**实验三 快速傅里叶算法（FFT）实验----实验箱C语言程序注释**

**#include** "myapp.h"

**#include** "ICETEK-VC5509-EDU.h"

**#include** "scancode.h"

**#include** <math.h>

**#define PI 3.1415926**

**#define** SAMPLENUMBER 128

**void InitForFFT();**/\*蝶形算子乘数(也称旋转因子)exp(-j\*2\*PI\*i/N)初始化;方便调用\*/

**void MakeWave();**/\* 该函数能产生各种各样的谐波\*/

**int** INPUT**[SAMPLENUMBER],**DATA**[SAMPLENUMBER];**

**float** fWaveR**[SAMPLENUMBER],**fWaveI**[SAMPLENUMBER],**w**[SAMPLENUMBER];**

**float** cos\_tab**[SAMPLENUMBER],**sin\_tab**[SAMPLENUMBER];**

**main()**



**{**

**int i;**

**InitForFFT();**

/\*蝶形算子中的乘数(也称旋转因子)exp(-j\*2\*PI\*i/N)事先初始化好，方便调用\*/

**MakeWave();** /\* 该函数能产生各种各样的谐波\*/

**for ( i=0;i<SAMPLENUMBER;i++ )**

**{**

**fWaveR[i]=INPUT[i];**  /\* 因输入时间序列信号INPUT为实数序列信号，INPUT也是输入信号的实数部分fWaveR，序列fWaveR作为FFT()函数的实参，对应FFT()函数的第1个形参dataR：也对应信号流图的输入输出的实数部分dataR \*/

**fWaveI[i]=0.0f;**  /\* 因输入时间序列信号INPUT为实数序列信号，所以输入信号的虚数部分fWaveI=0，序列fWaveI作为FFT()函数的实参，对应FFT()函数的第2个形参dataI：也对应信号流图蝶形算子的输入输出的虚数部分dataI \*/

**w[i]=0.0f;**  /\*初始化输入信号波形的频谱值为0\*/

**}**

**FFT(fWaveR,fWaveI);** /\* 对输入波形进行FFT变换,FFT算法采用同址运算，实参fWaveR,fWaveI既是信号流图第一级的输入信号，也作为信号流图每级蝶形算子的输出的存储数组，也是每级蝶形算子的输入数组，即可递推下去 \*/

**for ( i=0;i<SAMPLENUMBER;i++ )**

**{**

**DATA[i]=w[i];** /\* 对输入信号波形进行FFT变换后频域输出序列的幅值w[i]即是输入波形的频谱，将其存储到输出数组DATA接口\*/

**}**

**while ( 1 );** // break point

**}**

/\*下面 FFT()函数算法逐级计算FFT，一级一级递推，可进行同址运算，dataR[]，dataI[]既是 FFT()函数的输入参数（信号流图第一级的输入信号序列的实数部分dataR[]，此时实部dataR[]是实数，虚部dataI[]=0），

\* dataR[]，dataI[]又作为信号流图中每级 FFT蝶形算子的输出的存储数组单元，每级的输出是复数，此时虚部就可能不是0，

\* 因输入信号的虚部dataI[]=0，在计算每级蝶形算子的输出前，dataI[]又用来暂存将输入信号序列进行序号码位倒置后的重新排列的数据，将序号码位倒置后的重排数据传给dataR[]后，再恢复dataI[]=0\*/

**void FFT(float dataR[SAMPLENUMBER],float dataI[SAMPLENUMBER])**

**{**  /\* 原程序是128点的FFT算法(可改成256点或其他值)，采用时间抽取DIT的FFT算法， 对应算法信号流图的输入信号时间序列序号按码位倒置顺序排列，输出频域序列顺序排列。\*/

**int x0,x1,x2,x3,x4,x5,x6,xx;** /\*二进制序号值(x6x5x4x3x2x1x0)的各个位\*/

**int i,j,k,b,p,L;**

**float TR,TI,temp;**

/\*\*\*\*\*\*\*\*\*\* following code invert sequence \*\*\*\*\*下面进行时间序列序号的码位倒置，二进制序号值(x6x5x4x3x2x1x0)的各个位独立出来再反向\*\*\*\*\*\*\*/

**for ( i=0;i<SAMPLENUMBER;i++ )**

**{**

**x0=x1=x2=x3=x4=x5=x6=0;** //128点FFT序号值用7位二进制(x6x5x4x3x2x1x0)表示，

**x0=i&0x01; x1=(i/2)&0x01; x2=(i/4)&0x01; x3=(i/8)&0x01;x4=(i/16)&0x01; x5=(i/32)&0x01; x6=(i/64)&0x01;**  /\* /号代表除，可用右移实现\*/

//提取128点输入信号序列的原顺序排列的序号(即算法信号流图上从上向下按顺序排列的位置序号)i对应的7位二进制数(b6b5b4b3b2b1b0)每位的值(x6x5x4x3x2x1x0)

**xx=x0\*64+x1\*32+x2\*16+x3\*8+x4\*4+x5\*2+x6;** //实现序号的码位倒置，就是将最低位到最高位等依次反转倒置过来

/\*将输入信号序列的原顺序排列的序号(算法信号流图上从上向下按顺序排列的位置序号)i(对应7位二进制数的序号值(7x6x5x4x3x2x1x0))码位倒置, 得到算法信号流图上从上向下按顺序排列的序号位置上所对应的的输入信号时间序列的码位倒置序号xx(x0x1x2x3x4x5x6)\*/

**dataI[xx]=dataR[i];**  //将输入实数信号序列作为输入信号的实数部分序列，将序号按码位倒置顺序存放的数据，用dataI[]数组暂存，得到序号码位倒置后的数据dataI[xx]

**}**

**for ( i=0;i<SAMPLENUMBER;i++ )**

**{**

**dataR[i]=dataI[i]; dataI[i]=0;** //将dataR[i]变成按码位倒置顺序存放的数据，dataI[i]数据清0

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* following code FFT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*N(=2^m)点的FFT运算可以分成m级，每一级都有N／2个碟形。DIT FFT的基本思想是用3层循环完成全部运算(N点FFT)。

第一层循环：由于N=2^m需要m级计算，第一层循环对运算的级数进行控制。\*\*\*\*\*\*\*\*\*\*/

**for ( L=1;L<=7;L++ )**

/\* for(1) 循环是三层嵌套循环的最外层循环，用序号1表示，for(1) 循环的任务： 128点DFT需要7级,控制计算是在哪一级(L)进行\*/

**{**  /\* 该循环先进行计算 ：b= 2^(L-1)，再嵌套for (2)循环\*/

**b=1; i=L-1;** /\* b表示当前层不同乘数(也称旋转因子)exp(-j\*2\*PI\*i/N)的个数(乘数在教材中简记作(WN)^n), 即第L级一个2^L点DFT运算群组内蝶形算子个数 \*/

**while ( i>0 )**

**{**

**b=b\*2; i--;**  /\* 第L级需乘的旋转因子(WN)^n的个数是 b= 2^(L-1) \*/

**}**

**for ( j=0;j<=b-1;j++ )** /\*for (2)循环是三层嵌套循环的第二层,该循环先进行计算: p= 2^(7-L)\*j (p最大2^6:每级蝶形算子个数),再嵌套for (3)循环 \*/

**{**

/\* 第二层循环根据乘数个数进行控制，保证对于每一个蝶形因子在第三层循环要执行一次，这样，第三层循环在第二层循环控制下，每一级要进行b= 2^(L-1) 次循环计算。\*/

**p=1; i=7-L;**

**while ( i>0 )** /\*while循环进行计算:p= 2^(7-L),或写成 p=pow(2,7-L) \*/

**{**

**p=p\*2; i--;**/\*计算第L级的2^L点DFT运算群组个数:p= 2^(7-L) \*/

**}**

**p=p\*j;**  /\*p\*j后，得结果：p= 2^(7-L)\*j，或写成 p=pow(2,7-L)\*j; \*/

/\*第L级每个群组内 2^(L-1)个碟形算子的旋转因子(WN)^p中的P,可表示为p=j\*2^(m-L),其中j=0,1,2,…,(2^(L-1)-1)。(N=2^m)\*/



/\*每个群组内每个碟形算子对应一个p

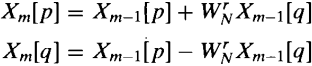
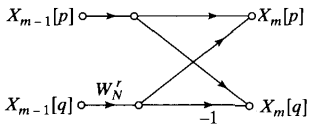
**for (k=j;k<SAMPLENUMBER;k=k+2\*b)** /\*for(3)循环是三层嵌套循环的最内层循环\*/

/\*第三层循环的次数为：第L级的群组个数,N=2^7时,共有N／2^L= 2^(7-L)个群组(每个群组内有2^(L-1)个蝶形算子，形成2^L点的DFT运算，最后一级只有一个N点的蝶形算子群组)，并且同一级内不同群组的乘数分布相同。(可推广至N=2^m)

\*当第二层循环确定某一乘数后，第三层循环要将本级中每个群组中具有同一乘数的蝶形算子计算一次，即第三层循环每执行完一次循环要进行N／2^L= 2^(7-L)个群组的碟形计算。\*/

/\*同一乘数个数对应着相邻间隔为2^L个点的N／2^L= 2^(7-L)个DFT算子群组个数（最后一级只有一个群组，没有相同的乘数，每个乘数都不同）。\*/

/\*第2层循环次数为每个群组内乘数个数：b=2^(L-1)=每个群组内蝶形算子个数。第3层循环计算每个蝶形算子的2点输出,这样2,3层循环共计算2\*2^(L-1)\*2^(7-L)=2^7点输出\*/



{ /\*下面式子计算FFT信号流图中每级的每个蝶形算子的2点输出，每个蝶形算子有2点输出，分别包含实部（dataR[k],dataR[k+b]）和虚部（dataI[k],dataI[k+b]），每级蝶形算子2点输出间的间距b各不相同\*/

**TR=dataR[k]; TI=dataI[k]; temp=dataR[k+b];**//因蝶形算子采用同址运算,dataR,dataI既存输入,也存输出,下面3,4行式中用到输入dataR,dataI,但在1,2,3行中dataR,dataI已存为输出,因此需暂存输入值

**dataR[k]=dataR[k]+dataR[k+b]\*cos\_tab[p]+dataI[k+b]\*sin\_tab[p];**/\* 第1,2行计算蝶形算子上部的复数输出: 复数data[k]+复数data[k+b]\*exp(-j\*2\*PI\*p/N)，第1行计算蝶形算子上部输出的实部\*/

**dataI[k]=dataI[k]-dataR[k+b]\*sin\_tab[p]+dataI[k+b]\*cos\_tab[p];**/\* 第2行计算蝶形算子下部输出的虚部\*/

**dataR[k+b]=TR-dataR[k+b]\*cos\_tab[p]-dataI[k+b]\*sin\_tab[p];**/\* 第3,4行计算蝶形算子下部的复数输出: 复数data[k]-复数data[k+b]\*exp(-j\*2\*PI\*p/N)，第3行计算蝶形算子下部输出的实部\*/

**dataI[k+b]=TI+temp\*sin\_tab[p]-dataI[k+b]\*cos\_tab[p];**  /\* 第4行计算蝶形算子下部输出的虚部\*/



**}** /\* END for (3) \*/

**}** /\* END for (2) \*/

**}** /\* END for (1) \*/

**for ( i=0;i<SAMPLENUMBER/2;i++ )**

**{**

**w[i]=sqrt(dataR[i]\*dataR[i]+dataI[i]\*dataI[i]);**  /\*计算DFT变换的幅值,即信号的频谱，利用信号频谱的对称性，只需计算前半部分即可\*/

**}**

**}** /\* END FFT \*/

**void InitForFFT()**  /\*蝶形算子中的乘数（也称旋转因子）exp(-j\*2\*PI\*i/N)事先初始化好，方便调用\*/

**{**

**int i;**

**for ( i=0;i<SAMPLENUMBER;i++ )**

**{**

**sin\_tab[i]=sin(PI\*2\*i/SAMPLENUMBER);** /\* 产生蝶形算子中的乘数exp(-j\*2\*PI\*i/N)的 虚部\*/

**cos\_tab[i]=cos(PI\*2\*i/SAMPLENUMBER);** /\* 产生蝶形算子中的乘数exp(-j\*2\*PI\*i/N)的 实部\*/

**}**

**}**

**void MakeWave()** /\* 该函数能产生各种各样的谐波\*/

**{**

**int i;**

**for ( i=0;i<SAMPLENUMBER;i++ )**

**{**  /\* 1024是产生的正弦信号相应的幅值，只要弄出个基波，以之为标准即可产生相应的谐波，并且谐波赋值相应减小\*/

/\* INPUT[i]=sin(PI\*2\*i/SAMPLENUMBER)\*1024

+sin(PI\*2\*i/SAMPLENUMBER\*3)\*1024/3

+sin(PI\*2\*i/SAMPLENUMBER\*5)\*1024/5

+sin(PI\*2\*i/SAMPLENUMBER\*7)\*1024/7

+sin(PI\*2\*i/SAMPLENUMBER\*9)\*1024/9; \*/ /\* 产生输入时间序列信号，为实数序列信号 \*/

**INPUT[i]=sin(PI\*2\*i/SAMPLENUMBER\*3)\*1024;**

**}**

**}**