計測コラム　　emm86 号用

ディジタル計測の基礎 – 第 14 回「バイナリデータとその形式」

前回は A/D コンバータについてお話ししましたが、その続きとして今回は、バイナリデータ（2 進数データ）についてお話しします。

現在のコンピュータの世界で信号処理するデータは、すべて 2 進数（バイナリ）で処理されています。もちろん前回話した A/D コンバータの出力も 2 進数のディジタルデータです。

2 進数で数値データを表現する方法は、大きく分けて、固定小数点形式と浮動小数点形式があります。これらについて説明します。

固定小数点形式は、小数点をどこに固定するかで表現する数字が変わってきますが、とりあえず、小数点が右端にある、すなわち整数だけを考えます。一般に固定小数点形式の 2 進数表現は、以下の種類があります。

① ストレートバイナリ（符号なしバイナリ）
② オフセットバイナリ
③ 2 の補数バイナリ

①は、正の整数（自然数）と 0（零）だけ表すことができ、負の整数は表現できません。それに対して、②と③は正だけでなく負の整数も表現できます。

次ページ表 1 は、8 ビットの 2 進数数値を 3 種類のバイナリ表現を比較した例です。
8 ビットバイナリですから 256 $(= 2^8)$ の数値表現が可能ですが、①は 0 ～ 255 までの整数、②と③は、-128 ～ 127 までの整数を表現できます。また、②と③では、最上位ビット（MSB）は符号を表していて MSB を反転させることにより簡単に相互変換可能で実質同じ表現ですが、0（零）の表現において、③はすべてのビットが 0 となりわかりやすいので、コンピュータの世界では、2 の補数形式のバイナリ表現が一般的に使用されています。すなわち、2 の補数バイナリでは、正の数は MSB が 0、負の数は MSB が 1 、0（零）はすべてのビットが 0 となります。なお、A/D コンバータや D/A コンバータにおける入出力バイナリ表現では、②や③がよく用いられています。

10 進数	ストレートバイナリ	10 進数	オフセットバイナリ	10 進数	2 の補数バイナリ
255	1111 1111	127	1111 1111	127	0111 1111
254	1111 1110	126	1111 1110	126	0111 1110
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・
129	1000 0001	1	1000 0001	1	0000 0001
128	1000 0000	0	1000 0000	0	0000 0000
127	0111 1111	-1	0111 1111	-1	1111 1111
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・
・	・	・	・	・	・
0	0000 0000	-128	0000 0000	-128	1000 0000

表 1. 3 種類のバイナリ表現の比較

例えば、プログラム言語 C の変数において、整数型は以下があり、これも内部の表現は、2 の補数バイナリです。

整数型	バイト数	ビット数	数値範囲
int	4	32	-2,147,483,648 ～ 2,147,483,647
long	4	32	-2,147,483,648 ～ 2,147,483,647
short	2	16	-32,768 ～ 32,767
char	1	8	-128 ～ 127

表 2. 固定小数点表現の整数変数の数値範囲（C 言語の例）

（注 1）符号なし（Unsigned）整数もあります。

（注 2）最近の 64 ビット PC では、long long int（64 ビット）もあるそうです。

さて、固定小数点形式で、小数のある数値を表すためには、小数点を適当な桁に固定する必要があります。例えば、8 ビット（2 の補数形式）で 4 桁目と 5 桁目の間に小数点があるとすると（図 1）、10 進数で（5.625）$_{10}$ を 2 進数に変換すると、（0101.1010）$_2$ となり、この結果を小数点のない整数と見なすと、（01011010）$_2$ =（90）$_{10}$ となります。一般に 2 進数で小数点を 1 桁左に移動することは数値を 1/2 することに相当して、このケースでは 4 桁移動していることになりますので、90 を 16（= 2^4）で割ると 5.625 となることが分かります。

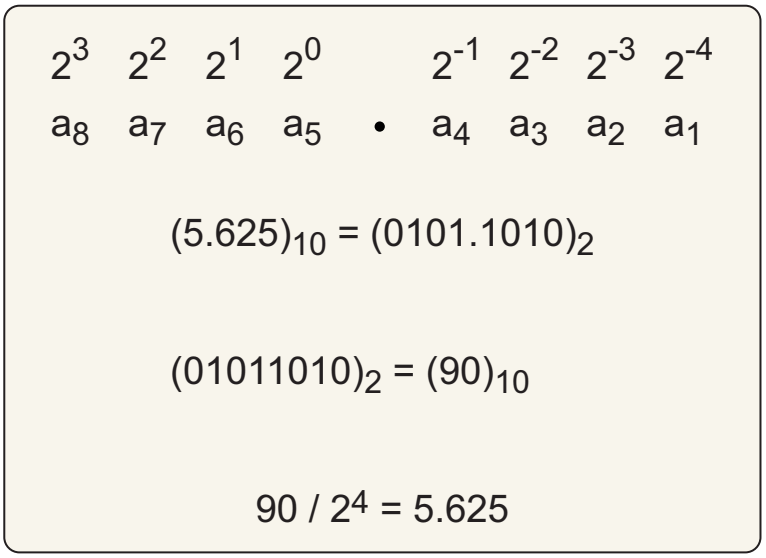

図 1. 小数点を含む固定小数点形式の例

このように、固定小数点形式で小数点の移動は非常に簡単な信号処理なので、小数点の位置は一番右（すなわち整数）と見なすか、または、一番左（すなわちすべて小数値、最大値が 1）と見なして、最終的に適当な数値に換算する方法が一般的です。
固定小数点形式は、① 演算処理が比較的簡単で高速演算に適する、② 表現できる数値範囲内では桁落ちが少ない、などのメリットがあるので、DSP を使った画像処理やディジタルフィルタ処理などによく用いられます。
デメリットしては、① 表現できる数値範囲が狭い（オーバフローしやすい）、② 小数を含む実数値を扱いにくい（小数点を常に考えてプログラムする必要がある）などがあり、複雑な信号処理では、あまり使われていません。

浮動小数点形式は、小数点を含む任意の実数をコンピュータ上で表現できる数値データ形式で、その表現形式は、以前は、コンピュータメーカや半導体メーカ各社でバラバラでしたが、最近は、IEEE754 規格で規定されている標準的な形式が最も広く採用されています。

その形式には、32 ビット（単精度）と 64 ビット（倍精度）の２つの表現形式があり、図 2 と図 3 に示します。また、表現できる数値範囲は表 3 のようになります。

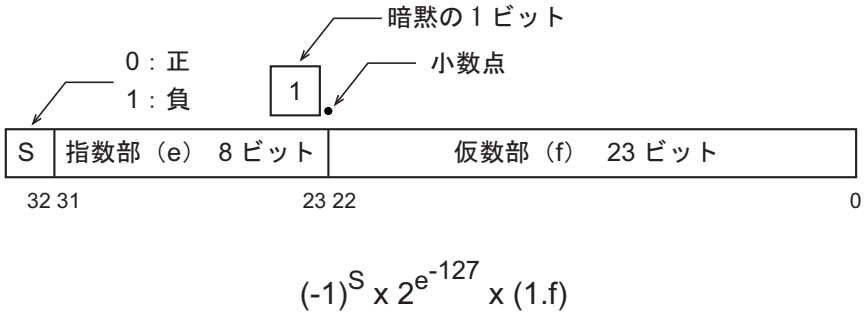

図 2. 浮動小数点（単精度）の表現形式

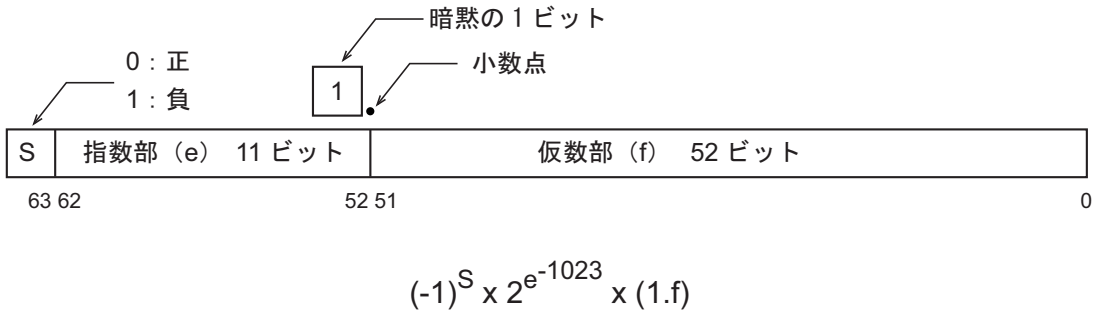

図 3. 浮動小数点（倍精度）の表現形式

	実数型 （C 言語）	最大の数 （絶対値）	最小の数 （絶対値）	有効数字 （10 進数）
単精度	float	約 3.40×10^{38}	約 1.18×10^{-38}	約 6 桁相当
倍精度	double	約 1.80×10^{308}	約 2.23×10^{-308}	約 16 桁相当

表 3. 浮動小数点形式で表現できる数値範囲

この形式は、仮数部と指数部とも絶対値で表現して、特に指数部は、バイアス（単精度は 127、倍精度は 1023）をのせた正の数値となっています。また、数値を正規化すると仮数部の最上位ビットは必ず 1 となる（0 でない限り）ことを利用して、最上位ビットを省略（暗黙の 1 ビット）した表現となっています。このような巧妙な方式に最初に出合ったのは、米国マイクロソフト社の MS Basic の浮動小数点形式で、筆者も非常に感心した記憶があります。

浮動小数点形式は、このように表現できる数値範囲が非常に大きく、演算オーバフローを気にせずプログラミングすることができるため、科学技術の計算や数値解析などに広く使われています。ただ、その結果の精度をそれなりに求めるならば、その誤差に関して留意して使用すべきです。普通の演算には、処理スピードの都合上単精度で良いのかもしれませんが、倍精度演算でその演算精度を確認するようにするべきです。
浮動小数点演算の誤差解析に関しては、筆者の能力を超えるため、お話しできませんが、代表的な弱点としては、「ほぼ同じ値の減算において有効数字が大きく減少する」という桁落ち現象があります。このような原理的な限界を考慮して、プログラミングするように心がけるべきと思います。

以上