```
#include <iom128.h>
#include <intrinsics.h>

/*****************************************************************
                    快速福利叶变换C函数
函数简介：此函数是通用的快速傅里叶变换C语言函数，移植性强，以下部分不依
        赖硬件。此函数采用联合体的形式表示一个复数，输入为自然顺序的复
        数（输入实数是可令复数虚部为0），输出为经过FFT变换的自然顺序的
        复数
使用说明：使用此函数只需更改宏定义FFT_N的值即可实现点数的改变，FFT_N的
        应该为2的N次方，不满足此条件时应在后面补0
函数调用：FFT(s);
时    间：2010-2-20
版    本：Ver1.0
参考文献：

*****************************************************************/
#include<math.h>

#define PI 3.14159265358979323846264338327950288419 71          //定义圆周率值
#define FFT_N 128                                   //定义福利叶变换的点数

struct compx {float real,imag;};                    //定义一个复数结构
struct                                  compx                                   s[FFT_N];
//FFT输入和输出：从S[1]开始存放，根据大小自己定义


/*****************************************************************
函数原型：struct compx EE(struct compx b1,struct compx b2)
函数功能：对两个复数进行乘法运算
输入参数：两个以联合体定义的复数a,b
输出参数：a和b的乘积，以联合体的形式输出
*****************************************************************/
struct compx EE(struct compx a,struct compx b)
{
 struct compx c;
 c.real=a.real*b.real-a.imag*b.imag;
 c.imag=a.real*b.imag+a.imag*b.real;
 return(c);
}

/*****************************************************************
函数原型：void FFT(struct compx *xin,int N)
函数功能：对输入的复数组进行快速傅里叶变换（FFT）
```

```
输入参数：*xin复数结构体组的首地址指针，struct型
************************************************************/
void FFT(struct compx *xin)
{
 int f,m,nv2,nm1,i,k,l,j=0;
 struct compx u,w,t;

 nv2=FFT_N/2;              //变址运算，即把自然顺序变成倒位序，采用雷德算法
 nm1=FFT_N-1;
 for(i=0;i<nm1;i++)
 {
  if(i<j)                 //如果i<j,即进行变址
   {
    t=xin[j];
    xin[j]=xin[i];
    xin[i]=t;
   }
  k=nv2;                  //求j的下一个倒位序
  while(k<=j)             //如果k<=j,表示j的最高位为1
   {
    j=j-k;                //把最高位变成0
    k=k/2;                //k/2，比较次高位，依次类推，逐个比较，直到某个位为0
   }
 j=j+k;                   //把0改为1
}

 {
 int le,lei,ip;                    //FFT运算核，使用蝶形运算完成FFT运算
  f=FFT_N;
 for(l=1;(f=f/2)!=1;l++)           //计算l的值，即计算蝶形级数
       ;
 for(m=1;m<=l;m++)                 // 控制蝶形结级数
 {                                 //m表示第m级蝶形，l为蝶形级总数l=log（2）N
  le=2<<(m-1);                     //le蝶形结距离，即第m级蝶形的蝶形结相距le点
  lei=le/2;                        //同一蝶形结中参加运算的两点的距离
  u.real=1.0;                      //u为蝶形结运算系数，初始值为1
  u.imag=0.0;
  w.real=cos(PI/lei);              //w为系数商，即当前系数与前一个系数的商
  w.imag=-sin(PI/lei);
  for(j=0;j<=lei-1;j++)            //控制计算不同种蝶形结，即计算系数不同的蝶形结
   {
    for(i=j;i<=FFT_N-1;i=i+le)     //控制同一蝶形结运算，即计算系数相同蝶形结
     {
      ip=i+lei;                    //i，ip分别表示参加蝶形运算的两个节点
```

```
      t=EE(xin[ip],u);                //蝶形运算，详见公式
      xin[ip].real=xin[i].real-t.real;
      xin[ip].imag=xin[i].imag-t.imag;
      xin[i].real=xin[i].real+t.real;
      xin[i].imag=xin[i].imag+t.imag;
     }
    u=EE(u,w);                       //改变系数，进行下一个蝶形运算
   }
  }
 }

}

/************************************************************
函数原型：void main()
函数功能：测试FFT变换，演示函数使用方法
输入参数：无
输出参数：无
************************************************************/
void main()
{
 int i;
 for(i=0;i<FFT_N;i++)                //给结构体赋值
 {
   s[i].real=sin(2*3.141592653589793*i/FFT_N); //实部为正弦波FFT_N点采样，赋值为1
   s[i].imag=0;                      //虚部为0
 }

 FFT(s);                             //进行快速福利叶变换

 for(i=0;i<FFT_N;i++)                //求变换后结果的模值，存入复数的实部部分
 s[i].real=sqrt(s[i].real*s[i].real+s[i].imag*s[i].imag);

  while(1);
}
```

```
#include <iom128.h>
#include <intrinsics.h>

/*****************************************************************
                快速福利叶变换C程序包
函数简介：此程序包是通用的快速傅里叶变换C语言函数，移植性强，以下部分不依
      赖硬件。此程序包采用联合体的形式表示一个复数，输入为自然顺序的复
      数（输入实数是可令复数虚部为0），输出为经过FFT变换的自然顺序的
      复数.此程序包可在初始化时调用create_sin_tab()函数创建正弦函数表，
      以后的可采用查表法计算耗时较多的sin和cos运算，加快可计算速度
使用说明：使用此函数只需更改宏定义FFT_N的值即可实现点数的改变，FFT_N的
      应该为2的N次方，不满足此条件时应在后面补0。若使用查表法计算sin值和
      cos值，应在调用FFT函数前调用create_sin_tab()函数创建正弦表
函数调用：FFT(s);
时   间：2010-2-20
版   本：Ver1.1
参考文献：
*****************************************************************/
#include<math.h>

#define FFT_N 128                                   //定义福利叶变换的点数
#define PI 3.14159265358979323846264338327950288419971          //定义圆周率值

struct compx {float real,imag;};    //定义一个复数结构
struct compx s[FFT_N];          //FFT输入和输出：从S[0]开始存放，根据大小自己定义
float SIN_TAB[FFT_N/2];         //定义正弦表的存放空间

/*****************************************************************
函数原型：struct compx EE(struct compx b1,struct compx b2)
函数功能：对两个复数进行乘法运算
输入参数：两个以联合体定义的复数a,b
输出参数：a和b的乘积，以联合体的形式输出
*****************************************************************/
struct compx EE(struct compx a,struct compx b)
{
 struct compx c;
 c.real=a.real*b.real-a.imag*b.imag;
 c.imag=a.real*b.imag+a.imag*b.real;
 return(c);
}

/*****************************************************************
函数原型：void create_sin_tab(float *sin_t)
函数功能：创建一个正弦采样表，采样点数与福利叶变换点数相同
```

```
输入参数：*sin_t存放正弦表的数组指针
输出参数：无
*****************************************************************/
void create_sin_tab(float *sin_t)
{
 int i;
 for(i=0;i<FFT_N/2;i++)
 sin_t[i]=sin(2*PI*i/FFT_N);
}
/*****************************************************************
函数原型：void sin_tab(float pi)
函数功能：采用查表的方法计算一个数的正弦值
输入参数：pi 所要计算正弦值弧度值，范围0--2*PI，不满足时需要转换
输出参数：输入值pi的正弦值
*****************************************************************/
float sin_tab(float pi)
{
 int n;
 float a;
  n=(int)(pi*FFT_N/2/PI);

 if(n>=0&&n<FFT_N/2)
  a=SIN_TAB[n];
 else if(n>=FFT_N/2&&n<FFT_N)
   a=-SIN_TAB[n-FFT_N/2];
 return a;
}
/*****************************************************************
函数原型：void cos_tab(float pi)
函数功能：采用查表的方法计算一个数的余弦值
输入参数：pi 所要计算余弦值弧度值，范围0--2*PI，不满足时需要转换
输出参数：输入值pi的余弦值
*****************************************************************/
float cos_tab(float pi)
{
  float a,pi2;
  pi2=pi+PI/2;
  if(pi2>2*PI)
    pi2-=2*PI;
  a=sin_tab(pi2);
  return a;
}
/*****************************************************************
函数原型：void FFT(struct compx *xin,int N)
```

```
函数功能：对输入的复数组进行快速傅里叶变换（FFT）
输入参数：*xin复数结构体组的首地址指针，struct型
输出参数：无
*****************************************************************/
void FFT(struct compx *xin)
{
 int f,m,nv2,nm1,i,k,l,j=0;
 struct compx u,w,t;

 nv2=FFT_N/2;          //变址运算，即把自然顺序变成倒位序，采用雷德算法
 nm1=FFT_N-1;
 for(i=0;i<nm1;i++)
 {
 if(i<j)               //如果i<j,即进行变址
  {
   t=xin[j];
   xin[j]=xin[i];
   xin[i]=t;
  }
 k=nv2;                //求j的下一个倒位序
 while(k<=j)           //如果k<=j,表示j的最高位为1
  {
   j=j-k;              //把最高位变成0
   k=k/2;              //k/2，比较次高位，依次类推，逐个比较，直到某个位为0
  }
 j=j+k;                //把0改为1
}

 {
 int le,lei,ip;                  //FFT运算核，使用蝶形运算完成FFT运算
  f=FFT_N;
 for(l=1;(f=f/2)!=1;l++)         //计算l的值，即计算蝶形级数
       ;
for(m=1;m<=l;m++)                // 控制蝶形结级数
 {                        //m表示第m级蝶形，l为蝶形级总数l=log（2）N
  le=2<<(m-1);              //le蝶形结距离，即第m级蝶形的蝶形结相距le点
  lei=le/2;                 //同一蝶形结中参加运算的两点的距离
  u.real=1.0;               //u为蝶形结运算系数，初始值为1
  u.imag=0.0;
  //w.real=cos(PI/lei);             //不适用查表法计算sin值和cos值
  // w.imag=-sin(PI/lei);
  w.real=cos_tab(PI/lei);         //w为系数商，即当前系数与前一个系数的商
  w.imag=-sin_tab(PI/lei);
  for(j=0;j<=lei-1;j++)           //控制计算不同种蝶形结，即计算系数不同的蝶形结
```

```
    {
     for(i=j;i<=FFT_N-1;i=i+le)      //控制同一蝶形结运算，即计算系数相同蝶形结
      {
       ip=i+lei;                    //i，ip分别表示参加蝶形运算的两个节点
       t=EE(xin[ip],u);             //蝶形运算，详见公式
       xin[ip].real=xin[i].real-t.real;
       xin[ip].imag=xin[i].imag-t.imag;
       xin[i].real=xin[i].real+t.real;
       xin[i].imag=xin[i].imag+t.imag;
      }
     u=EE(u,w);                     //改变系数，进行下一个蝶形运算
    }
  }
 }

}

/************************************************************
函数原型：void main()
函数功能：测试FFT变换，演示函数使用方法
输入参数：无
输出参数：无
************************************************************/
void main()
{
 int i;
 create_sin_tab(SIN_TAB);
 for(i=0;i<FFT_N;i++)                //给结构体赋值
 {
   s[i].real=sin(2*3.141592653589793*i/FFT_N); //实部为正弦波FFT_N点采样，赋值为1
   s[i].imag=0;                     //虚部为0
 }

 FFT(s);                            //进行快速福利叶变换

 for(i=0;i<FFT_N;i++)               //求变换后结果的模值，存入复数的实部部分
 s[i].real=sqrt(s[i].real*s[i].real+s[i].imag*s[i].imag);

  while(1);
}
```

```
#include <iom128.h>
#include <intrinsics.h>

/*****************************************************************
                快速福利叶变换C程序包
函数简介：此程序包是通用的快速傅里叶变换C语言函数，移植性强，以下部分不依
      赖硬件。此程序包采用联合体的形式表示一个复数，输入为自然顺序的复
      数（输入实数是可令复数虚部为0），输出为经过FFT变换的自然顺序的
      复数.此程序包可在初始化时调用create_sin_tab()函数创建正弦函数表，
      以后的可采用查表法计算耗时较多的sin和cos运算，加快可计算速度.与
      Ver1.1版相比较，Ver1.2版在创建正弦表时只建立了1/4个正弦波的采样值，
      相比之下节省了FFT_N/4个存储空间
使用说明：使用此函数只需更改宏定义FFT_N的值即可实现点数的改变，FFT_N的
      应该为2的N次方，不满足此条件时应在后面补0。若使用查表法计算sin值和
      cos值，应在调用FFT函数前调用create_sin_tab()函数创建正弦表
函数调用：FFT(s);
时   间：2010-2-20
版   本：Ver1.2
参考文献：
*****************************************************************/
#include<math.h>

#define FFT_N 128                                   //定义福利叶变换的点数
#define PI 3.14159265358979323846264338327950288419971      //定义圆周率值

struct compx {float real,imag;};                            //定义一个复数结构
struct compx s[FFT_N];          //FFT输入和输出：从S[0]开始存放，根据大小自己定义
float SIN_TAB[FFT_N/4+1];       //定义正弦表的存放空间

/*****************************************************************
函数原型：struct compx EE(struct compx b1,struct compx b2)
函数功能：对两个复数进行乘法运算
输入参数：两个以联合体定义的复数a,b
输出参数：a和b的乘积，以联合体的形式输出
*****************************************************************/
struct compx EE(struct compx a,struct compx b)
{
 struct compx c;
 c.real=a.real*b.real-a.imag*b.imag;
 c.imag=a.real*b.imag+a.imag*b.real;
 return(c);
}

/*****************************************************************
```

```
函数原型：void create_sin_tab(float *sin_t)
函数功能：创建一个正弦采样表，采样点数与福利叶变换点数相同
输入参数：*sin_t存放正弦表的数组指针
输出参数：无
*****************************************************************/
void create_sin_tab(float *sin_t)
{
 int i;
 for(i=0;i<=FFT_N/4;i++)
 sin_t[i]=sin(2*PI*i/FFT_N);
}
/*****************************************************************
函数原型：void sin_tab(float pi)
函数功能：采用查表的方法计算一个数的正弦值
输入参数：pi 所要计算正弦值弧度值，范围0--2*PI，不满足时需要转换
输出参数：输入值pi的正弦值
*****************************************************************/
float sin_tab(float pi)
{
 int n;
 float a;
  n=(int)(pi*FFT_N/2/PI);

 if(n>=0&&n<=FFT_N/4)
   a=SIN_TAB[n];
 else if(n>FFT_N/4&&n<FFT_N/2)
   {
    n-=FFT_N/4;
    a=SIN_TAB[FFT_N/4-n];
   }
 else if(n>=FFT_N/2&&n<3*FFT_N/4)
   {
    n-=FFT_N/2;
    a=-SIN_TAB[n];
  }
 else if(n>=3*FFT_N/4&&n<3*FFT_N)
   {
    n=FFT_N-n;
    a=-SIN_TAB[n];
  }

 return a;
}
/*****************************************************************
```

```
函数原型：void cos_tab(float pi)
函数功能：采用查表的方法计算一个数的余弦值
输入参数：pi 所要计算余弦值弧度值，范围0--2*PI，不满足时需要转换
输出参数：输入值pi的余弦值
*****************************************************************/
float cos_tab(float pi)
{
  float a,pi2;
  pi2=pi+PI/2;
  if(pi2>2*PI)
    pi2-=2*PI;
  a=sin_tab(pi2);
  return a;
}
/*****************************************************************
函数原型：void FFT(struct compx *xin,int N)
函数功能：对输入的复数组进行快速傅里叶变换（FFT）
输入参数：*xin复数结构体组的首地址指针，struct型
输出参数：无
*****************************************************************/
void FFT(struct compx *xin)
{
 int f,m,nv2,nm1,i,k,l,j=0;
 struct compx u,w,t;

  nv2=FFT_N/2;              //变址运算，即把自然顺序变成倒位序，采用雷德算法
  nm1=FFT_N-1;
  for(i=0;i<nm1;i++)
  {
   if(i<j)                 //如果i<j,即进行变址
    {
     t=xin[j];
     xin[j]=xin[i];
     xin[i]=t;
    }
   k=nv2;                  //求j的下一个倒位序
   while(k<=j)             //如果k<=j,表示j的最高位为1
    {
     j=j-k;                //把最高位变成0
     k=k/2;                //k/2，比较次高位，依次类推，逐个比较，直到某个位为0
    }
  j=j+k;                   //把0改为1
 }
```

```
 {
  int le,lei,ip;                    //FFT运算核，使用蝶形运算完成FFT运算
   f=FFT_N;
  for(l=1;(f=f/2)!=1;l++)             //计算l的值，即计算蝶形级数
        ;
 for(m=1;m<=l;m++)                    // 控制蝶形结级数
  {                                 //m表示第m级蝶形，l为蝶形级总数l=log（2）N
   le=2<<(m-1);                     //le蝶形结距离，即第m级蝶形的蝶形结相距le点
   lei=le/2;                        //同一蝶形结中参加运算的两点的距离
   u.real=1.0;                      //u为蝶形结运算系数，初始值为1
   u.imag=0.0;
   //w.real=cos(PI/lei);             //不适用查表法计算sin值和cos值
   // w.imag=-sin(PI/lei);
   w.real=cos_tab(PI/lei);           //w为系数商，即当前系数与前一个系数的商
   w.imag=-sin_tab(PI/lei);
   for(j=0;j<=lei-1;j++)             //控制计算不同种蝶形结，即计算系数不同的蝶形结
    {
     for(i=j;i<=FFT_N-1;i=i+le)      //控制同一蝶形结运算，即计算系数相同蝶形结
      {
       ip=i+lei;                    //i，ip分别表示参加蝶形运算的两个节点
       t=EE(xin[ip],u);             //蝶形运算，详见公式
       xin[ip].real=xin[i].real-t.real;
       xin[ip].imag=xin[i].imag-t.imag;
       xin[i].real=xin[i].real+t.real;
       xin[i].imag=xin[i].imag+t.imag;
      }
     u=EE(u,w);                     //改变系数，进行下一个蝶形运算
    }
  }
 }

}

/************************************************************
函数原型：void main()
函数功能：测试FFT变换，演示函数使用方法
输入参数：无
输出参数：无
************************************************************/
void main()
{
 int i;
 create_sin_tab(SIN_TAB);
 for(i=0;i<FFT_N;i++)                //给结构体赋值
```

```
  {
    s[i].real=sin(2*3.141592653589793*i/FFT_N); //实部为正弦波FFT_N点采样，赋值为1
    s[i].imag=0;                              //虚部为0
  }

  FFT(s);                                    //进行快速福利叶变换

  for(i=0;i<FFT_N;i++)                       //求变换后结果的模值，存入复数的实部部分
  s[i].real=sqrt(s[i].real*s[i].real+s[i].imag*s[i].imag);

   while(1);
}
```