29. 付録

29.1 R8Cマイコンの変数のサイズ

C言語は”手続き型言語”と呼ばれ、変数(データの入れ物)は初めに用意し、しかも大きさも決めておかなくてはなりません。

```
void main( void )
{
  int a,b,c ;
  a = 100 ; b = 2000 ;
  c = a * b ;
  printf("¥n c = %d ",c) ;
}
```

画面にはどのように表示されるでしょうか。200000 と正確に表示されるものもあれば 3392 と、とんでもない表示をするものもあります。実は int 型の大きさは”処理系に依存する”と言う言葉で表現されており何ビットであるかは決まっていないのです。C コンパイラを作成した開発者に任されている部分なのです。

R8C マイコンのコンパイラで結果をだすと 3392 になります(65,536×3＋3,392＝200,000)。

コンパイラは、桁あふれしたかどうかの確認を取るようなことはしません。データの取り扱いはプログラマに任されています。

R8C のコンパイラは以下のようになっています。

●整数型(R8C マイコンのコンパイラの場合)

型	値の範囲	データサイズ
char (unsigned char 型として扱われる) **※H8 は、signed char 型として扱われます。 R8C と H8 では異なります。**	0 ～ 255	1 バイト
signed char	-128 ～ 127	1 バイト
unsigned char	0 ～ 255	1 バイト
short	-32768 ～ 32767	2 バイト
unsigned short	0 ～ 65535	2 バイト
int	-32768 ～ 32767	2 バイト
unsigned int	0 ～ 65535	2 バイト
long	-2147483648 ～ 2147483647	4 バイト
unsigned long	0 ～ 4294967295	4 バイト

●実数型(R8C マイコンのコンパイラの場合)

型	データサイズ	限界値	
		最大値	正の最小値
float	4 バイト	3.4028235677973364e+38f (0x7F7FFFFF)	7.0064923216240862e-46f (0x00000001)
double long double	8 バイト	1.7976931348623158e+308 (0x7FEFFFFFFFFFFFFF)	4.9406564584124655e-324 (0x0000000000000001)

29.2 演算子

演算子を用いると、値をいろいろと加工することができます。演算というのはちょっと硬い表現ですが、簡単に言えば、足す、引く、掛ける、割るなどの計算です。

演算子の機能を下記に示します。

演算子		機　能	備　考	例
単項演算子	－	負		-a
	＋	正		+b
	～	ビットごとの反転	チルダと読む	~a
	－－	デクリメント		a--
	＋＋	インクリメント		b++
	＆	変数のアドレス	＆a　はaの変数が格納されているアドレス	&a
	＊	ポインタ変数の指す内容	＊p　はpの指す内容	*p
2項演算子	－	減算		a = b - c;
	＋	加算		a = b + c;
	＊	積		a = b * c;
	／	商		a = b / c;
	％	整数除算の余り		a = b % c;
	＆	ビットごとの論理積		a = b & 0x7f;
	｜	ビットごとの論理和		a = b | 0x80;
	＾	ビット毎の排他的論理和		a = b ^ 0x55;
	＆＆	論理積	答えは真か偽	if(a==b && c==d)
	｜｜	論理和	答えは真か偽	if(a==b || c==d)
	＞＞	右シフト	(変数名)＞＞(シフトするビット数)	a = a >> 2;
	＜＜	左シフト	(変数名)＜＜(シフトするビット数)	a = a << 2;
代入演算子	＝	代入		a = b;
	＋＝	加算して代入		a += b;
	－＝	減算して代入		a -= b;
	／＝	除算して代入		a /= b;
	％＝	剰余演算して代入		a %= b;
	＜＜＝	左シフト演算して代入		a <<= 5;
	＞＞＝	右シフト演算して代入		a >>= 2;
	＆＝	論理積して代入		a &= 0x55;
	｜＝	論理和して代入		a |= b;
	＾＝	排他的論理和して代入		a ^= b;
比較演算子	＝＝	等しい		if(a == b)
	！＝	等しくない		if(a != b)
	＞	より大きい		if(a > b)
	＜	より小さい		if(a < b)
	＞＝	より大きいか等しい		if(a >= b)
	＜＝	より小さいか等しい		if(a <= b)

29.3 優先順位

式は優先順位の高い順に評価され、同順位なら結合規則にしたがって、左→右または右→左に評価されます。優先順位を変えるには( )を用います。

優先順位 / 演算子	1 高	2	3	4	5	6	7	8	9	10	11	12	13	14	15 低
関数、カッコ	( )														
配列	[ ]														
構造体	. ->														
型		(型) sizeof													
ポインタ		* &													
インクリメント ／デクリメント		++ —													
算術		+ -	* / %	+ -	※優先順位２は、単項演算子　例）-a 優先順位４は、二項演算子　例）a-b										
関係						< <= > >=	== !=								
ビット		~			<< >>			&	^	|					
論理		!									&&	||			
条件													?:		
代入														= += *= など	
コンマ															,
結合規則	左→右	左←右	左　→　右										左←右		左→右
演算子 / 優先順位	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

29.4 型が混合したときの演算

違う型が混合した計算の場合、下記のような決まりで演算されます。

・char と unsigned char と short は int
・unsigned short は unsigned int
・float は double

} に変換され、

long double＞double＞unsigned long＞long＞unsigned int＞int
の優先度で型の高い方に変換されて演算されます。
（ただし、char＜short＝int とする）

29.5 printf関数の使い方

printf 関数の呼び出しは次の形式で記述します。

```
        ret = printf( fmt , arg1 , arg2 , ... ) ;

ただし  ret             :int 型。出力した文字数(エラー時は -1)。
        fmt             :char 型へのポインタ型。フォーマット変換を指定する文字列。
        arg1, arg2, ...  :定数または表示データの格納された変数や式(書式に依存)。
```

(A) printf 関数は、fmt で示される文字列をそのまま表示します。
ただし
(B) 文字列の中に「%」があると、それに続く文字により、2番目以降(arg1)の引数の示す値を変換し、変換結果を文字列に埋め込んで表示します。
(C) 「%」の後に続くものはオプションと変換指定文字です。

%[オプション]変換指定文字　　[ ]は省略可

(1)変換指定文字

変換指定文字	引数の型	表示のされ方
d	int	10 進数
o	int	8進数
x	int	16 進数
u	int	符号なし 10 進数
e	double	[-]m.nnnnnne[±]xx の形式の 10 進浮動小数点数 (n の桁数はオプションで可変だが標準では6桁)
f	double	[-]m.nnnnnn の形式の 10 進浮動小数点数 (n の桁数はオプションで可変だが標準では6桁)
c	int	単一文字
s	char *	文字列
p	void *	ポインタ
%		%

※printf 関数は多くのメモリを消費するため R8C マイコンでは、「%e, %E, %f, %g, %G」の変換指定文字は使用できません。

(2)オプション

l	:	引数を long 型のサイズとして扱う。なお、l は整数型に適用可能。
数値	:	出力幅の指定。変換結果の文字数が指定値より少ないときは右詰めとなる。 また、「数値」の左に「-」を付けると出力欄に左詰めされる。
「.」を含む数字列	:	浮動小数点形式に変換する場合の出力欄の幅と精度(小数点以下の桁数)を「.」で区切って示す。指定がなければ小数点以下 6 桁表示となる。

プログラム

```
 2: #include <stdio.h>
 3:
 4: void main( void )
 5: {
 6:   int    a = 64  ,  b = -64;
 7:   double c = 6.4;
 8:   char   data[20] = { "I Love You !" };
 9:
10:   printf("decimal     :%20d%20d¥n",a,b);
11:   printf("unsigned    :%20u%20u¥n",a,b);
12:   printf("octal       :%20o%20o¥n",a,b);
13:   printf("hexadecimal :%20x%20x¥n",a,b);
14:   printf("character   :%20c¥n",a);
15:   printf("double      :%20f¥n",c);
16:   printf("double - e  :%20e¥n",c);
17:   printf("string      :%20s¥n",data);
18: }
```

実行結果

```
decimal     :                  64                 -64
unsigned    :                  64               65472
octal       :                 100              177700
hexadecimal :                  40                ffc0
character   :                   @
double      :            6.400000
double - e  :        6.400000e+00
string      :        I Love You !
```

解説

2 行	printf 関数を使用するため「 stdio.h 」をインクルードします。
6～8 行	必要となる変数を初期値付きで宣言します。
10 行	変換指定 %d により 64 と -64 が表示されます。
11 行	変換指定 %u により -64 を符号なし 2 進数とみなして 10 進数に変換し、65472 が表示されます。
12 行	変換指定 %o により 64 と -64 の 8 進表現が表示されます。
13 行	変換指定 %x により 64 と -64 の 16 進表現が表示されます。
14 行	変換指定 %c により 64 を文字に変換し @ が表示されます。
15 行	変換指定 %f により 6.4 が小数点のみの形式で表示されます。
16 行	変換指定 %e により 6.4 が指数形式で表示されます。
17 行	変換指定 %s により char 型配列の内容が文字列として表示されます。

29.6 scanf関数の使い方

scanf 関数の呼び出しは次の形式で記述します。

```
        ret = scanf( fmt , arg1 , arg2 , ... ) ;

ただし  ret             :int 型。読み込んで変換されたデータの数(エラー時は -1)。
        fmt             :char 型へのポインタ型。フォーマット変換を指定する文字列。
        arg1,arg2,...   :変換したデータの格納先を示すアドレスや式(書式に依存)。
```

(A) キーボードから改行キーが入力されるまで文字をバッファに入力する。改行が入力されてはじめて変換作業が始まり、バッファから読み込まれて処理される。バッファの管理はライブラリ関数側で自動的に行われる。

(B) fmt が示す文字列中の「%」に続く「オプション」(省略可)と「変換指定文字」に従って変換を行い、結果を2番目以降(arg1)の引数が示す格納先に格納する。

(C) fmt が示す文字列は、原則として「%」と「変換指定文字」の列で構成する。ただし、例外的に「 % 」の付かない文字を挿入することがある。(F)参照

(D) 変換指定文字が「c」以外の場合、改行やスペースなどの非印字文字は区切り記号とみなされる。また、変換データよりも前にある場合は読み飛ばされ、後の場合はバッファ内に読み残される。したがって、文字列の読み込みの際にスペースも含めて入力したい場合、変換指定文字の「s」は使えない。
　変換結果が正常でない場合、例えば 10 進入力にもかかわらず数字以外のものが入力されていた場合は、その文字を読み込まずに作業は終了する。

(E) 変換指定文字が「c」の場合は、ASCII コードはすべて(改行やスペースも含め)処理の対象となる。そのため、それ以前の scanf 関数の処理によってバッファに読み残された非印字文字があれば、それを読んでしまうことになる。これを避けるため `" %c"` とスペースを挿入して記述する方法が用いられる。

(F) オプション「l」は long 型および double 型のデータを入力する場合に使用する。

(1)変換指定文字

変換指定文字	引数の型	入力データ
d	int *	10 進数
o	int *	8進数
x	int *	16 進数
f	float *	浮動小数点数
e	float *	浮動小数点数
c	char *	単一文字
s	char *	文字列(最後にヌルコードが付加される)
p	void *	ポインタ
[文字]	char *	入力データの中から[ ]内に出てくる文字のみを入力する。[ ]にない最初の文字で入力が終了し、この文字は入力されない
[^文字]	char *	入力データの中から[]内の文字のみを読み飛ばす

また、「%」と変換指定文字の間に「*」(代入抑止文字)を付けると、変換指定文字を「読み捨てる」という意味になります。

30. 参考文献

・ルネサス エレクトロニクス(株)
R8C/35A グループ ハードウェアマニュアル Rev.0.40
・ルネサス エレクトロニクス(株)
High-performance Embedded Workshop V.4.00 ユーザーズマニュアル Rev.3.00
・ルネサス半導体トレーニングセンター C言語入門コーステキスト 第 1 版
・電波新聞社 マイコン入門講座 大須賀威彦著 第 1 版
・ソフトバンク(株) 新C言語入門シニア編 林晴比古著 初版
・共立出版(株) プログラマのための ANSI C 全書 L.Ammeraal 著
吉田敬一・竹内淑子・吉田恵美子訳 初版

マイコンカーラリーについての詳しい情報は、マイコンカーラリー公式ホームページをご覧ください。

http://www.mcr.gr.jp/

R8C マイコンについての詳しい情報は、ルネサス エレクトロニクスのホームページをご覧ください。

http://japan.renesas.com/

の「製品情報」欄→「マイコン」→「R8C」でご覧頂けます

※リンクは、2010 年 6 月現在の情報です。