窗函数的系数就是在这个函数中生成的。

```
void Lib_genWindow(void *win,
    uint32_t windowDatumType,
    uint32_t winLen,
    uint32_t winGenLen,
    int32_t oneQformat,
    uint32_t winType)
```

> win: 样本窗,这个函数执行完，窗系数就存在了win指针指向的地址里 windowDatumType：数据格式，可指定是int型还是short型 winLen：窗长度,，1D处理就是ADC采的数据长度 winGenLen：这个是窗函数系数的长度，虚实部的原因，取1/2 oneQformat：于虚位运算的，这里就是放大倍数，数值为：*#define ONE_Q15 (1 << 15) * winType：窗类型

```
uint32_t winIndx;//窗序号
    int32_t winVal;//溢出保护
    int16_t * win16 = (int16_t *)win;//demo中选了这个
    int32_t * win32 = (int32_t *)win;
      float phi = 2 * PI_ / ((float)winLen - 1);
      //phi是最小相位值，取了读音，其实就是那个希腊字母ϕ
      //窗函数一个周期是2π，每个点的相位就是2π/(winlen-1)
      //phi和phy应当都是指相位
```

N长度的Blackman窗的数学公式，时域表达式:

The following equation defines the Blackman window of length *N*:

$$w(n) = 0.42 - 0.5\cos\frac{2\pi n}{N-1} + 0.08\cos\frac{4\pi n}{N-1}, \quad 0 \le n \le M-1$$

where *M* is *N*/2 for *N* even and (*N* + 1)/2 for *N* odd.

> M:表示窗函数的有效长度,即winGenLen

代码中采用了复数的计算方式，然后取了实部输出。

$$eR = \cos\frac{2\pi n}{N-1} = real(e^{j\frac{2\pi n}{N-1}})$$

$$e2R = \cos\frac{4\pi n}{N-1} = real((e^{j\frac{2\pi n}{N-1}})^2)$$

```
for (winIndx = 0; winIndx < winGenLen; winIndx++)
```

某次运算n=k时，$$eR = real(e^{j\frac{2\pi k}{N-1}})$$

$$eR = real(e^{j\frac{2\pi(k+1)}{N-1}})$$

$$e^{j\frac{2\pi(k+1)}{N-1}} = e^{j\frac{2\pi k}{N-1}} \cdot e^{j\frac{2\pi}{N-1}}$$

而

$$e^{j\frac{2\pi}{N-1}} = e^{j\cdot phi}$$

下一次循环，n=k+1时，

phi就是代码中的phi，每次for循环都将上一次的运算结果存到tmpR中间变量中，然后再乘 （同理e2R乘 ），最后取实部输出。eR和e2R的计算是一个累复乘的过程。

这样计算，减少了正余弦运算的次数，整个函数只计算了一次正余弦，节省了资源提高了运算性能。

这个算法对我们有借鉴意义，相信各位都能有所收获。

```
//以布莱克曼窗Blackman为例讲解

switch(winType)
   #ifdef MMW_WIN_BLACKMAN_ENABLE
       case MMW_WIN_BLACKMAN:
       {
           float a0 = 0.42;
           float a1 = 0.5;
           float a2 = 0.08;
           for (winIndx = 0; winIndx < winGenLen; winIndx++)
           {
               winVal = (int32_t)((oneQformat * (a0 - a1*cos(phi * winIndx) +
                   a2*cos(2 * phi * winIndx))) + 0.5);
                   //要计算更新的有变量winIndx，其他都是固定值
```

```
                    //oneQformat是放大倍数，放大的原因是按照公式，计算出来的值都是小于1的，
整形变量无法表示
                    //最后面的这个0.5我推测是因为取整方式，想向上取整，应当不重要
                if (winVal >= oneQformat)//这个if语句起溢出保护的作用，保证窗系数的最大值
就是oneQformat - 1
                {
                    winVal = oneQformat - 1;
                }

                switch (windowDatumType) //输出窗系数，强制转换数据格式
                {
                    case FFT_WINDOW_INT16:
                        win16[winIndx] = (int16_t)winVal;
                        break;
                    case FFT_WINDOW_INT32:
                        win32[winIndx] = (int32_t)winVal;
                        break;
                    default:break;
                }
            }
            break;
        } #endif
    #ifdef MMW_WIN_HAMMING_ENABLE
        case MMW_WIN_HAMMING:
        {} #endif
    #ifdef MMW_WIN_RECT_ENABLE
        case MMW_WIN_RECT:
        {} #endif
    #ifdef MMW_WIN_HANNING_RECT_ENABLE
        case MMW_WIN_HANNING_RECT:
        {} #endif
    #ifdef MMW_WIN_HANNING_ENABLE
        case MMW_WIN_HANNING:
        {} #endif
```