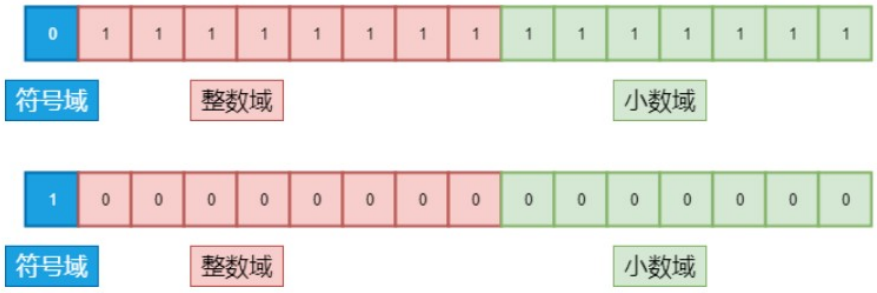
数据标幺化及定点计算

Q 格式：

Q1.31 格式中，数值由 1bit 符号位和 31bit 分数位构成，数值范围-1（0x80000000）到 $1-2^{-31}$（0X7FFFFFFF）；

Q1.15 格式中，数值由 1bit 符号位和 15bit 分数位构成，数值范围-1（0x8000）到 $1-2^{-15}$（0X7FFF）；



Q1.15 转 float：

判断最高位（16bit）是否为 1，再将低 15 位/32768.

float 转 Q1.15：

判断 float 正负，小数位乘 32768.

Q	Qmn	Max	Min
Q 15	Q 0.15	0.999969	-1.000000
Q 14	Q 1.14	1.999939	-2.000000
Q 13	Q 2.13	3.999878	-4.000000
Q 12	Q 3.12	7.999756	-8.000000
Q 11	Q 4.11	15.999512	-16.000000
Q 10	Q 5.10	31.999023	-32.000000
Q 9	Q 6.9	63.998047	-64.000000
Q 8	Q 7.8	127.996094	-128.000000
Q 7	Q 8.7	255.992188	-256.000000
Q 6	Q 9.6	511.984375	-512.000000
Q 5	Q 10.5	1023.968750	-1024.000000
Q 4	Q 11.4	2047.937500	-2048.000000
Q 3	Q 12.3	4095.875000	-4096.000000
Q 2	Q 13.2	8191.750000	-8192.000000
Q 1	Q 14.1	16383.500000	-16384.000000
Q 0	Q 15.0	32767.000000	-32768.000000

数据标幺化：

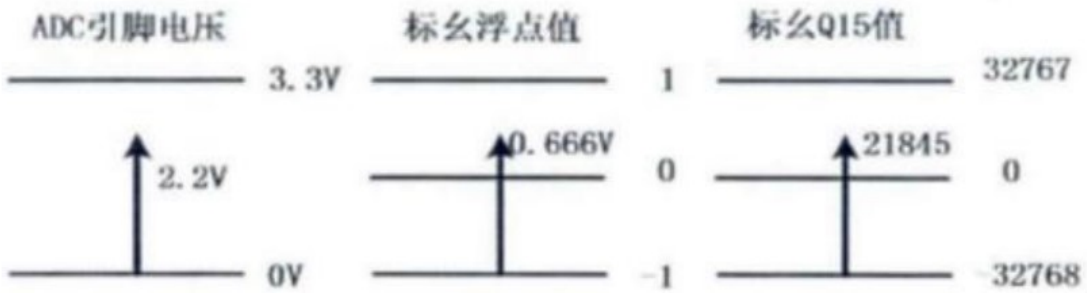
三角函数计算时，为加快计算速度，先对标幺化，再转为定点计算。

角度计算时，以 π 为基值对[$-\pi$，π]中的数标幺化，得到标幺值范围[-1,1]。例如，有

角度$\frac{\pi}{3}$，它的标幺值为$\frac{1}{3}$。

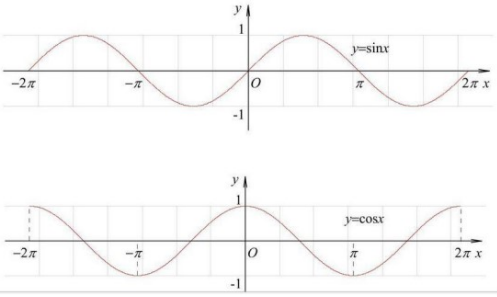
定点计算：

定点数有整数也有小数。一般来说，嵌入式处理器处理浮点数的速度是慢于整数计算的，因此可以将待计算数值标幺处理后，使用 Q1.15 格式（都是整数）的定点计算。



以 ST 电机库三角函数定点算法为例(查表法)：

```
#define SIN_COS_TABLE {\
    0x0000,0x00C9,0x0192,0x025B,0x0324,0x03ED,0x04B6,0x057F,\
    0x0648,0x0711,0x07D9,0x08A2,0x096A,0x0A33,0x0AFB,0x0BC4,\
    0x0C8C,0x0D54,0x0E1C,0x0EE3,0x0FAB,0x1072,0x113A,0x1201,\
    0x12C8,0x138F,0x1455,0x151C,0x15E2,0x16A8,0x176E,0x1833,\
    0x18F9,0x19BE,0x1A82,0x1B47,0x1C0B,0x1CCF,0x1D93,0x1E57,\
    0x1F1A,0x1FDD,0x209F,0x2161,0x2223,0x22E5,0x23A6,0x2467,\
    0x2528,0x25E8,0x26A8,0x2767,0x2826,0x28E5,0x29A3,0x2A61,\
    0x2B1F,0x2BDC,0x2C99,0x2D55,0x2E11,0x2ECC,0x2F87,0x3041,\
    0x30FB,0x31B5,0x326E,0x3326,0x33DF,0x3496,0x354D,0x3604,\
    0x36BA,0x376F,0x3824,0x38D9,0x398C,0x3A40,0x3AF2,0x3BA5,\
    0x3C56,0x3D07,0x3DB8,0x3E68,0x3F17,0x3FC5,0x4073,0x4121,\
    0x41CE,0x427A,0x4325,0x43D0,0x447A,0x4524,0x45CD,0x4675,\
    0x471C,0x47C3,0x4869,0x490F,0x49B4,0x4A58,0x4AFB,0x4B9D,\
    0x4C3F,0x4CE0,0x4D81,0x4E20,0x4EBF,0x4F5D,0x4FFB,0x5097,\
    0x5133,0x51CE,0x5268,0x5302,0x539B,0x5432,0x54C9,0x5560,\
    0x55F5,0x568A,0x571D,0x57B0,0x5842,0x58D3,0x5964,0x59F3,\
    0x5A82,0x5B0F,0x5B9C,0x5C28,0x5CB3,0x5D3E,0x5DC7,0x5E4F,\
    0x5ED7,0x5F5D,0x5FE3,0x6068,0x60EB,0x616E,0x61F0,0x6271,\
    0x62F1,0x6370,0x63EE,0x646C,0x64E8,0x6563,0x65DD,0x6656,\
    0x66CF,0x6746,0x67BC,0x6832,0x68A6,0x6919,0x698B,0x69FD,\
    0x6A6D,0x6ADC,0x6B4A,0x6BB7,0x6C23,0x6C8E,0x6CF8,0x6D61,\
    0x6DC9,0x6E30,0x6E96,0x6EFB,0x6F5E,0x6FC1,0x7022,0x7083,\
    0x70E2,0x7140,0x719D,0x71F9,0x7254,0x72AE,0x7307,0x735E,\
    0x73B5,0x740A,0x745F,0x74B2,0x7504,0x7555,0x75A5,0x75F3,\
    0x7641,0x768D,0x76D8,0x7722,0x776B,0x77B3,0x77FA,0x783F,\
    0x7884,0x78C7,0x7909,0x794A,0x7989,0x79C8,0x7A05,0x7A41,\
    0x7A7C,0x7AB6,0x7AEE,0x7B26,0x7B5C,0x7B91,0x7BC5,0x7BF8,\
    0x7C29,0x7C59,0x7C88,0x7CB6,0x7CE3,0x7D0E,0x7D39,0x7D62,\
    0x7D89,0x7DB0,0x7DD5,0x7DFA,0x7E1D,0x7E3E,0x7E5F,0x7E7E,\
    0x7E9C,0x7EB9,0x7ED5,0x7EEF,0x7F09,0x7F21,0x7F37,0x7F4D,\
    0x7F61,0x7F74,0x7F86,0x7F97,0x7FA6,0x7FB4,0x7FC1,0x7FCD,\
    0x7FD8,0x7FE1,0x7FE9,0x7FF0,0x7FF5,0x7FF9,0x7FFD,0x7FFE}
```



SIN_COS_TABLE 为 q1.15 格式数据。0x0000 到 0x7FFE，表示十进制 0 至 1；

SIN_COS_TABLE 共有 256 个数据，其表示将 $\frac{\pi}{2}$ 长度分割成 256 个单元。

```
typedef struct
{
  int16_t hCos;
  int16_t hSin;
} Trig_Components;
```

```
#define SIN_MASK        0x0300u
#define U0_90           0x0200u
#define U90_180         0x0300u
#define U180_270        0x0000u
#define U270_360        0x0100u
```

```
__weak Trig_Components MCM_Trig_Functions(int16_t hAngle)     输入参数hAngle为q1.15格式
{

  int32_t shindex;
  uint16_t uhindex;

  Trig_Components Local_Components;

  /* 10 bit index computation  */
  shindex = (((int32_t)32768) + ((int32_t)hAngle));  hAngle加32768；32768对应的是-Π
  uhindex = (uint16_t)shindex;   uhindex取低16位
  uhindex /= ((uint16_t)64);   uhindex右移6位

  switch (((uint16_t)uhindex) & SIN_MASK)
  {
    case U0_90:
    {
      Local_Components.hSin = hSin_Cos_Table[( uint8_t )( uhindex )];
      Local_Components.hCos = hSin_Cos_Table[( uint8_t )( 0xFFu - ( uint8_t )( uhindex ) )];
      break;
    }

    case U90_180:
    {
      Local_Components.hSin = hSin_Cos_Table[( uint8_t )( 0xFFu - ( uint8_t )( uhindex ) )];
      Local_Components.hCos = -hSin_Cos_Table[( uint8_t )( uhindex )];
      break;
    }

    case U180_270:
    {
      Local_Components.hSin = -hSin_Cos_Table[( uint8_t )( uhindex )];
      Local_Components.hCos = -hSin_Cos_Table[( uint8_t )( 0xFFu - ( uint8_t )( uhindex ) )];
      break;
    }

    case U270_360:
    {
      Local_Components.hSin =  -hSin_Cos_Table[( uint8_t )( 0xFFu - ( uint8_t )( uhindex ) )];
      Local_Components.hCos =  hSin_Cos_Table[( uint8_t )( uhindex )];
      break;
    }

    default:
      break;
  }
```

解释一下几个点：

（1）

```
shindex = (((int32_t)32768) + ((int32_t)hAngle));  hAngle加32768；32768对应的是-Π
```

[-Π，Π]对应[-32768,32767]，Q1.15 格式的角度 hAngle+32768，使得数据整体上移，此时[-Π，Π]对应[0,65535]。

（2）

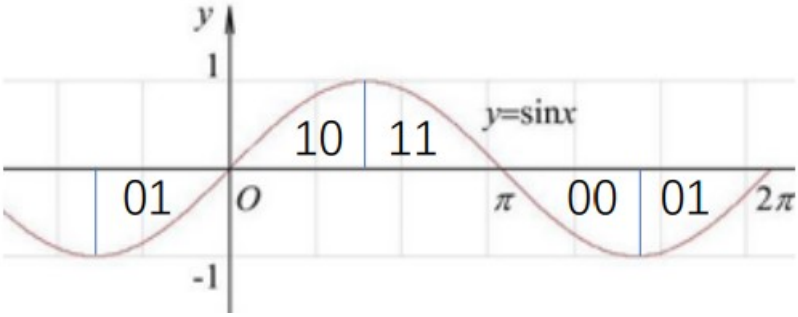
```
uhindex = (uint16_t)shindex;   uhindex取低16位
```

无符号 16 位数据取值范围[0,65535]，刚好和（1）中上移后得到的[-Π，Π]对应范围[0,65535]相吻合。

（3）

```
uhindex /= ((uint16_t)64);   uhindex右移6位
```

为什么要右移 6 位？先来看一张图。



由前面可知，定点角度在 x 轴[-Π,Π]定点为[-32768,32767]分了 2^{16} =65536 份（65536 数值可以理解成数据位置索引，下面同理），而 SIN_COS_TABLE 又将[0，Π/2]分了 2^8=256 份，即[-Π，Π]共分了 1024 份（含义同上）。

经过（1）和（2）的操作后，定点角度在 x 轴[0,2Π]定点为[0,65535]分了 2^{16} =65536 份（65536 数值可以理解成数据位置索引，下面同理），而 SIN_COS_TABLE 又将[0，Π/2]分了 2^8=256 份，即[0，2Π]共分了 1024 份（含义同上）。

从采样的角度来理解，TABLE 表在 2^{16} =65536 个数据中按一定间隔顺序采样了 2^{10} =1024 个数，采样点数据精度最大为 $2^6=64$ 。

因此 uhindex=uhindex>>6，将原来属于[0,65535]的 uhindex 按采样精度 $2^6=64$ 映射到[0,1024]，所以映射后的 uhindex 属于[$0,2^{10}$]， 对应 x 轴[0,2Π]。

又因为[0,Π/2]对应[$0,2^8$]，所以查看 10bit 的 uhindex 最高两位，就可以知道当前 uhindex 位于哪个编号区域。

实际计算例子：

1）

$$\text{Angle}=\frac{\pi}{3}\in[-\pi,\pi]\Rightarrow q1.15:\text{hAngle}=\frac{\pi}{3}/\pi*32768=10922$$

$$\text{shindex}=\text{hAngle}+32768=43680(o)=1010101010101010(\text{b})$$

$$\text{uhindex}=1010101010$$

Angle 属于 10 扇区，对应[0，Π/2]。

2）

$$\text{Angle}=\frac{5\pi}{4}\in[0,2\pi]\Rightarrow q1.15:\text{hAngle}=\frac{5\pi}{4}/\pi*32768=40958$$

$$\text{shindex}=\text{hAngle}+32768=73726(o)(overflow)=00010001111111111110(\text{b})$$

$$\text{uhindex}=0001111111111110>>6=0001111111$$

上面存在溢出的情况，低十六位后，右移 6 位后，可知 uhindex 属于 00 扇区。

SIN_MASK=0x0300u ， 是 判 断 的 扇 区 掩 码 ， 还 是 以 $\frac{\pi}{3}$ 举 例 ， 对 应

uhindex=1010101010。

```
switch (((uint16_t)uhindex) & SIN_MASK)
{
  case U0_90:
  {
    Local_Components.hSin = hSin_Cos_Table[( uint8_t )( uhindex )];
    Local_Components.hCos = hSin_Cos_Table[( uint8_t )( 0xFFu - ( uint8_t )( uhindex ) )];
    break;
  }
```

$uhindex \& 0x0300 = 0x0200$，对应 U0_90，此时

$\sin\frac{\pi}{3}$=SIN_COS_TABLE[(uint8_t)(uhindex)];cos 和 sin 在[0,Π/2]是互补的，所以有

$\mathrm{Cos}\frac{\pi}{3}$=SIN_COS_TABLE[(uint8_t)(0xFFu-(uint8_t)(uhindex))];

其余角度计算方式同理。

溢出检测：

定点计算会出现溢出现象，根据实际计算情况必须要做溢出处理。

在 park 与反 park 变换计算中，park 变换计算的时候已经做了溢出处理，所以在反 park 变换的计算时不用再做溢出处理。