お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。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。
3. 当社製品を改造、改変、複製等しないでください。
4. 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お客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、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に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   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

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障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。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、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については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害など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、機器またはシステムとしての出荷保証をお願いいたします。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、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、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。お客様が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。
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固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

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。

サンプルプログラム関数仕様(浮動小数点演算編)
使用法説明書(V850ES/Kx1,V850ES/Kx1+)

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。

NEC Electronics Corporation
January 2004

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 ＳＦＲ】	レジスタ名
【ｃａｌｌ 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
【ｃａｌｌ 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 ＳＦＲ】	無し
【ｃａｌｌ関数】	AddSub_MainCalculation 加減算の共通処理部
【変　　数】	*signed char* **scRwSign**　　RegW 符号部データ格納エリア
	signed char **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 ＳＦＲ】	無し
【ｃａｌｌ関数】	AddSub_MainCalculation 加減算の共通処理部
【変　　数】	*signed char* **scRwSign**　　RegW 符号部データ格納エリア
	signed char **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	
【ｃａｌｌ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】	*signed char* **scHoldData**	指数部計算用作業エリア
	unsigned char **ucHoldData**	仮数部計算用作業エリア
	unsigned char **ucBorrow**	ボローデータ格納エリア
	signed char **scRwExp**	RegW 指数部格納エリア
	signed char **scRxExp**	RegX 指数部格納エリア
	signed char **scRyExp**	RegY 指数部格納エリア
	unsigned char **ucRx[]**	RegX 仮数部格納エリア
	unsigned char **ucRy[]**	RegY 仮数部格納エリア
	signed char **scRxSign**	RegX 符号部格納エリア
	signed char **scRwSign**	RegW 符号部格納エリア
	unsigned char **ucFExChg**	レジスタ交換記録用フラグ
	unsigned char **ucF0ver**	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	
【ｃａｌｌ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】	*signed char* **scRwSign**	RegW 符号部格納エリア
	signed char **scRxSign**	RegX 符号部格納エリア
	signed char **scRySign**	RegY 符号部格納エリア
	signed char **scRwExp**	RegW 指数部格納エリア
	signed char **scRxExp**	RegX 指数部格納エリア
	signed char **scRyExp**	RegY 指数部格納エリア
	signed char **scRdigCn**	桁カウンタ
	unsigned char **ucFOver**	オーバフロー フラグ
	unsigned char **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	
【call 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】	*unsigned char* **ucBorrow**	ボローデータ格納エリア
	unsigned char **ucRw[]**	RegW 仮数部格納エリア
	unsigned char **ucRy[]**	RegY 仮数部格納エリア
	signed char **scRxExp**	RegX 指数部格納エリア
	signed char **scRyExp**	RegY 指数部格納エリア
	signed char **scRwSign**	RegW 符号部格納エリア
	signed char **scRxSign**	RegX 符号部格納エリア
	signed char **scRySign**	RegY 符号部格納エリア
	unsigned char **ucFDivErr**	除数ゼロ･エラーフラグ
	unsigned char **ucFOver**	オーバフロー フラグ
	unsigned char **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	
【call 関 数】	無し	
【変　　数】	*unsigned char* **ucCarry**	加算用キャリー格納エリア
	unsigned char **ucLowBit**	加算用下位桁格納エリア
	unsigned char **ucHighBit**	加算用上位桁格納エリア
	unsigned char **ucRx[]**	RegX 仮数部格納エリア
	unsigned char **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】	*unsigned char* **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】	*unsigned char* **ucCarry**	減算用キャリー格納エリア
	unsigned char **ucLowBit**	減算用下位桁格納エリア
	unsigned char **ucHighBit**	減算用上位桁格納エリア
	unsigned char **ucRx[]**	RegX 仮数部格納エリア
	unsigned char **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】	*unsigned char* **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】	*unsigned char* **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】	*unsigned char* **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 ＳＦＲ】	無し
【ｃａｌｌ関数】	無し
【変　　数】	*unsigned char* **ucBit**　　次バイト下位桁用退避エリア
	unsigned char **ucLowBit**　　アップ･シフト用下位桁退避エリア
	unsigned char **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 ＳＦＲ】	無し	
【ｃａｌｌ関数】	RegisterDownShiftMain	レジスタ･ダウン･シフト
【変　　数】	*signed char* **scRdigCn**	加算回数カウンタ
	unsigned char **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 ＳＦＲ】	無し	
【ｃａｌｌ関数】	RegisterDownShiftMain	レジスタ･ダウン･シフト
【変　　数】	*unsigned char* **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	
【ｃａｌｌ 関 数】	RegisterDownShiftMain	レジスタ･ダウン･シフト
【変　　数】	*unsigned char* **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】	*unsigned char* **ucReg[]**	レジスタ仮数部
	unsigned char **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	
【ｃａｌｌ 関 数】	無し	
【変　　数】	*unsigned char* **ucBit**	次バイト上位桁用退避エリア
	unsigned char **ucHighBit**	ダウン･シフト用上位桁退避エリア
	unsigned char **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	
【call 関 数】	無し	
【変　　数】	*unsigned char* **ucHoldData**	ｵｰﾊﾞﾌﾛｰ･ﾃﾞｰﾀ交換用作業エリア
	unsigned char **ucRx[]**	RegX 仮数部格納エリア
	unsigned char **ucRw[]**	RegW 仮数部格納エリア
	unsigned char **ucRxWorkReg**	RegX ワークレジスタ
	unsigned char **ucRwWorkReg**	RegW ワークレジスタ
【フ ァ イ ル 名】	V850_fp_main.c	
【注 意 事 項】	無し	

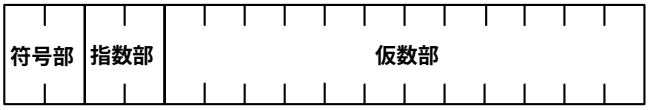
【関　数　名】	Regularization	
【引　　数】	無し	
【処 理 内 容】	RegX に対して正規化を行います。	
【使 用 S F R】	無し	
【call 関 数】	ShiftUpRegX	RegX レジスタ１桁アップ･シフト
	ClearRegX	RegX レジスタ０クリア
【変　　数】	*unsigned char* **ucRX[]**	RegX 仮数部格納エリア
	signed char **scRXExp**	RegX 指数部格納エリア
	unsigned char **ucFZero**	演算結果ゼロフラグ
	unsigned char **ucFUnder**	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 ＳＦＲ】	無し	
【ｃａｌｌ関数】	無し	
【変　　数】	*signed short* **scRxSign**	RegX 符号部格納エリア
	signed short **scRxExp**	RegX 指数部格納エリア
	unsigned char **ucRx[]**	RegX 仮数部格納エリア
	unsigned char **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 ＳＦＲ】	無し	
【ｃａｌｌ関数】	無し	
【変　　数】	*unsigned char* **ucRw[]**	RegW 仮数部格納エリア
	unsigned char **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

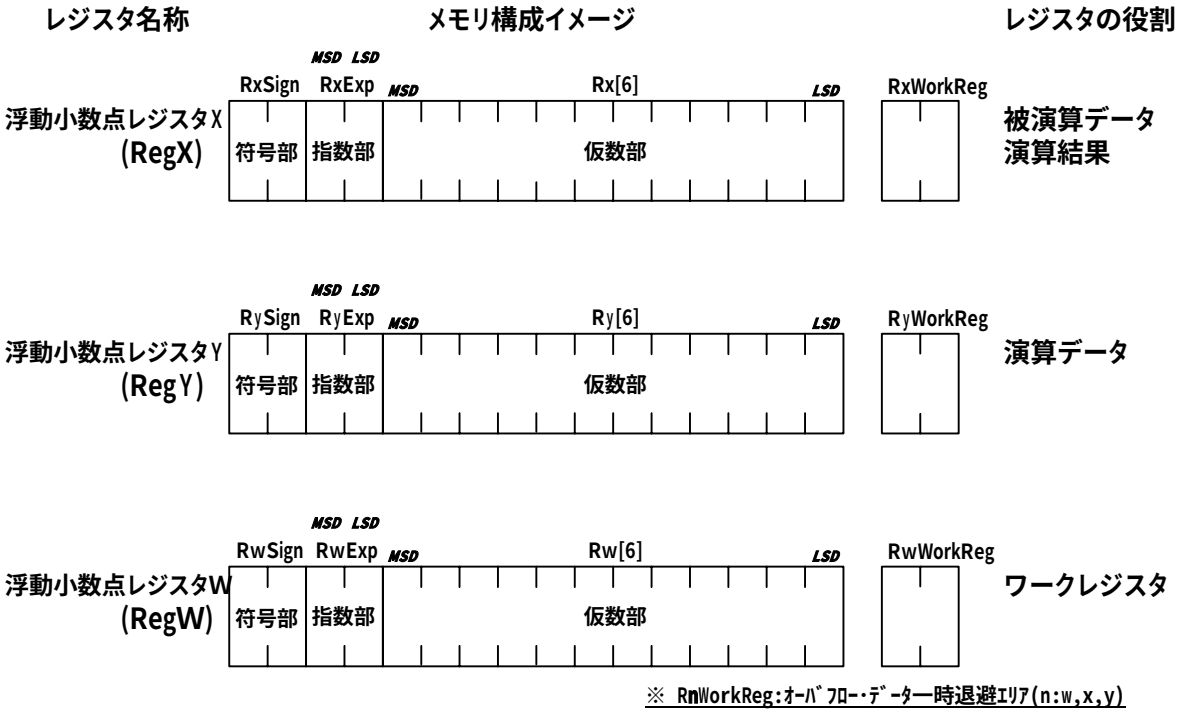
■ 仮数部（１０進１２桁）：１２桁×４ビット(整数部１桁、小数部１１桁)

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。



①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。

②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：2^{n}からその数を引いた残りをいう。

③仮数部

仮数部は有効数字の部分を、整数部１桁と小数部１１桁の合計１２桁で表現します。仮数部には正規化された１０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)＜１０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。

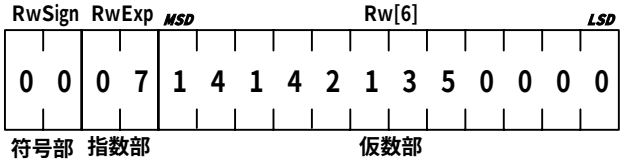
正規化の例を下表に示します。

正規化前の値	正規化後の値
１７３２．０５０８	１．７３２０５０８×１０^{3}
０．２２３６０６７９	２．２３６０６７９×１０$^{-1}$
０．０００５２９１５	５．２９１５×１０$^{-4}$

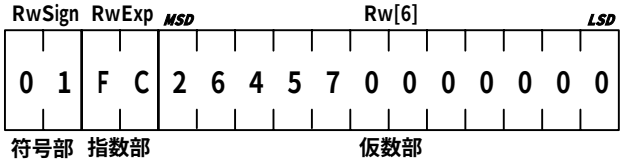
[浮動小数点レジスタへの格納例]

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。

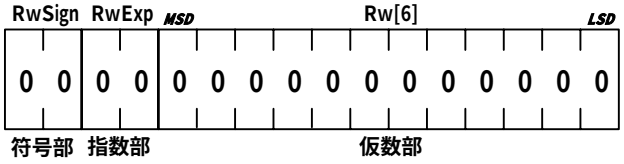
14142135 → 1.41421350000×10^{7}



−0.00026457 → −2.64570000000×10^{-4}



0 → 0.00000000000×10^{0}



［浮動小数点演算方法］

①浮動小数点加算（関数名：SFPADD）

計算式　：　ＲｅｇＸ　＝　ＲｅｇＸ　＋　ＲｅｇＹ

●処理内容

ＲｅｇＸとＲｅｇＹ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、ＲｅｇＸに格納します。

●入力設定

ＲｅｇＸ：正規化された被加数値

ＲｅｇＹ：正規化された加数値

●出力結果

ＲｅｇＸ：正規化された演算結果

②浮動小数点減算（関数名：SFPSUB）

計算式　：　ＲｅｇＸ　＝　ＲｅｇＸ　－　ＲｅｇＹ

●処理内容

ＲｅｇＸとＲｅｇＹ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、ＲｅｇＸに格納します。

●入力設定

ＲｅｇＸ：正規化された被減数値

ＲｅｇＹ：正規化された減数値

●出力結果

ＲｅｇＸ：正規化された演算結果

③浮動小数点乗算（関数名：SFPMULT）

計算式　：　ＲｅｇＸ　＝　ＲｅｇＸ　×　ＲｅｇＹ

●処理内容

ＲｅｇＸとＲｅｇＹ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、ＲｅｇＸに格納します。

●入力設定

ＲｅｇＸ：正規化された乗数値

ＲｅｇＹ：正規化された被乗数値

●出力結果

ＲｅｇＸ：正規化された演算結果

④浮動小数点除算（関数名：SFPDIV）

計算式　：　ＲｅｇＸ　＝　ＲｅｇＸ　÷　ＲｅｇＹ

●処理内容

ＲｅｇＸとＲｅｇＹ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、ＲｅｇＸに格納します。

●入力設定

ＲｅｇＸ：正規化された被除数値

ＲｅｇＹ：正規化された除数値

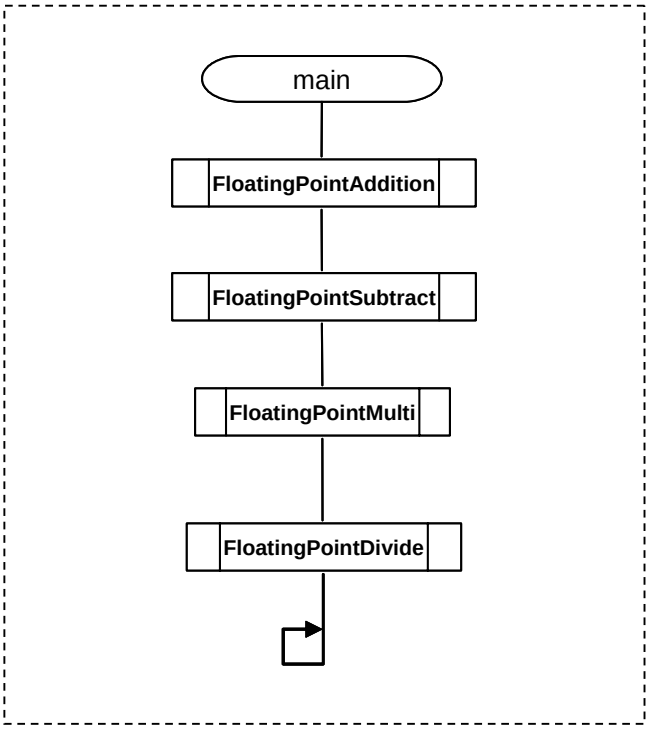
●出力結果

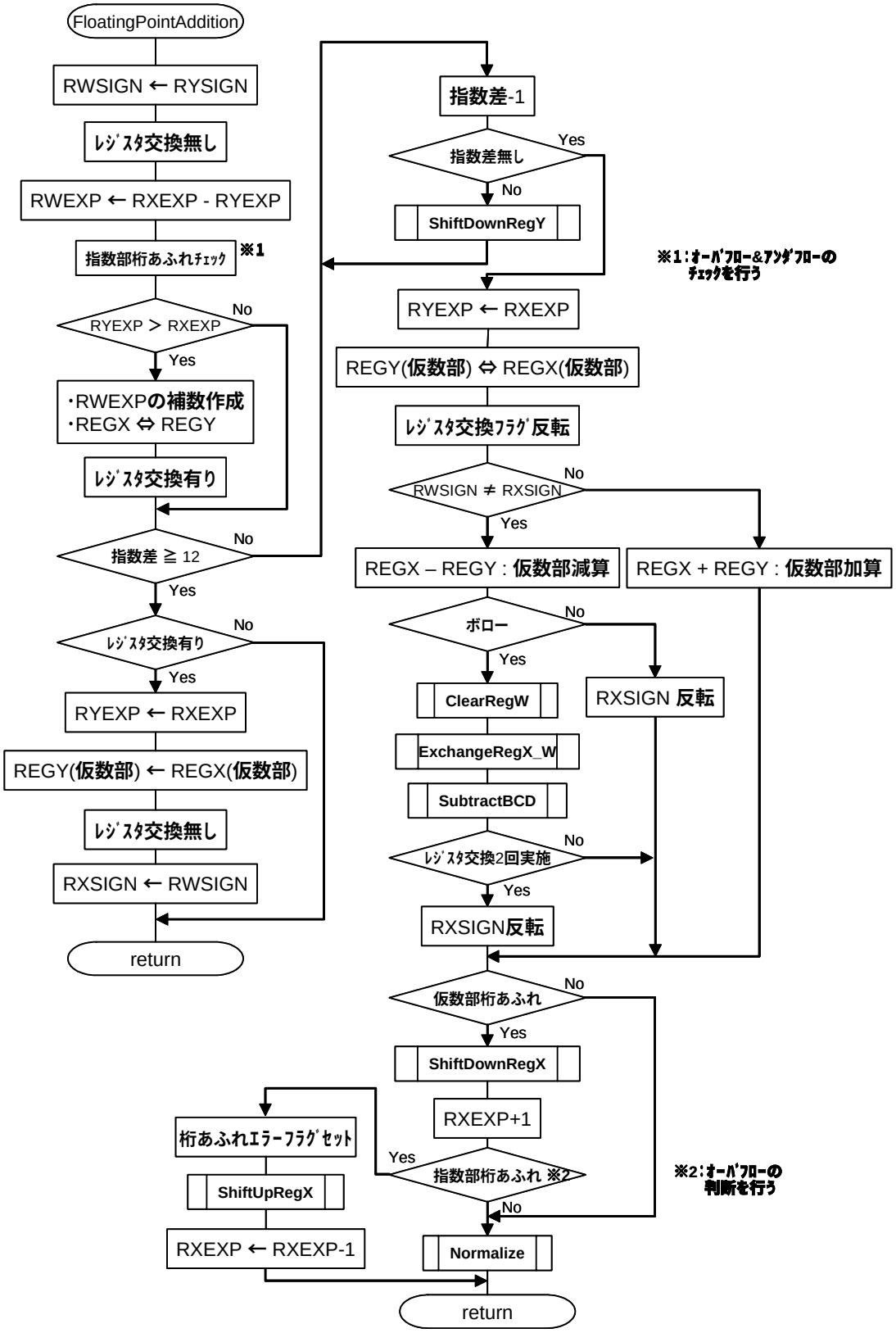
ＲｅｇＸ：正規化された演算結果

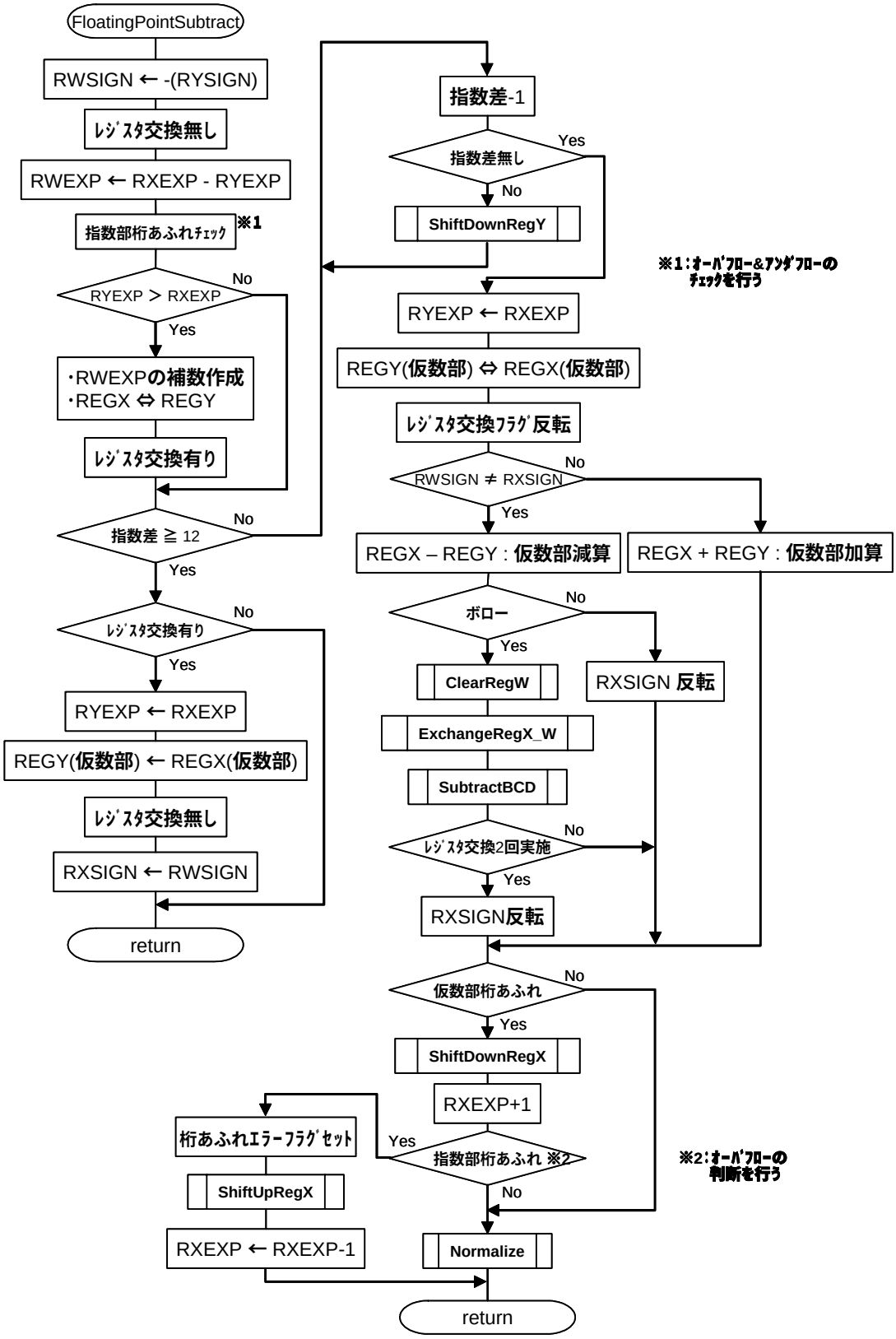
除数がゼロの場合、エラーとなりＲｅｇＸに格納されている被除数は保持されて処理を終了します。

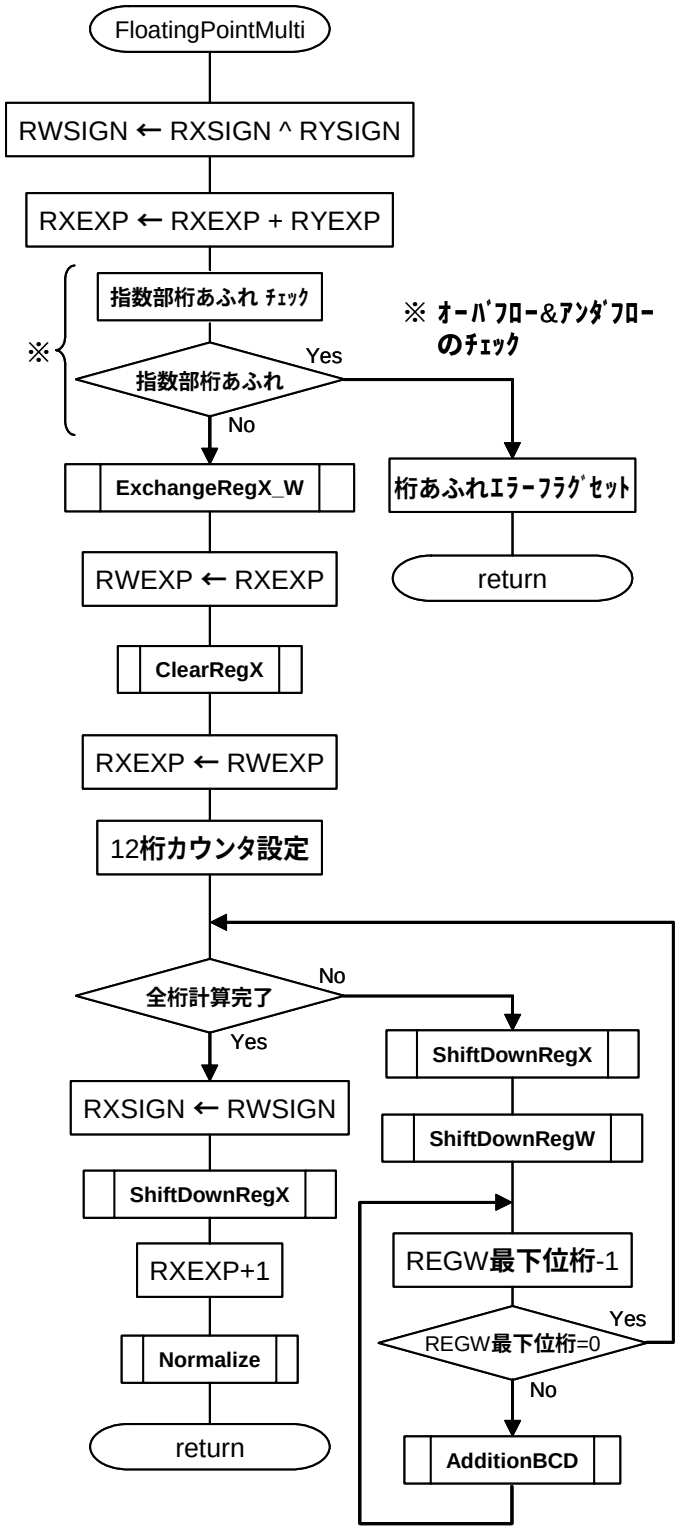
フローチャート

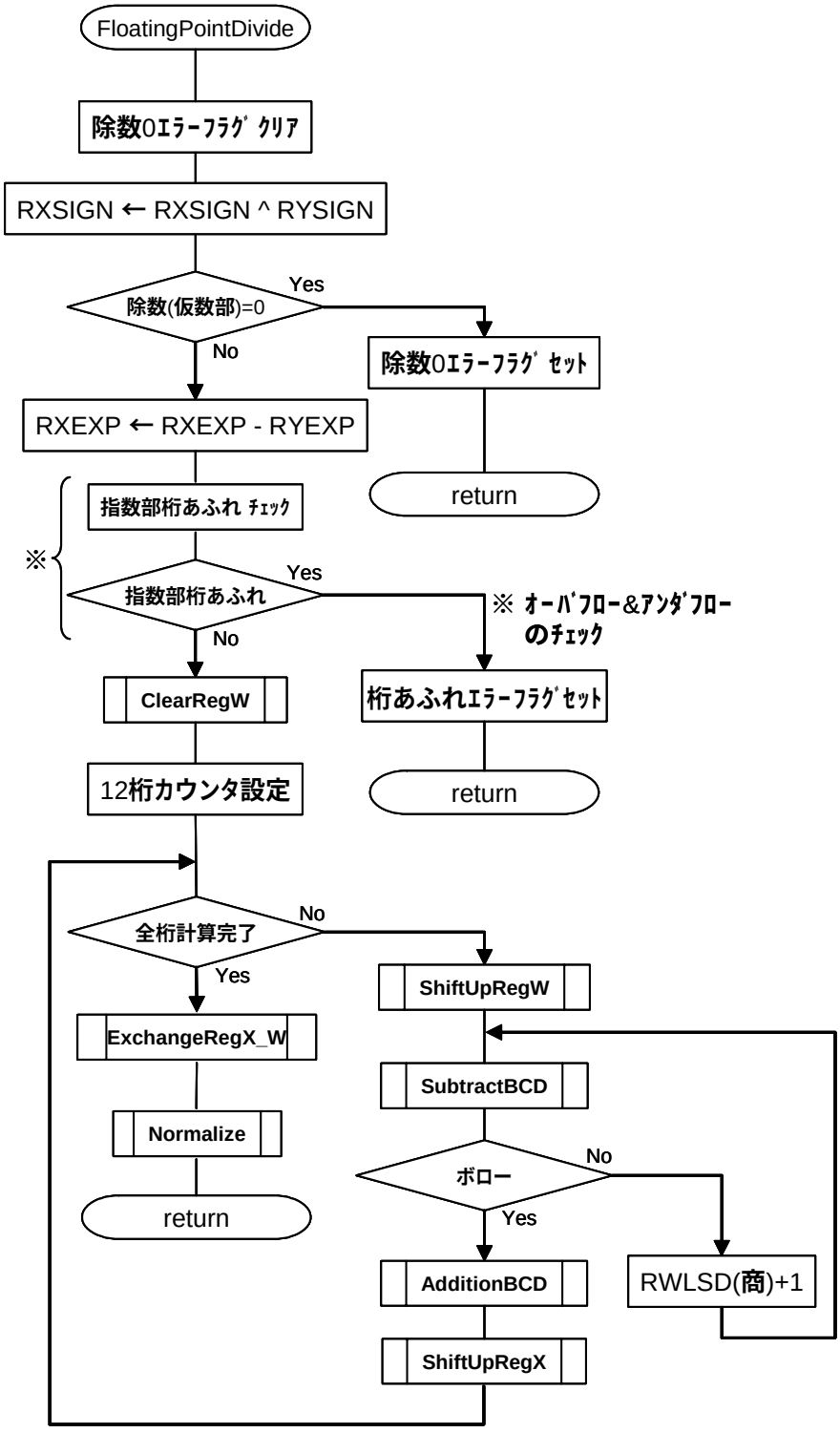
サンプルプログラム動作例

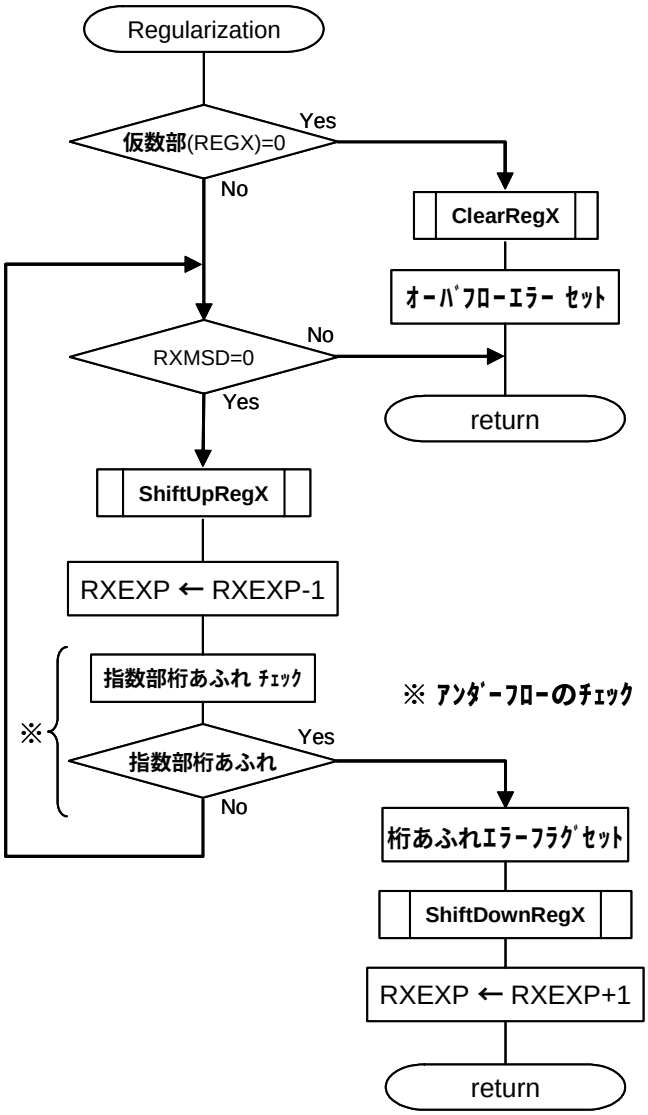


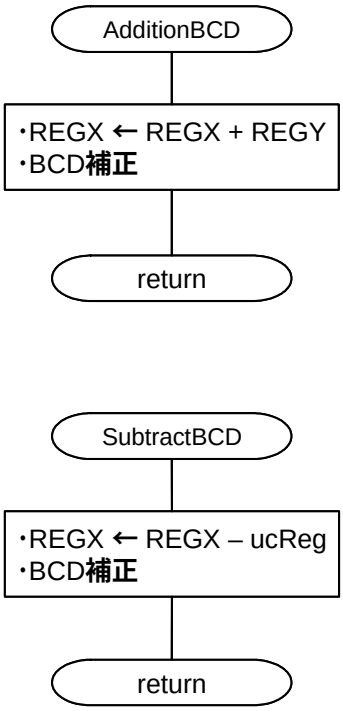

FloatingPointAddition
RWSIGN ← RYSIGN
ﾚｼﾞｽﾀ交換無し
RWEXP ← RXEXP - RYEXP
指数部桁あふれﾁｪｯｸ
※1
RYEXP ＞ RXEXP
No
Yes
・RWEXPの補数作成
・REGX ⇔ REGY
ﾚｼﾞｽﾀ交換有り
指数差 ≧ 12
No
Yes
ﾚｼﾞｽﾀ交換有り
No
Yes
RYEXP ← RXEXP
REGY(仮数部) ← REGX(仮数部)
ﾚｼﾞｽﾀ交換無し
RXSIGN ← RWSIGN
return
指数差-1
指数差無し
Yes
No
ShiftDownRegY
※1：ｵｰﾊﾞﾌﾛｰ&ｱﾝﾀﾞﾌﾛｰの
ﾁｪｯｸを行う
RYEXP ← RXEXP
REGY(仮数部) ⇔ REGX(仮数部)
ﾚｼﾞｽﾀ交換ﾌﾗｸﾞ反転
RWSIGN ≠ RXSIGN
No
Yes
REGX – REGY : 仮数部減算
REGX + REGY : 仮数部加算
ボロー
No
Yes
ClearRegW
ExchangeRegX_W
SubtractBCD
ﾚｼﾞｽﾀ交換2回実施
No
Yes
RXSIGN反転
RXSIGN 反転
仮数部桁あふれ
No
Yes
ShiftDownRegX
RXEXP+1
桁あふれｴﾗｰﾌﾗｸﾞｾｯﾄ
Yes
指数部桁あふれ ※2
No
※2：ｵｰﾊﾞﾌﾛｰの
判断を行う
ShiftUpRegX
RXEXP ← RXEXP-1
Normalize
return

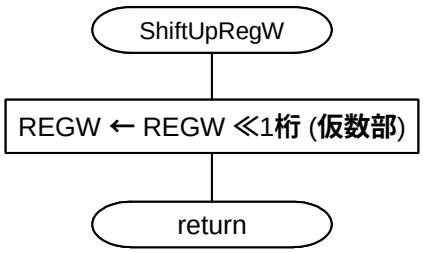

FloatingPointSubtract
RWSIGN ← -(RYSIGN)
ﾚｼﾞｽﾀ交換無し
RWEXP ← RXEXP - RYEXP
指数部桁あふれﾁｪｯｸ
※1
RYEXP ＞ RXEXP
No
Yes
・RWEXPの補数作成
・REGX ⇔ REGY
ﾚｼﾞｽﾀ交換有り
指数差 ≧ 12
No
Yes
ﾚｼﾞｽﾀ交換有り
No
Yes
RYEXP ← RXEXP
REGY(仮数部) ← REGX(仮数部)
ﾚｼﾞｽﾀ交換無し
RXSIGN ← RWSIGN
return
指数差-1
指数差無し
Yes
No
ShiftDownRegY
※1：ｵｰﾊﾞﾌﾛｰ&ｱﾝﾀﾞﾌﾛｰの
ﾁｪｯｸを行う
RYEXP ← RXEXP
REGY(仮数部) ⇔ REGX(仮数部)
ﾚｼﾞｽﾀ交換ﾌﾗｸﾞ反転
RWSIGN ≠ RXSIGN
No
Yes
REGX – REGY : 仮数部減算
REGX + REGY : 仮数部加算
ボロー
No
Yes
ClearRegW
ExchangeRegX_W
SubtractBCD
RXSIGN 反転
ﾚｼﾞｽﾀ交換2回実施
No
Yes
RXSIGN反転
仮数部桁あふれ
No
Yes
ShiftDownRegX
RXEXP+1
桁あふれｴﾗｰﾌﾗｸﾞｾｯﾄ
Yes
指数部桁あふれ ※2
※2：ｵｰﾊﾞﾌﾛｰの
判断を行う
No
ShiftUpRegX
RXEXP ← RXEXP-1
Normalize
return

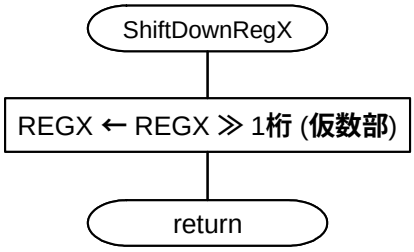

FloatingPointMulti
RWSIGN ← RXSIGN ^ RYSIGN
RXEXP ← RXEXP + RYEXP
指数部桁あふれ ﾁｪｯｸ
※
※ ｵｰﾊﾞﾌﾛｰ&ｱﾝﾀﾞﾌﾛｰ のﾁｪｯｸ
指数部桁あふれ
Yes
No
ExchangeRegX_W
桁あふれｴﾗｰﾌﾗｸﾞｾｯﾄ
return
RWEXP ← RXEXP
ClearRegX
RXEXP ← RWEXP
12桁カウンタ設定
全桁計算完了
No
Yes
ShiftDownRegX
RXSIGN ← RWSIGN
ShiftDownRegW
ShiftDownRegX
REGW最下位桁-1
RXEXP+1
REGW最下位桁=0
Yes
Normalize
No
return
AdditionBCD

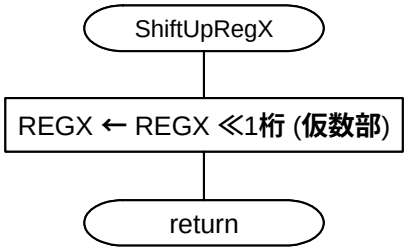

FloatingPointDivide
除数0ｴﾗｰﾌﾗｸﾞ ｸﾘｱ
RXSIGN ← RXSIGN ^ RYSIGN
除数(仮数部)=0
Yes
No
除数0ｴﾗｰﾌﾗｸﾞ ｾｯﾄ
return
RXEXP ← RXEXP - RYEXP
指数部桁あふれ ﾁｪｯｸ
※
指数部桁あふれ
Yes
No
※ ｵｰﾊﾞﾌﾛｰ&ｱﾝﾀﾞﾌﾛｰのﾁｪｯｸ
ClearRegW
桁あふれｴﾗｰﾌﾗｸﾞｾｯﾄ
return
12桁カウンタ設定
全桁計算完了
No
Yes
ShiftUpRegW
ExchangeRegX_W
SubtractBCD
Normalize
ボロー
No
Yes
return
AdditionBCD
RWLSD(商)+1
ShiftUpRegX

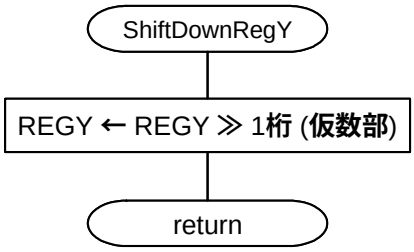

Regularization
仮数部(REGX)=0
Yes
No
ClearRegX
ｵｰﾊﾞﾌﾛｰｴﾗｰ ｾｯﾄ
RXMSD=0
No
Yes
return
ShiftUpRegX
RXEXP ← RXEXP-1
指数部桁あふれ ﾁｪｯｸ
※ ｱﾝﾀﾞｰﾌﾛｰのﾁｪｯｸ
※
指数部桁あふれ
Yes
No
桁あふれｴﾗｰﾌﾗｸﾞｾｯﾄ
ShiftDownRegX
RXEXP ← RXEXP+1
return

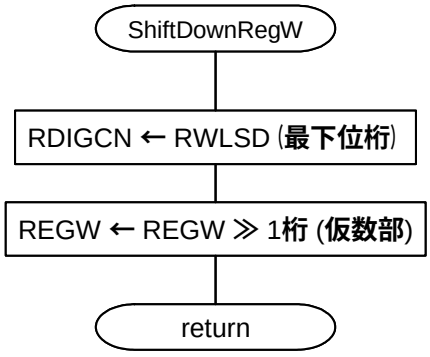

AdditionBCD
・REGX ← REGX + REGY
・BCD補正
return
SubtractBCD
・REGX ← REGX – ucReg
・BCD補正
return

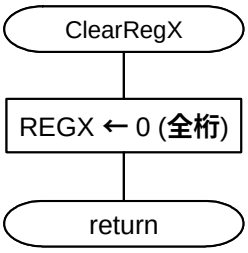

ShiftUpRegW
REGW ← REGW ≪1桁 (仮数部)
return

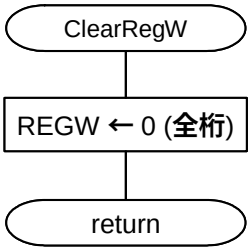

ShiftDownRegX
REGX ← REGX ≫ 1桁 (仮数部)
return

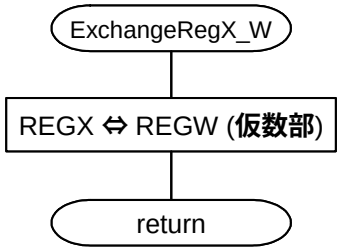

ShiftUpRegX
REGX ← REGX ≪1桁 (仮数部)
return


ShiftDownRegY
REGY ← REGY ≫ 1桁 (仮数部)
return


ShiftDownRegW
RDIGCN ← RWLSD (最下位桁)
REGW ← REGW ≫ 1桁 (仮数部)
return


ClearRegX
REGX ← 0 (全桁)
return


ClearRegW
REGW ← 0 (全桁)
return


ExchangeRegX_W
REGX ⇔ REGW (仮数部)
return

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
*
*                                    Copyright(C) NEC Electronics Corporation
*********************************************************************************
*/

#pragma ioreg

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


/*** プロトタイプ宣言 ***/
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 ^ CSIGHN;       /* 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 scHoldData;
    UCHAR ucHoldData;
    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 ){             /* RYEXP > RXEXP ? */
        scRwExp ^= 0xff;                            /* 指数差の補数をとる */
        scRwExp += 1;

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

```
        scRxExp    = scRyExp;
        scRyExp    = scHoldData;
        for( i=0; i<CDIGMAX; i++ ){                 /* REGX,REGY の仮数部を交換 */
            ucHoldData = ucRx[i];
            ucRx[i]    = ucRy[i];
            ucRy[i]    = ucHoldData;
        }

        ucFExChg = 1;                               /* レジスタ交換フラグをセット */
    }

    if( scRwExp < CEXOVER ){                        /* 指数差 < 12 ? */
        /*** 仮数部調整 ***/
        while(1){
            scRwExp -= 1;                           /* 指数差デクリメント */
            if( scRwExp >= 0 ){
                ShiftDownRegY();
            }else{
                scRyExp = scRxExp;                  /* RYEXP ← RXEXP */

                for( i=0; i<CDIGMAX; i++){          /* REGX,REGY を交換(仮数部) */
                    ucHoldData = ucRx[i];
                    ucRx[i]    = ucRy[i];
                    ucRy[i]    = ucHoldData;
                }

                ucFExChg ^= 1;                      /* ﾚｼﾞｽﾀ交換ﾌﾗｸﾞを変更(1 度交換していれば 0 それ以外は 1
に設定) */

             /*** 符号チェック ***/
                if( scRwSign != scRxSign ){
                    /*** 仮数部計算(異符号時) ***/
                    ucBorrow = SubtractBCD(ucRy);   /* 仮数部減算(REGX-REGY) */
                    if( ucBorrow == 1 ){            /* 減算でボローがでた? */
                        ClearRegW();                /* REGW をゼロクリア */
                        ExchangeRegX_W();           /* REGX,REGW を交換(仮数部) */

                        SubtractBCD(ucRw);          /* 仮数部減算(REGX-REGW), 仮数部の補数をとる */

                        if( ucFExChg == 0 ){
                            scRxSign ^= CSIGHN;     /* 符号反転 */
                        }

                    }else{
                        scRxSign ^= CSIGHN;         /* 符号反転 */
                    }
                }else{
                    /*** 仮数部計算 (同符号時) ***/
                    AdditionBCD();
                }
                /*** 正規化 ***/
                if( ucRxWorkReg > 0 ){
                    ShiftDownRegX();                /* REGX を 1 桁ダウン・シフト */
                    if( scRxExp >= 0 ){             /* 加算のオーバーフローチェック */
                        scRxExp++;                  /* 指数加算 */
                        if( scRxExp < 0 ){
                            ucFOver = 1;
                            ShiftUpRegX();
                            scRxExp--;
                            return;
                        }
                    }else{
                        scRxExp++;                  /* 指数加算 */
                    }
                }
                Normalize();                        /* 正規化 */
                break;
            }
        }
    }else{
```

```
        if( ucFExChg == 1 ){                        /* レジスタを交換した？ */
            /*** 転送(REGX → REGY) ***/
            scRyExp = scRxExp;

            for( i=0; i<CDIGMAX; i++ ){             /* REGY に元の値を復帰 */
                ucRy[i] = ucRx[i];
            }

            ucFExChg = 0;                           /* レジスタ交換フラグクリア */
            scRxSign = scRwSign;                    /* 退避していた結果符号を格納 */
        }
    }
    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;                    /* 指数加算 */
    if( (scRxExp >= 0) && (scRyExp >= 0) ){         /* 乗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) ){     /* 乗算のアンダーフローチェック */
        if( scRwExp >= 0 ){
            ucFUnder = 1;
            return;
        }
    }
    scRxExp = scRwExp;

    /*** 繰り返し加算カウンタ作成 ***/
    ExchangeRegX_W();                               /* REGX,REGW を交換(仮数部) */

    scRwExp = scRxExp;                              /* 指数退避 */
    ClearRegX();                                    /* REGX をゼロクリア */
    scRxExp = scRwExp;                              /* 指数復帰 */
    scRwExp = CDIGIT;                               /* 仮数部 10 桁カウンタ */

    while(1){
        /*** 乗算 ***/
        scRwExp -= 1;
        if( scRwExp >= 0   ){
            ShiftDownRegX();                        /* REGX を 1 桁ダウン・シフト */
            ShiftDownRegW();                        /* REGW を 1 桁ダウン・シフト */
            while( scRdigCn > 0 ){
                scRdigCn -= 1;                      /* RWEXP+1 = REGW 最下位桁 */
                AdditionBCD();                      /* 仮数部加算(REGX+REGY) */
            }
        }else{
            scRxSign = scRwSign;                    /* 結果符号復帰 */
            if( ucRxWorkReg > 0 ){
                ShiftDownRegX();                    /* REGX を 1 桁ダウン・シフト */
                scRxExp++;                          /* 指数加算 */
            }
            Normalize();                            /* 正規化 */
            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 += 0b0000110;
            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)
```