AVRc00：平方根演算

序説

本書はAVR446応用記述で使用されている平方根演算について説明します。

理屈

概要

この平方根演算は基本的に人が平方根を求めるのと同じ方法です。ここで人が平方根を求める手順を考察します。元の値は比較的大きな値の整数値と仮定します。従って一見で結果の平方根値を予測することができません。そこでこれと思われる結果の予測値を2乗して元の値と比較し、その大小の結果によって次の結果の平方根値を予測します。このようにして徐々に元値の最近値を求めて行きます。

更に予測するのが困難な値の場合はどうでしょう。現実ではそのような場合には電卓を探した方が早そうです。電卓もﾊﾟｿｺﾝも無ければ人力で行うしかありません。そこで上記の方法から予測を排除したやり方を考慮します。最初の予測値は充分に大きな値、例えば元値の半分から始めます。結果の平方根値の2乗が元値なので、この元値の半分とか、元値そのもののような値はその2乗値が必ず元値よりも大きくなります。そこでより小さな値を仮定して最近似値を求めます。ここでの要点は最初の予測値を確実な大きな値(または小さな値)とすることで、試行方向を1方向にできることです。

次は2回目以降の効率的な試行方法です。ここでも闇雲な予測の排除を考えます。そこで各桁毎での判定を思い付きます。例えば、元値が12345の場合、10000の桁に注目し、これで少なくても結果の平方根値は100以上でそこそこ100に近い値であることが分かります。ここでは10000の平方根が100であるのが容易に分かることが重要で、例えば1000の平方根は31.622～なのでこれは適しません。故にこれは結果(平方根値)での1桁毎に通じます。例えば前記の比較値10000に対応する平方根=100の1桁小さい桁は10(比較値は100)となり、従って元値と比較する仮定平方根値の2乗値では2桁毎になります。ここでの要点は桁毎に比較して結果の平方根値の桁を確定して行くことと、この桁毎の意味が平方根値側で1桁毎、元値または比較値(仮定平方根値の2乗値)で2桁毎になることです。

上記の方法をﾌﾟﾛｸﾞﾗﾑとする場合、最初の比較値(仮定平方根値の2乗値)を探すことよりも、有り得る最大値から始めた方がｺｰﾄﾞ量と実行速度の両方共に有利なので、これが上記と異なります。また全ての処理は2進数として扱われ、故に桁毎の処理がより容易になります。これらを考慮してﾌﾟﾛｸﾞﾗﾑ化する場合の手順は以下のようになります。

1. 元値との初期桁基準値として、その型での最大値を仮定します。この最大値は結果の最大平方根値の最上位桁(ﾋﾞｯﾄ)だけが1の値で、且つその値の2乗値がその型の範囲内になる最大値です。例えばﾊﾞｲﾄ(uchar)の場合の最大値は10000000ではなく01000000です。この場合の結果の平方根最大値は$10ですが、これは2乗した場合に$100となり、型範囲を超えます。故に単一桁(ﾋﾞｯﾄ)が1の値は$08で、この2乗値が01000000です。
2. 比較値(桁基準値+中間結果値)を元値(初回は元値そのもの、2回目以降は元値の残り)と比較し、元値が比較値よりも大きいかまたは同じ(今回比較桁が有り(=1))なら、元値から比較値を減じ、中間結果を1/2(1桁右移動)してその中間結果に桁基準値を加えます。さもなければ(今回比較桁がなし(=0))なら、単に中間結果を1/2(1桁右移動)します。
3. 桁基準値を1/4(2桁右移動)します。これは対応する平方根値での1桁右移動の意味です。
4. 桁基準値が0になる(全桁比較終了)まで2.～3.を繰り返します。
5. 最終元値(元値の残り)が最終結果よりも大きければ、最終結果に1を加えます(10進数での四捨五入処理)。

この手順で注意しなければならないことは、比較が元値に基く値と比較値で行われ、この比較値が桁基準値と中間結果の加算であることです。そして桁基準値は平方根値の2乗値の意味を持つために1桁処理毎に2桁ずつ移動されます。また中間結果はかなり分かり辛く、これは最終的に平方根値そのものなのですが、処理中は比較値の桁基準値以外の部分の数値として使用されます。基本的に平方根値ですので1桁処理毎に1桁ずつ移動されます。中間結果として加算される桁基準値は1桁処理毎に2桁移動しますが、中間結果自身が1桁処理毎に1桁移動するので加算される桁基準値は最終的な結果の平方根値から見ると、1桁処理毎に1桁の移動になります。これに関係する桁移動の順番が重要なので(特にﾌﾟﾛｸﾞﾗﾑ化の場合に)注意してください。

HERO and heavy friends

8ﾋﾞｯﾄ AVR® マイクロ コントローラ

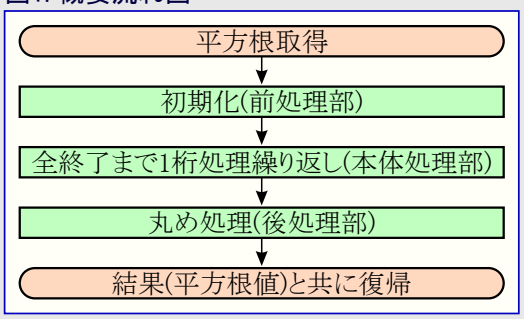
応用記述

Rev. AVRc00-01/14

詳細説明

平方根ルーチンの概要の流れ図が図1.に示されます。これはまず全体の流れを把握するための図です。全体的な流れは必要な初期値を設定する前処理(初期化)部、各桁毎の処理を繰り返す本体処理部、0捨1入の丸めを行う後処理部から成ります。

図1. 概要流れ図



初期化(前処理部)

前処理部となる初期化部で実際に初期化を必要とする変数は次の2つです。

・各桁毎の処理でその桁の基準となる桁基準値
・最終結果と中間結果を保持する結果変数

桁基準値変数は先に言及されたようにｺｰﾄﾞ量と実行速度を最適化するために初期値を検索して初期化するのでなく、その型で可能な最大値に初期化します。この値は後述されるように各桁(ﾋﾞｯﾄ)での大小判定の基になる2のべき乗値ですが、これの元はあくまでも結果としての平方根値でのことであり、この桁基準値変数は元値と直接的に比較する関係でその値の更に2乗された値です。従って初期値は結果としての平方根値で以下の両条件を満足する最大値の2乗値になります。

・1ﾋﾞｯﾄだけが1で残り全てのﾋﾞｯﾄが0である値
・2乗値が元値の型の範囲内である値

例えば元値がﾊﾞｲﾄ(uchar)の場合に於ける結果の平方根値の最大は$10ですが、これは2乗した場合に$100となり、型範囲を超えます。そこで前記条件の前者に従った次に大きな値である$08を2乗してみると、これは$40となり、型範囲内で両条件を満足します。従ってﾊﾞｲﾄ(uchar)の場合の初期桁基準値は$40(01000000)になります。結果だけを別の言葉で表現すると、桁基準値の初期値は元値の型に於ける最上位ﾋﾞｯﾄの1つ手前のﾋﾞｯﾄだけが1で、他の全ﾋﾞｯﾄが0の値です。この変数は本体処理部で各桁処理毎に1/4(結果としての平方根値で1/2、即ち1桁(ﾋﾞｯﾄ)右移動)されます。

これに対して結果変数は非常に明快で、単に0で初期化するだけです。この変数は本体処理部での中間結果(とその最後での丸め前の結果)を保持します。各桁処理毎に於いて、元値の残りと比較値の比較結果によってその時の桁基準値が加算されます。

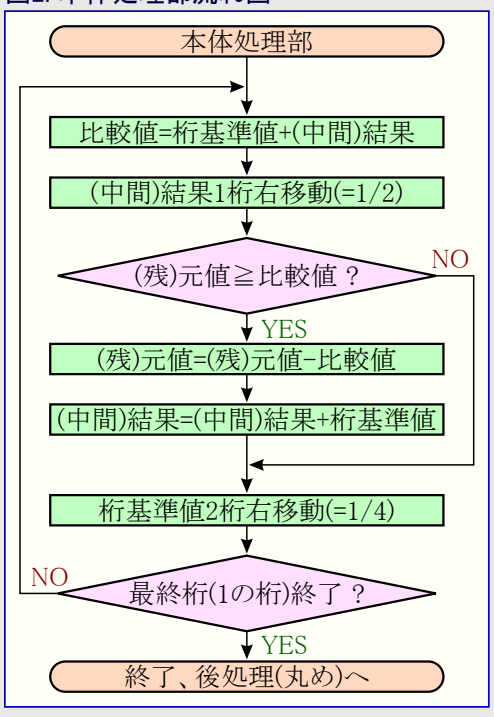
桁単位処理繰り返し(本体処理部)

ここでは初期桁基準値から始めて全桁が終了するまで、桁毎に比較とその関連処理を繰り返します。各桁処理は前で簡単に記述したように現在の元値の残りが比較値(桁基準値+中間結果値)と同じかそれよりも大きい場合に、中間結果へ桁基準値を加算して元値の残りから比較値を引きます。この処理の流れ図は図2.で、具体例は次頁の図4.で図解されます。

比較値は桁基準値+中間結果値で得ますが、桁基準値は今回の桁での基準値で、中間結果は前回までの桁処理に於けるその桁に基く比較結果(1/0)です。ここは前に言及したように分かり難いところですが、或る桁処理で元値の残りが大きい場合に中間結果に桁基準値が加算されます。これはこの桁に基く比較に於いて、結果(平方根値)の対応ﾋﾞｯﾄが1であることを示します。桁基準値は元が結果としての平方根の2乗値として扱うので、平方根値での1桁(ﾋﾞｯﾄ)に対応する各桁処理毎に2桁(ﾋﾞｯﾄ)ずつ右移動されます。中間結果自体は各桁処理毎に1桁(ﾋﾞｯﾄ)ずつ右移動され、結果として中間結果のその桁処理での桁に対応するﾋﾞｯﾄの上位(左)側にはそれ以前の桁での比較結果が並びます。図4.の具体例を参照してください。

繰り返しに関しては計数器を用いて行うこともができますが、丁度繰り返し終了時に桁基準値が0になるので、これを利用します。

図2. 本体処理部流れ図

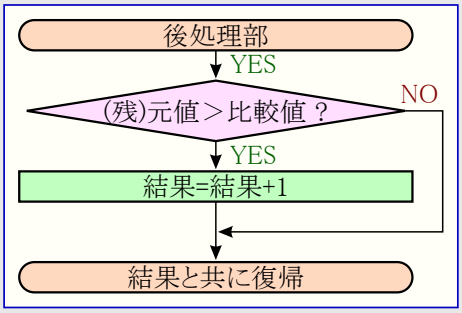


丸め処理(後処理部)

後処理部は単純な丸め(0捨1入、10進数での四捨五入)処理だけです。流れ図は図3.で示されます。ここの比較は本体処理部での比較と同一でその延長となる桁での比較です。ﾌﾟﾛｸﾞﾗﾑ作成上で異なるのは比較値の元となる桁基準値と中間結果の内、桁基準値が既に0なので、結果として元値の残りと暫定結果値との比較になることです。

この比較に於いて元値の残りの方が小さければ、暫定結果が最終結果となり、そのまま処理を終了します。そうでなければ、暫定結果に1を加えて最終結果とします。

図3. 後処理部流れ図



処理の推移

以下は元値がﾊﾞｲﾄ(uchar)型の255の場合に於ける各処理と各変数の流れです。255の場合は各桁処理の全てに於いて比較で元値の残りの方が大きくなり、更に0捨1入での切り上げ処理もあります。

図4. 概要流れ図　元値(の残り)　桁基準値　(中間)結果　比較値　定数

1-1.	0	1	0	0	0	0	0	0	+	0	0	0	0	0	0	0	0	=	0	1	0	0	0	0	0	0	比較値取得 (桁基準値+中間結果値)
1-2.	0	0	0	0	0	0	0	0	—— 1桁(ﾋﾞｯﾄ)右移動 ——→									0	0	0	0	0	0	0	0	中間結果1桁右移動(=1/2)
1-3.	1	1	1	1	1	1	1	1	←—— 大小比較 ——→									0	1	0	0	0	0	0	0	元値と比較値の比較、元値<で1-6.へ
1-4.	1	1	1	1	1	1	1	1	−	0	1	0	0	0	0	0	0	=	1	0	1	1	1	1	1	1	元値の残り取得 (元値−比較値)
1-5.	0	0	0	0	0	0	0	0	+	0	1	0	0	0	0	0	0	=	0	1	0	0	0	0	0	0	中間結果更新 (=中間結果+桁基準値)
1-6.	0	1	0	0	0	0	0	0	—— 2桁(ﾋﾞｯﾄ)右移動 ——→									0	0	0	1	0	0	0	0	桁基準値2桁右移動(=1/4)
2-1.	0	0	0	1	0	0	0	0	+	0	1	0	0	0	0	0	0	=	0	1	0	1	0	0	0	0	比較値取得 (桁基準値+中間結果値)
2-2.	0	1	0	0	0	0	0	0	—— 1桁(ﾋﾞｯﾄ)右移動 ——→									0	0	1	0	0	0	0	0	中間結果1桁右移動(=1/2)
2-3.	1	0	1	1	1	1	1	1	←—— 大小比較 ——→									0	1	0	1	0	0	0	0	元値の残りと比較値の比較、元値<で2-6.へ
2-4.	1	0	1	1	1	1	1	1	−	0	1	0	1	0	0	0	0	=	0	1	1	0	1	1	1	1	元値の残り値更新 (元値の残り−比較値)
2-5.	0	0	1	0	0	0	0	0	+	0	0	0	1	0	0	0	0	=	0	0	1	1	0	0	0	0	中間結果更新 (=中間結果+桁基準値)
2-6.	0	0	0	1	0	0	0	0	—— 2桁(ﾋﾞｯﾄ)右移動 ——→									0	0	0	0	0	1	0	0	桁基準値2桁右移動(=1/4)
3-1.	0	0	0	0	0	1	0	0	+	0	0	1	1	0	0	0	0	=	0	0	1	1	0	1	0	0	比較値取得 (桁基準値+中間結果値)
3-2.	0	0	1	1	0	0	0	0	—— 1桁(ﾋﾞｯﾄ)右移動 ——→									0	0	0	1	1	0	0	0	中間結果1桁右移動(=1/2)
3-3.	0	1	1	0	1	1	1	1	←—— 大小比較 ——→									0	0	1	1	0	1	0	0	元値の残りと比較値の比較、元値<で3-6.へ
3-4.	0	1	1	0	1	1	1	1	−	0	0	1	1	0	1	0	0	=	0	0	1	1	1	0	1	1	元値の残り値更新 (元値の残り−比較値)
3-5.	0	0	0	1	1	0	0	0	+	0	0	0	0	0	1	0	0	=	0	0	0	1	1	1	0	0	中間結果更新 (=中間結果+桁基準値)
3-6.	0	0	0	0	0	1	0	0	—— 2桁(ﾋﾞｯﾄ)右移動 ——→									0	0	0	0	0	0	0	1	桁基準値2桁右移動(=1/4)
4-1.	0	0	0	0	0	0	0	1	+	0	0	0	1	1	1	0	0	=	0	0	0	1	1	1	0	1	比較値取得 (桁基準値+中間結果値)
4-2.	0	0	0	1	1	1	0	0	—— 1桁(ﾋﾞｯﾄ)右移動 ——→									0	0	0	0	1	1	1	0	中間結果1桁右移動(=1/2)
4-3.	0	0	1	1	1	0	1	1	←—— 大小比較 ——→									0	0	0	1	1	1	0	1	元値の残りと比較値の比較、元値<で4-6.へ
4-4.	0	0	1	1	1	0	1	1	−	0	0	0	1	1	1	0	1	=	0	0	0	1	1	1	1	0	元値の残り値更新 (元値の残り−比較値)
4-5.	0	0	0	0	1	1	0	0	+	0	0	0	0	0	0	0	1	=	0	0	0	0	1	1	1	1	中間結果更新 (=中間結果+桁基準値)
4-6.	0	0	0	0	0	0	0	1	—— 2桁(ﾋﾞｯﾄ)右移動 ——→									0	0	0	0	0	0	0	0	桁基準値2桁右移動(=1/4)
	丸め(0捨1入)																										
5-1.	0	0	0	0	0	0	0	0	+	0	0	0	0	1	1	1	1	=	0	0	0	0	1	1	1	1	比較値取得 (桁基準値+中間結果値)
5-2.	0	0	0	1	1	1	1	0	←—— 大小比較 ——→									0	0	0	0	1	1	1	1	元値の残りと比較値の比較、元値<で終了へ
5-3.	0	0	0	0	1	1	1	1	+	0	0	0	0	0	0	0	1	=	0	0	0	1	0	0	0	0	0捨1入による切り上げ

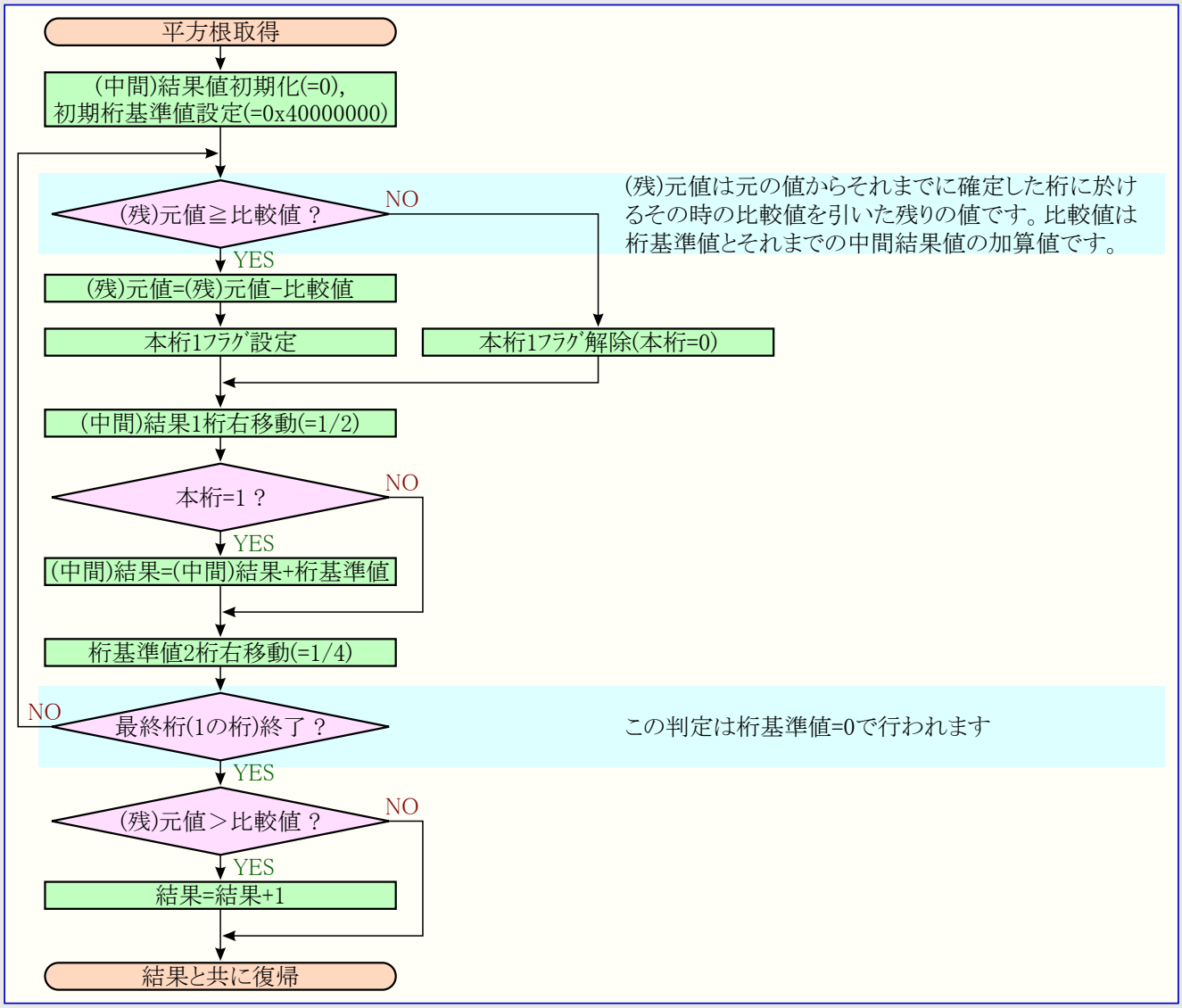
実装

本書ではAVR446応用記述で使用されているC言語の例(流れ図)と、基本的にそれと対応するｱｾﾝﾌﾞﾘ言語での例を示します。

C言語での例

C言語での具体的な例はAVR446応用記述のspeed_cntr.cでのsqrt(unsigned long x)関数です。ここではこれに対応する流れ図を示します。

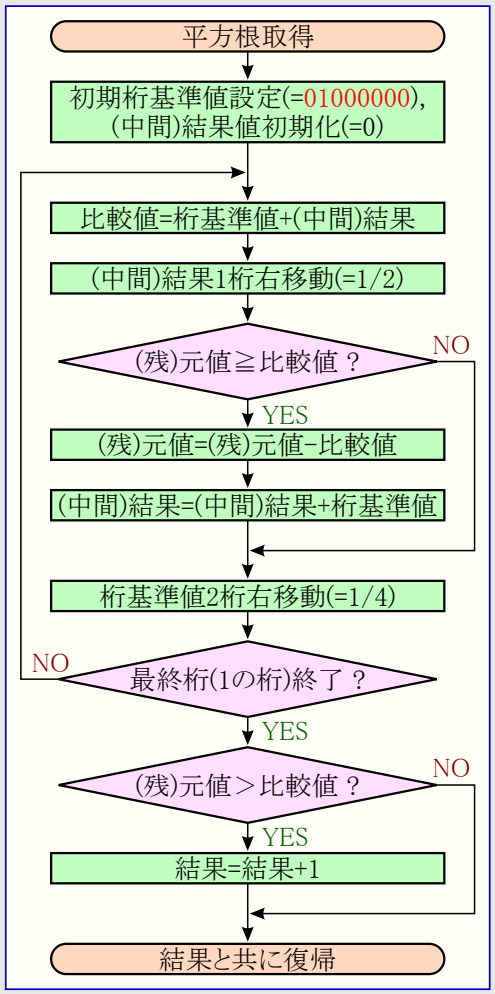
図1. sqrt()の流れ図



ｱｾﾝﾌﾞﾘ言語での例

ｱｾﾝﾌﾞﾘ言語例ではより効率的なｺｰﾄﾞ量と速度のために、C言語例の流れ図をｱｾﾝﾌﾞﾘ言語向けに変更しています。大きな変更点は処理順序の入れ替えによって現在処理中の桁の結果を保持するﾌﾗｸﾞを取り去り、流れの中の処理に一元化したことです。以下にその流れ図と符号なしﾊﾞｲﾄ整数でのﾌﾟﾛｸﾞﾗﾑ例を示します。多倍長への拡張については同梱ｿｰｽ内の16ﾋﾞｯﾄ(ﾜｰﾄﾞ)と32ﾋﾞｯﾄ(ﾛﾝｸﾞ)の平方根例を参考にしてください。

図2. "SQRT"の流れ図




```
SQRT:   LDI     R17,0b01000000  ;桁基準値初期化(初期桁基準値)
        CLR     R15             ;(中間及び最終の)結果値初期化
SQRT1:  MOV     R14,R17         ;現桁基準値複写
        ADD     R14,R15         ;比較値(桁基準値+(中間)結果)取得
        LSR     R15             ;中間結果1桁右移動
        CP      R14,R16         ;対応桁1/0判定
        BRSH    SQRT2           ;残元値＜比較値(=対応桁が0)で分岐
;
        SUB     R16,R14         ;新残元値=旧残元値-現比較値
        ADD     R15,R17         ;(中間)結果に現桁基準値加算
SQRT2:  LSR     R17             ;比較基準値2桁右移動
        LSR     R17             ;(結果値としての1桁分)
        BRNE    SQRT1           ;1の桁終了まで継続へ
;
        CP      R15,R16         ;丸め判定(残元値>最終結果でCF=1)
        ADC     R15,R17         ;四捨五入丸め(CF加算,R17は0)
        RET                     ;呼び出し元へ復帰
```

使用ﾚｼﾞｽﾀについては下の**表1.**をご覧ください。ﾌﾟﾛｸﾞﾗﾑの下から5行目のBRNE命令は最終桁終了時にR17が必ず0になることによって繰り返しの終了を判別します。下から3行目と2行目は10進数で言うところの四捨五入(端数丸め)処理です。下から3行目で判定結果をｷｬﾘｰ ﾌﾗｸﾞに反映させ、同じく2行目でｷｬﾘｰ ﾌﾗｸﾞを含めて0と加算することによってこれを行っています。AVRが即値加算命令を持たないので、直前に必ず0になっているR17を加算相手として使用しています。

表1. "SQRTB"使用ﾚｼﾞｽﾀ

ﾚｼﾞｽﾀ	入力	内部	出力
R14		比較値	
R15		中間結果値	最終結果(平方根)
R16	元値		
R17		桁基準値	

表2. "SQRT"性能値

項目	SQRTB (ﾊﾞｲﾄ長)	SQRTW (ﾜｰﾄﾞ長)	SQRTL (ﾛﾝｸﾞ長)
ｺｰﾄﾞ量(語)	14+RET	26+RET	48+RET
実行時間	43～47+RET	133～157+RET	441～553+RET

HERO and heavy friends

本書はAVRの応用に関する補助情報AVRc00の記録です。

青字の部分はﾘﾝｸとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。