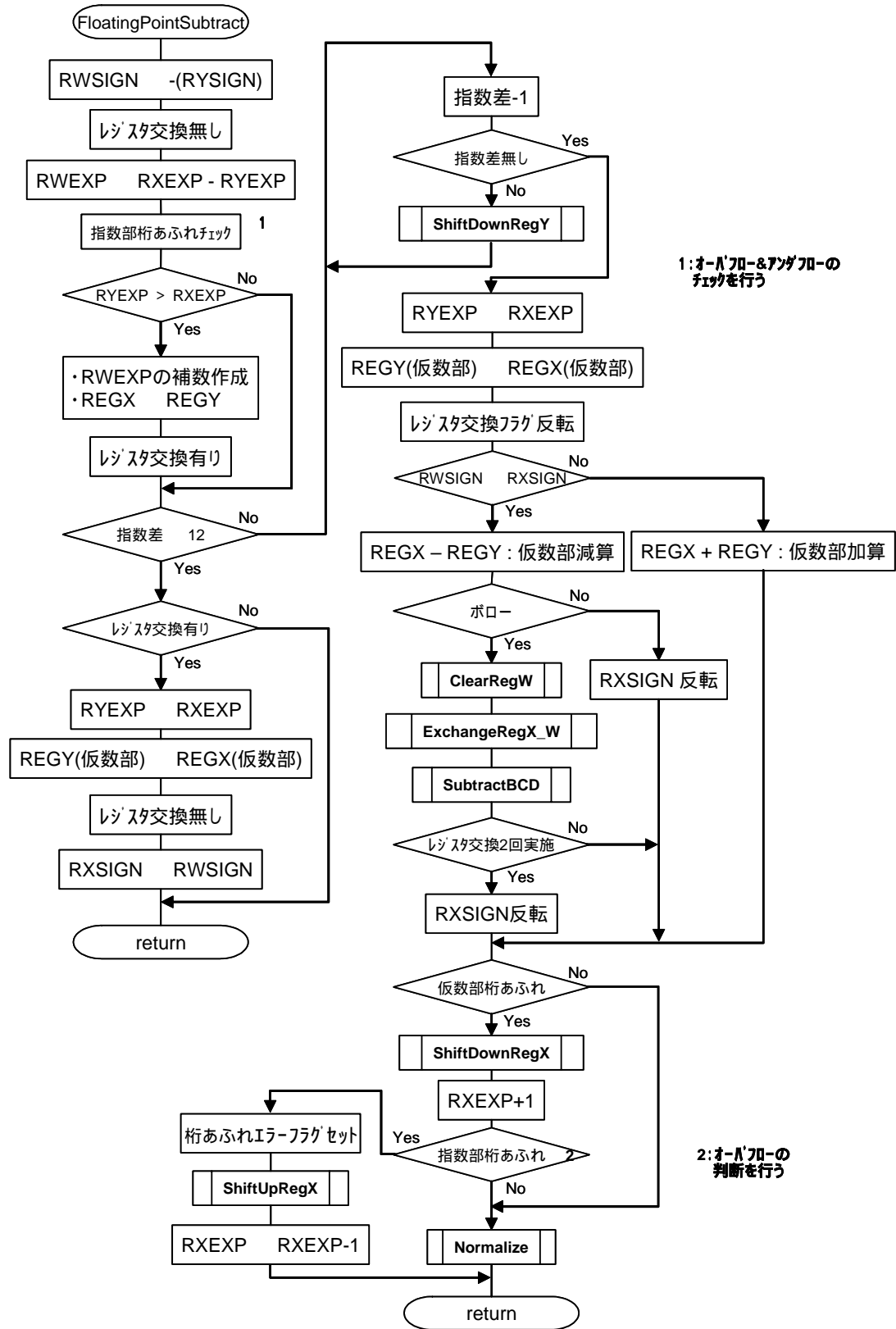
除数がゼロの場合、エラーとなりReg Xに格納されている被除数は保持されて処理を終了します。

フローチャート

サンプルプログラム動作例

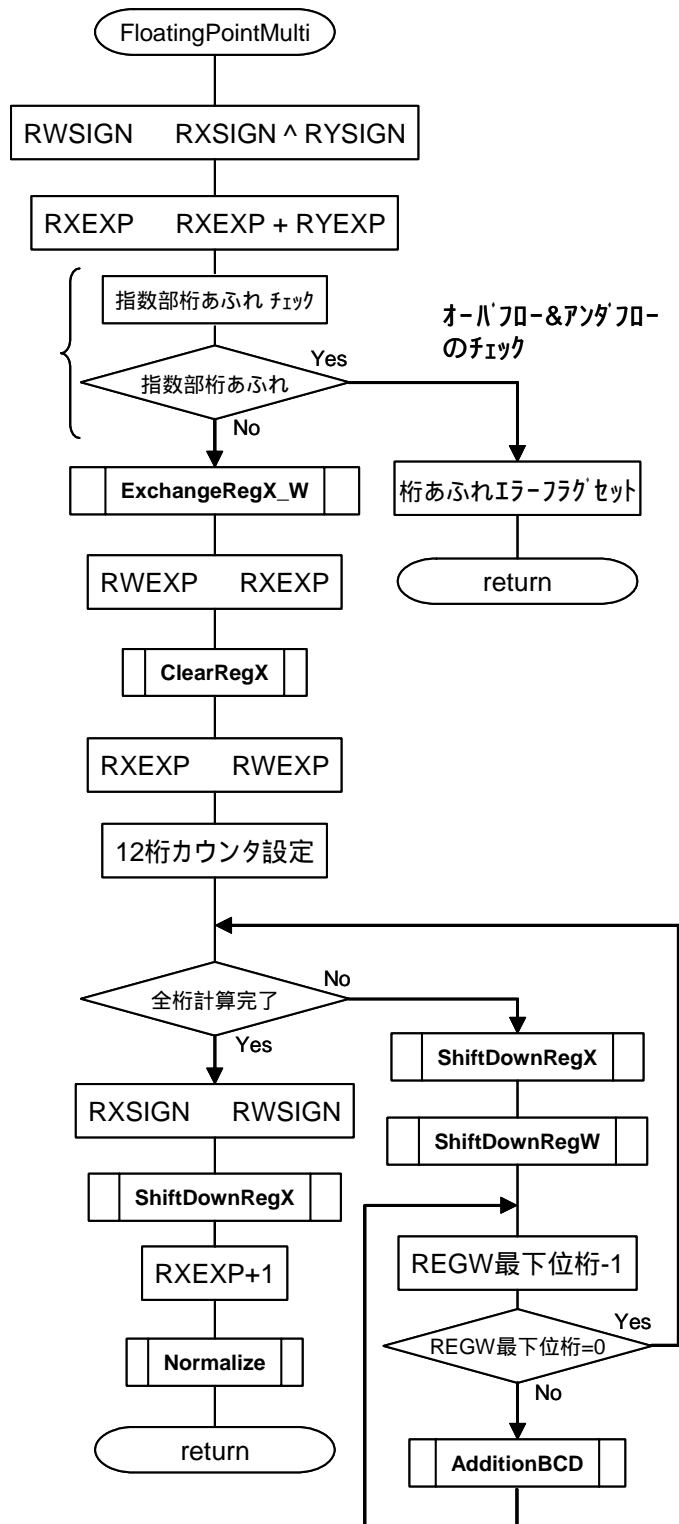


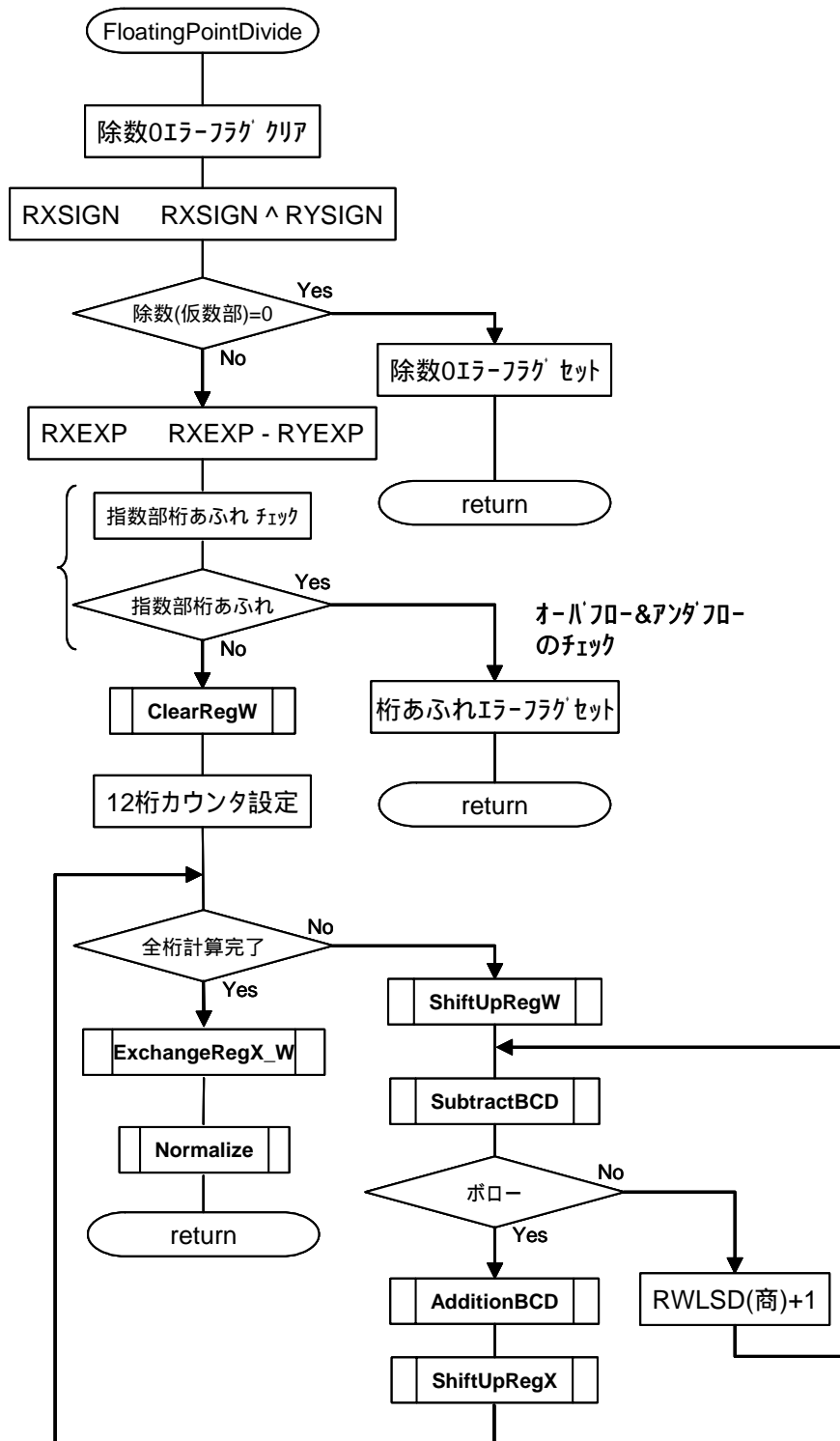


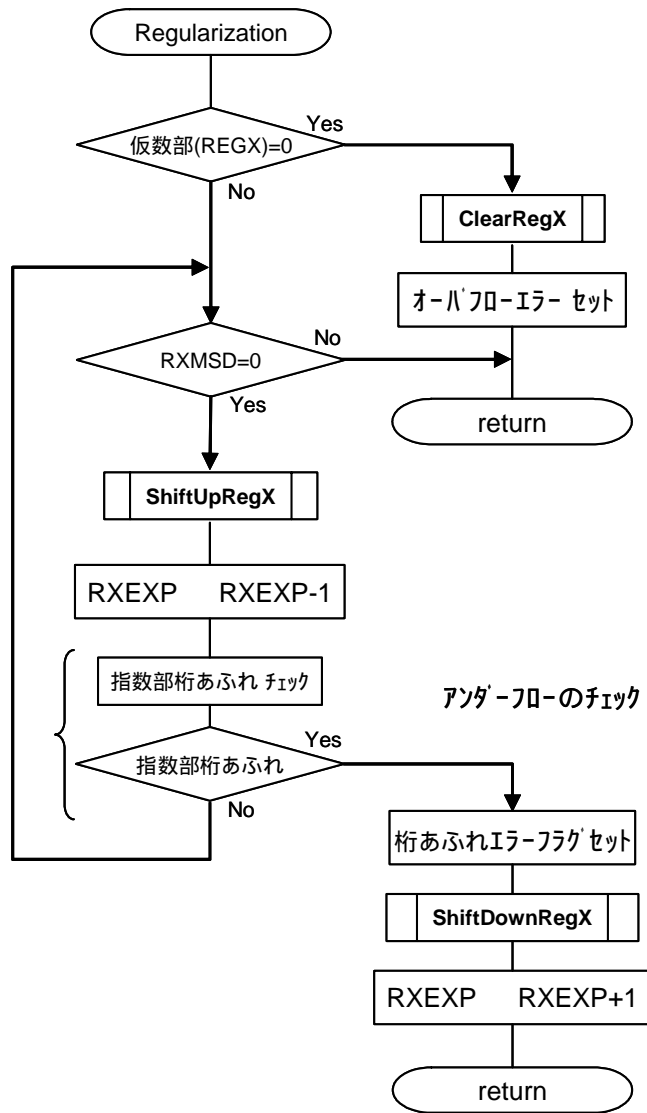


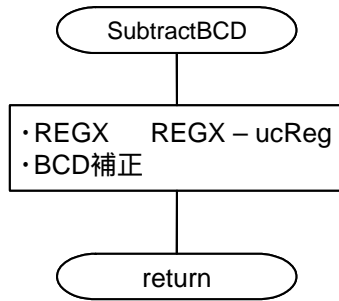
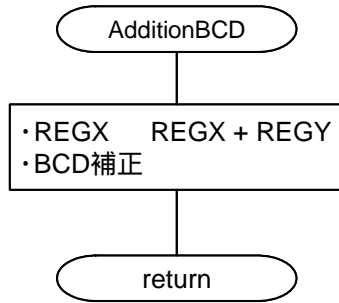
1:オーバーフロー&アンダフローのチェックを行う

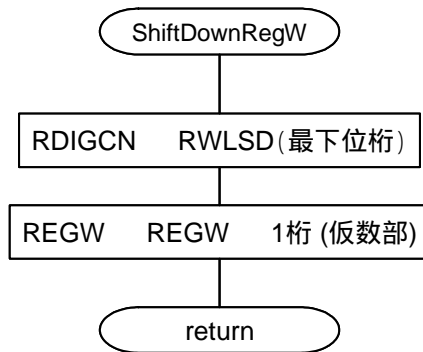
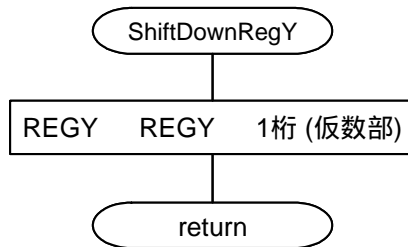
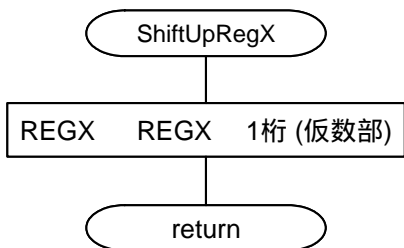
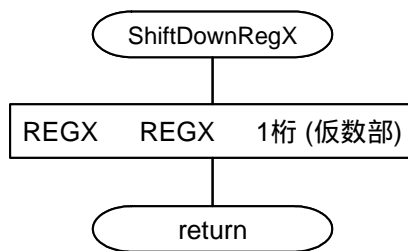
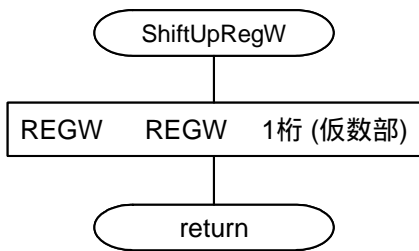
2:オーバーフローの判断を行う

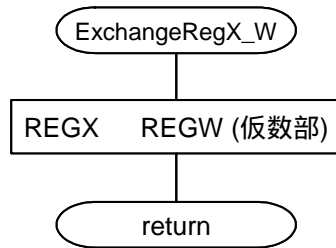
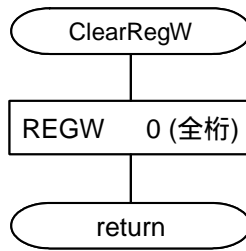
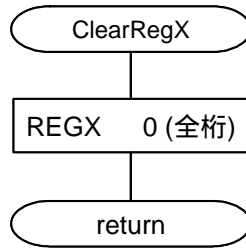












付 録 (ソースリスト)

```
/*
*****
*
* Title: 浮動小数点演算ソフト
*
* System   : Floating point arithmetic
* Version  : 1.01 2003/09/30
* File name: V850_fp_main.c
*
*
*
*
*****
*/

#pragma ioreg

/*
-----
      Constants/Variables
-----
*/
#include "V850_fp_type.h"
#include "V850_fp_symbol.h"
#include "V850_fp_RAM.h"

/**/
void FloatingPointSubtract(void);
void FloatingPointAddition(void);
void FloatingPointMulti(void);
void FloatingPointDivide(void);
void AddSub_MainCalculation(void);
void AdditionBCD(void);
UCHAR SubtractBCD(UCHAR ucReg[]);
void ShiftUpRegW(void);
void ShiftUpRegX(void);
UCHAR RegisterUpShiftMain(UCHAR ucReg[]);
void ShiftDownRegW(void);
void ShiftDownRegY(void);
void ShiftDownRegX(void);
void RegisterDownShiftMain(UCHAR ucReg[], UCHAR ucWorkReg);
void ExchangeRegX_W(void);
void Normalize(void);
void Over_Under_Check(void);
void ClearRegX(void);
void ClearRegW(void);

/*
-----
      Main Program
-----
*/
void main(void)
{
    FloatingPointAddition();           /* 浮動小数点加算 */
    FloatingPointSubtract();           /* 浮動小数点減算 */
    FloatingPointMulti();              /* 浮動小数点乗算 */
    FloatingPointDivide();             /* 浮動小数点除算 */

    while(1){                          /* Endless loop */
        __nop();
    }
    return;
}
```

```

/*
-----
  Functions Group
-----
*/
/*
-----
Abstract:
  This function ....浮動小数点演算(減算)
-----
*/
void FloatingPointSubtract(void)
{
  scRwSign = scRySign ^ CSIGN;      /* 減数の符号を反転したものを RWSIGN に格納 */
  AddSub_MainCalculation();

  return;
}

/*
-----
Abstract:
  This function ....浮動小数点演算(加算)
-----
*/
void FloatingPointAddition(void)
{
  scRwSign = scRySign;              /* 減数の符号を RWSIGN に格納 */
  AddSub_MainCalculation();

  return;
}

/*
-----
Abstract:
  This function ....加減算処理メイン
-----
*/
void AddSub_MainCalculation(void)
{
  UCHAR i;
  SCHAR scOldData;
  UCHAR ucOldData;
  UCHAR ucBorrow;

  /*** 指数部を揃える ***/
  ucFExChg = 0;                      /* レジスタ交換フラグをクリア */

  scRwExp = scRxExp - scRyExp;

  if( (scRxExp >= 0) && (scRyExp < 0) ){
    if( ((scRwExp & 0x80) >> 7) == 1 ){
      scRwExp = CEXOVER;
    }
  }
  }else if( (scRxExp < 0) && (scRyExp >= 0) ){
    if( (scRwExp <= (-128)) || (scRwExp >= 0) ){
      scRwExp = 0x81;
    }
  }
}

if( ((scRwExp & 0x80) >> 7) == 1 ){
  scRwExp ^= 0xff;                    /* RYEXP > RXEXP ? */
  scRwExp += 1;                      /* 指数差の補数をとる */
}

scOldData = scRxExp;                 /* REGX,REGY の指数部を交換 */

```



```

        if( ucFExChg == 1 ){
            /*** 転送(REGX REGY) ***/
            scRyExp = scRxExp;

            for( i=0; i<CDIGMAX; i++ ){
                ucRy[i] = ucRx[i];
            }

            ucFExChg = 0;
            scRxSign = scRwSign;
        }
    }
    return;
}

/*
-----
Abstract:
This function ....浮動小数点演算(乗算)
-----
*/
void FloatingPointMulti(void)
{
    /*** 結果符号を求める ***/
    scRwSign = scRxSign ^ scRySign;

    scRwExp = scRxExp + scRyExp;
    if( (scRxExp >= 0) && (scRyExp >= 0) ){
        if( scRwExp < 0 ){
            ucFOver = 1;
            return;
        }
    }
    else if( (scRxExp < 0) && (scRyExp < 0) ){
        if( scRwExp >= 0 ){
            ucFUnder = 1;
            return;
        }
    }
    scRxExp = scRwExp;

    /*** 繰り返し加算カウンタ作成 ***/
    ExchangeRegX_W();

    scRwExp = scRxExp;
    ClearRegX();
    scRxExp = scRwExp;
    scRwExp = CDIGIT;

    while(1){
        /*** 乗算 ***/
        scRwExp -= 1;
        if( scRwExp >= 0 ){
            ShiftDownRegX();
            ShiftDownRegW();
            while( scRdigCn > 0 ){
                scRdigCn -= 1;
                AdditionBCD();
            }
        }
        else{
            scRxSign = scRwSign;
            if( ucRxWorkReg > 0 ){
                ShiftDownRegX();
                scRxExp++;
            }
            Normalize();
            return;
        }
    }
}

```

```

    return;
}

/*
-----
Abstract:
  This function ....浮動小数点演算(除算)
-----
*/
void FloatingPointDivide(void)
{
    UCHAR i;
    UCHAR ucBorrow;

    ucFDivErr = 0;                                /* 除数ゼロ・フラグをクリア */

    /*** 結果符号を求める ***/
    scRxSign ^= scRySign;

    /*** 除数ゼロチェック ***/
    for( i=0; i<CDIGMAX; i++){
        if( ucRy[i] > 0 ){                        /* 仮数部 0 でない? */
            /*** 指数計算 ***/
            scRwExp = scRxExp - scRyExp;          /* 指数減算 */
            if( (scRxExp >= 0) && (scRyExp < 0) ){ /* 除算のオーバーフローチェック */
                if( scRwExp < 0 ){
                    ucFOver = 1;
                    return;
                }
            }else if( (scRxExp < 0) && (scRyExp >= 0) ){ /* 除算のアンダーフローチェック */
                if( scRwExp >= 0 ){
                    ucFUnder = 1;
                    return;
                }
            }
        }
        scRxExp = scRwExp;
        /*** 除算 ***/
        ClearRegW();                               /* REGW をゼロクリア */
        scRwSign = CDIGIT;                         /* 仮数部 12 桁カウンタ */

        while(1){
            scRwSign -= 1;
            if( scRwSign >= 0 ){
                ShiftUpRegW();                     /* REGW を 1 桁アップ・シフト */
                while(1){
                    ucBorrow = SubtractBCD(ucRy); /* 仮数部減算(REGX-REGY) */
                    if( ucBorrow == 0 ){
                        ucRw[0] += 1;             /* 商+1 */
                    }else{
                        AdditionBCD();           /* 仮数部加算(REGX+REGY) */
                        ShiftUpRegX();           /* REGX を 1 桁アップ・シフト */
                        break;
                    }
                }
            }else{
                ExchangeRegX_W();                 /* REGX, REGW を交換(仮数部) */
                Normalize();                       /* 正規化 */
            }
            return;
        }
    }
}

ucFDivErr = 1;                                /* 除数ゼロ・フラグセット */

return;
}

```

```

/*
-----
Abstract:
  This function ....仮数部加算
-----
*/
void AdditionBCD(void)
{
  UCHAR ucCarry;
  UCHAR ucLowBit;
  UCHAR ucHighBit;
  UCHAR i;

  ucCarry = 0;
  for( i=0; i<CDIGMAX; i++ ){
    ucLowBit = (ucRy[i] & 0x0f) + (ucRx[i] & 0x0f) + ucCarry;
    ucCarry = (ucLowBit & 0x10) >> 4;
    if( (ucCarry == 1) || (ucLowBit >= 10) ){
      ucLowBit += 0b00000110;
      ucCarry = 1;
    }else{
      ucCarry = 0;
    }
    ucLowBit &= 0x0f;

    ucHighBit = ((ucRy[i] & 0xf0) >> 4) + ((ucRx[i] & 0xf0) >> 4) + ucCarry;
    ucCarry = (ucHighBit & 0x10) >> 4;
    if( (ucCarry == 1) || (ucHighBit >= 10) ){
      ucHighBit += 0b00000110;
      ucCarry = 1;
    }else{
      ucCarry = 0;
    }
    ucHighBit = (ucHighBit & 0x0f) << 4;

    ucRx[i] = ucHighBit + ucLowBit;
  }
  ucRxWorkReg += ucCarry;

  return;
}

```

```

/*
-----
Abstract:
  This function ....仮数部減算
-----
*/
UCHAR SubtractBCD(UCHAR ucReg[])
{
  UCHAR ucCarry;
  UCHAR ucLowBit;
  UCHAR ucHighBit;
  UCHAR i;

  ucCarry = 0;

  for( i=0; i<CDIGMAX; i++ ){
    ucLowBit = (ucRx[i] & 0x0f) - (ucReg[i] & 0x0f) - ucCarry;
    if( ucLowBit > 18 ){
      ucLowBit = ((ucRx[i] & 0x0f)+0xa) - (ucReg[i] & 0x0f) - ucCarry;
      ucCarry = 1;
    }else{
      ucCarry = 0;
    }
  }

  ucHighBit = ((ucRx[i] & 0xf0) >> 4) - ((ucReg[i] & 0xf0) >> 4) - ucCarry;
  if( ucHighBit > 18 ){

```

```

        ucHighBit = (((ucRx[i] & 0xf0) >> 4)+0xa) - ((ucReg[i] & 0xf0) >> 4) - ucCarry;
        ucCarry = 1;
    }else{
        ucCarry = 0;
    }
    ucRx[i] = (ucHighBit << 4) + ucLowBit;
}

if( ucRxWorkReg > 0 ){
    ucRxWorkReg -= ucCarry;
    ucCarry = 0;
}

return (ucCarry);
}

```

/*

 Abstract:
 This functionレジスタ・アップ・シフト(REGW)

```

*/
void ShiftUpRegW(void)
{
    ucRwWorkReg = RegisterUpShiftMain(ucRw);

    return;
}

```

/*

 Abstract:
 This functionレジスタ・アップ・シフト(REGX)

```

*/
void ShiftUpRegX(void)
{
    ucRxWorkReg = RegisterUpShiftMain(ucRx);

    return;
}

```

/*

 Abstract:
 This functionレジスタ・アップ・シフトメイン

```

*/
UCHAR RegisterUpShiftMain(UCHAR ucReg[])
{
    UCHAR i;
    UCHAR ucBit;
    UCHAR ucLowBit;
    UCHAR ucHoldData;

    ucBit = 0;

    for( i=0; i<CDIGMAX; i++){
        ucLowBit = (ucReg[i] & 0x0f) << 4;          /* (b7-4) (b3-0)*/
        ucReg[i] = (ucReg[i] >> 4) | ucLowBit;
        ucHoldData = ucReg[i];                    /* データ退避 */
        ucReg[i] = (ucReg[i] & 0xf0) | ucBit;

        ucBit = ucHoldData & 0x0f;
    }
}

```

```

    return (ucBit);
}

/*
-----
Abstract:
This function ....レジスタ・ダウン・シフト(REGW)
-----
*/
void ShiftDownRegW(void)
{
    scRdigCn = ucRw[0] & 0x0f;          /* 桁カウントセット */

    RegisterDownShiftMain(ucRw, ucRwWorkReg);
    ucRwWorkReg = 0;

    return;
}

/*
-----
Abstract:
This function ....レジスタ・ダウン・シフト(REGY)
-----
*/
void ShiftDownRegY(void)
{
    RegisterDownShiftMain(ucRy, ucRyWorkReg);
    ucRyWorkReg = 0;

    return;
}

/*
-----
Abstract:
This function ....レジスタ・ダウン・シフト(REGX)
-----
*/
void ShiftDownRegX(void)
{
    RegisterDownShiftMain(ucRx, ucRxWorkReg);
    ucRxWorkReg = 0;

    return;
}

/*
-----
Abstract:
This function ....レジスタ・ダウン・シフトメイン
-----
*/
void RegisterDownShiftMain(UCHAR ucReg[], UCHAR ucWorkReg)
{
    SCHAR i;
    UCHAR ucBit;
    UCHAR ucHighBit;
    UCHAR ucHoldData;

    ucBit = 0;
    for( i=(CDIGMAX-1); i>=0; i-- ){
        ucHighBit = (ucReg[i] & 0x0f) << 4;
        ucReg[i] = (ucReg[i] >> 4) | ucHighBit;
        ucHoldData = ucReg[i];
        ucReg[i] = (ucReg[i] & 0x0f) | ucBit;
    }
}

```

```

        ucBit = ucHoldData & 0xf0;
    }

    ucReg[CDIGMAX-1] = (ucWorkReg << 4) + ucReg[CDIGMAX-1];

    return;
}

```

/*

 Abstract:
 This functionレジスタ交換

```

*/
void ExchangeRegX_W(void)
{
    UCHAR i;
    UCHAR ucHoldData;

    for( i=0; i<CDIGMAX; i++){
        ucHoldData = ucRx[i];
        ucRx[i] = ucRw[i];
        ucRw[i] = ucHoldData;
    }
    ucHoldData = ucRxWorkReg;
    ucRxWorkReg = ucRwWorkReg;
    ucRwWorkReg = ucHoldData;

    return;
}

```

/*

 Abstract:
 This function正規化

```

*/
void Normalize(void)
{
    UCHAR i;

    for( i=0; i<CDIGMAX; i++){
        if( ucRx[i] > 0 ){
            /** 正規化 **/
            while( (ucRx[CDIGMAX-1] & 0xf0) == 0 ){ /* MSD = 0 ? */
                ShiftUpRegX(); /* 1桁アップ・シフト */
                if( scRxExp < 0 ){ /* 減算のアンダーフローチェック */
                    scRxExp--; /* 指数-1 */
                    if( (scRxExp) >= 0 ){
                        ucFUnder = 1;
                        ShiftDownRegX();
                        scRxExp++;
                        return;
                    }
                }else{
                    scRxExp--; /* 指数-1 */
                }
            }
        }

        return;
    }
}

ClearRegX(); /* REGX をゼロクリア */
ucFZero = 1; /* 演算結果ゼロフラグセット */

return;

```

```

}

/*
-----
Abstract:
  This function ....REGX ゼロクリア
-----
*/
void ClearRegX(void)
{
  UCHAR i;

  scRxSign = 0;          /* 符号部クリア */
  scRxExp = 0;          /* 指数部クリア */

  for( i=0; i<CDIGMAX; i++){ /* 仮数部クリア */
    ucRx[i] = 0;
  }

  ucRxWorkReg = 0;      /* ワークエリア クリア */

  return;
}

/*
-----
Abstract:
  This function ....REGW ゼロクリア(仮数部)
-----
*/
void ClearRegW(void)
{
  UCHAR i;

  for( i=0; i<CDIGMAX; i++){
    ucRw[i] = 0;
  }

  ucRwWorkReg = 0;      /* ワークエリア クリア */

  return;
}

```

```

/*
*****
*
* Title: グローバル変数定義
*
*****
*/

SCHAR scRxSign;          /* REGX の符号部 */
SCHAR scRxExp;           /* REGX の指数部 */
UCHAR ucRx[CDIGMAX];    /* REGX の仮数部 */

SCHAR scRySign;          /* REGY の符号部 */
SCHAR scRyExp;           /* REGY の指数部 */
UCHAR ucRy[CDIGMAX];    /* REGY の仮数部 */

SCHAR scRwSign;          /* REGW の符号部 */
SCHAR scRwExp;           /* REGW の指数部 */
UCHAR ucRw[CDIGMAX];    /* REGW の仮数部 */

UCHAR scRdigCn;          /* 桁カウンタ */

UCHAR ucFZero;           /* 演算結果ゼロフラグ */
UCHAR ucFOver;           /* オーバーフロー・フラグ */
UCHAR ucFUnder;         /* アンダーフロー・フラグ */
UCHAR ucFDivErr;        /* 除数ゼロ・エラー・フラグ */
UCHAR ucFExChg;         /* レジスタ交換フラグ */

UCHAR ucRxWorkReg;       /* REGX ワークレジスタ */
UCHAR ucRyWorkReg;       /* REGY ワークレジスタ */
UCHAR ucRwWorkReg;       /* REGW ワークレジスタ */

/*
*****
*
* Title: 定数シンボル定義
*
*****
*/

#define CDIGMAX (12/2)          /* 仮数部サイズ */
#define CREGSIZ (CDIGMAX+2)    /* 演算レジスタサイズ(仮数部+指数部+符号部) */

#define CDIGIT (12)
#define CEXOVER (12)           /* オーバーフロー限界値 */
#define CEXUNDER (0x100-CEXOVER) /* アンダーフロー限界値 */

#define CSIGHN (0b00000001)

```