**同济大学电子与信息工程学院实验中心实验报告**



**实验课程名称：\_\_ \_ \_\_\_数字信号处理 \_\_任课教师 周俊鹤**

**实验项目名称：\_FFT、FIR、IIR的软硬件仿真 实验教师 任浩琪**

**姓名:\_\_\_\_ 学号: 专业：\_ 微电子科学与工程\_\_**

**同组同学：\_ \_ 实验日期:\_ 实验地点：\_**

**实验名称： FIR数字滤波器的设计**

**姓名: 学号: 同组： 无 实验日期:**

**一、实验目的**

(1) 了解和熟悉 ICETECK-DSP 教学实验系统平台；

(2) 了解和熟悉 DSP 开发集成环境 Code Composer Studio 3.3 软件的设置和使用；

(3) 掌握用窗函数法设计 FIR 数字滤波器的原理和方法；

(4) 了解各种窗函数对滤波器特性的影响。

**二、实验内容**

使用配置好的PC机以及ICETECK-DSP实验箱，熟悉有限冲激响应数字滤波器的基础理论；了解各种窗函数对滤波器特性的影响；了解数字滤波器系数的确定方法；根据要求设计低通FIR数字滤波器。

**滤波器的设计指标为：**通带边缘频率10kHz，阻带边缘频率22kHz，阻带衰减75dB，采样频率50kHz。

**具体设计：**

过渡带宽度=阻带边缘频率-通带边缘频率=22-10=12kHz

采样频率：

f1=通带边缘频率+(过渡带宽度)/2=10000+12000/2=16kHz

Ω1=2πf1/fs=0.64π

理想低通滤波器脉冲响应：

h1[n]=sin(nΩ1)/n/π=sin(0.64πn)/n/π

根据要求，选择布莱克曼窗，窗函数长度为：

N=5.98fs/过渡带宽度=5.98\*50/12=24.9

选择 N=25，窗函数为：

w[n]=0.42+0.5cos(2πn/24)+0.8cos(4πn/24)

滤波器脉冲响应为：

h[n]=h1[n]w[n] |n|≤12

h[n]=0 |n|＞12

根据上面计算，各式计算出 h[n]，然后将脉冲响应值移位为因果序列。

完成的滤波器的差分方程为：

y[n] =-0.001x[n-2]-0.002x[n-3]-0.002x[n-4]+0.01x[n-5]

-0.009x[n-6]-0.018x[n-7]-0.049x[n-8]-0.02x[n-9]

+0.11x[n-10]+0.28x[n-11]+0.64x[n-12]

+0.28x[n-13]-0.11x[n-14]-0.02x[n-15]

+0.049x[n-16]-0.018x[n-17]-0.009x[n-18]+0.01x[n-19]

-0.002x[n-20]-0.002x[n-21]+0.001x[n-22]

**三、实验方案及流程图**

**1.软件仿真Simulator**

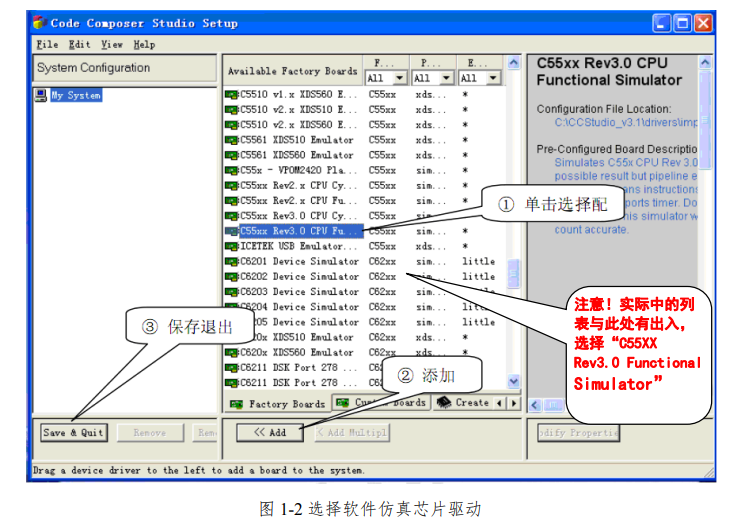
**（1）设置CCS**

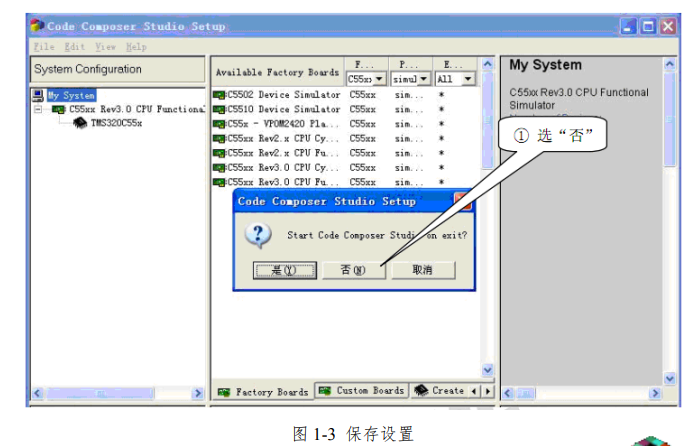


(a) 双击桌面上“Setup CCStudio v3.3”图标 进入 CCS 设置窗口；

(b) 在出现的如图 1-1 的窗口中，按标号顺序进行相应设置；



(c) 在出现的如图 1-2 的窗口中，按标号顺序进行相应设置；

(d) 在图 1-3 出现的窗口中，选择“否（N）”

此时CCS已经被设置成 Simulator 方式(软件仿真TMS320C55509器件的方式)，如果以后一直使用这一方式就不需要重新进行以上设置操作了，**在后续两个是实验中的simulator环境配置与此处相同，因此在后续两个实验中不再重复叙述。**

**（2）启动CCS；**

设置 CCS 为软件仿真驱动后，双击桌面上“CCStudio v3.3”图标：即可启动 Simulator（软件仿真）方式。

**(3) 打开工程文件；**

浏览程序：工程文件目录为 C:\ CCStudio\_v3.3\ICETEK\ VC5009Ae\ VC5009Ae\ Lab0501-FIR\Fir.pjt

打开方法：选择 Project->Open，选择 Lab0501-FIR 文件夹下的 Fir.pjt。

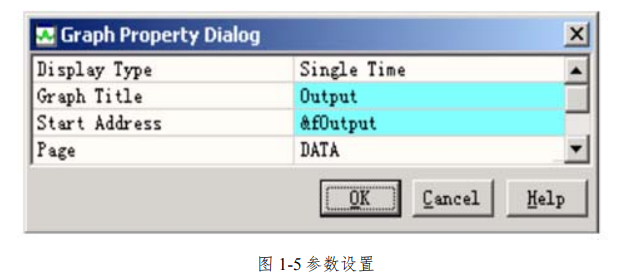
**(4) 编译并下载程序；**

选择 Project->Rebuild All 编译，然后下载程序：选 File->Load Program，在Debug 文件夹下选择 Fir.out，并打开。

**(5) 打开观察窗口；**

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 1-4 所示：

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 1-5 所示：

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 1-6 所示：

**(6) 清除显示；**

在以上打开的窗口中单击鼠标右键，选择弹出式菜单中“Clear Display”。

**(7) 设置断点；**

在工程管理窗中，双击 Fir.c，在有注释“break point”的语句处设置软件断点，方法：双击该语句左侧灰色控制条（会出现红色圆点）。

**(8) 运行并观察结果；**

(a) 选择“Debug”菜单的“Animate”项，或按 Alt+F5键运行程序。

(b) 观察“Input”、“Output”窗口中时域图形，并保存；观察滤波效果。

(c) 鼠标右键单击“Input”和“Output”窗口，选择“Properties…”项，设置“Display Type”为“FFT Magitude”，再单击“OK”按钮结束设置。

(d) 观察“Input”、“Output”窗口中频域图形，并保存；理解滤波效果。

**(9) 退出 CCS。**

**2.硬件仿真Emulator**

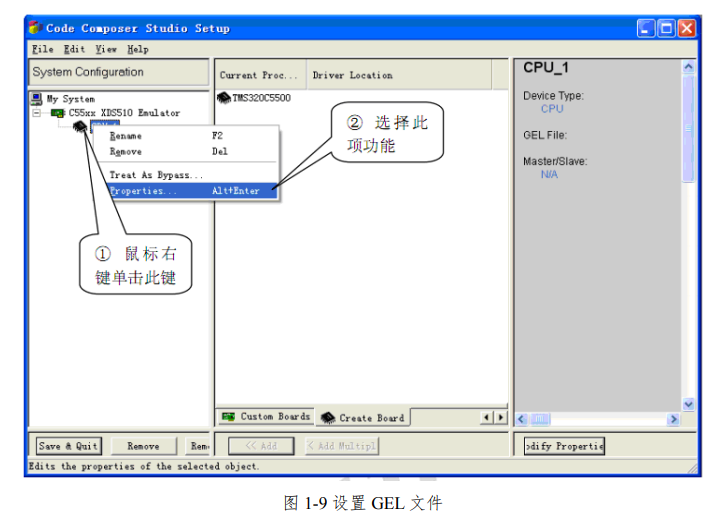
**（1）设置CCS**



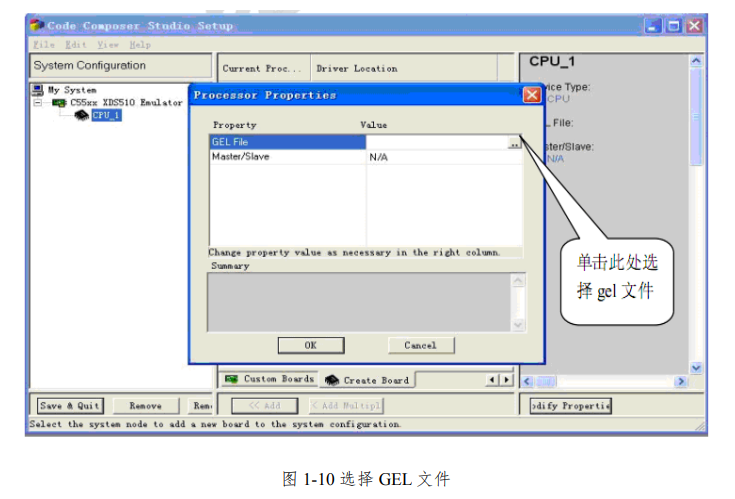
(a)双击桌面上“Setup CCStudio v3.3”图标，进入CCS设置窗口；

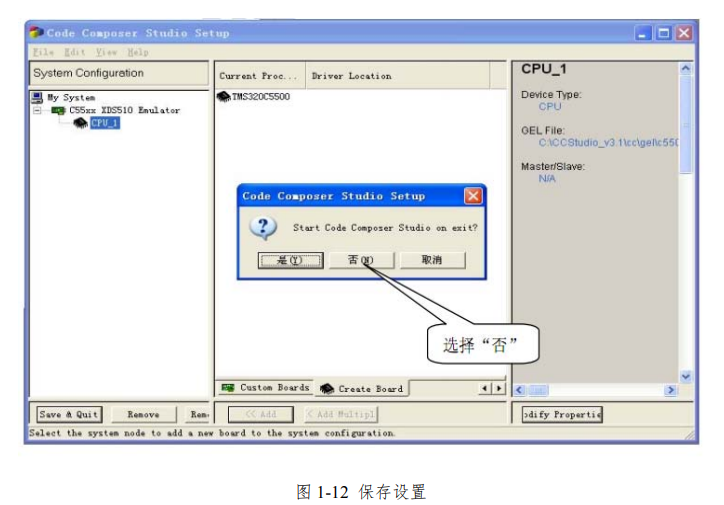
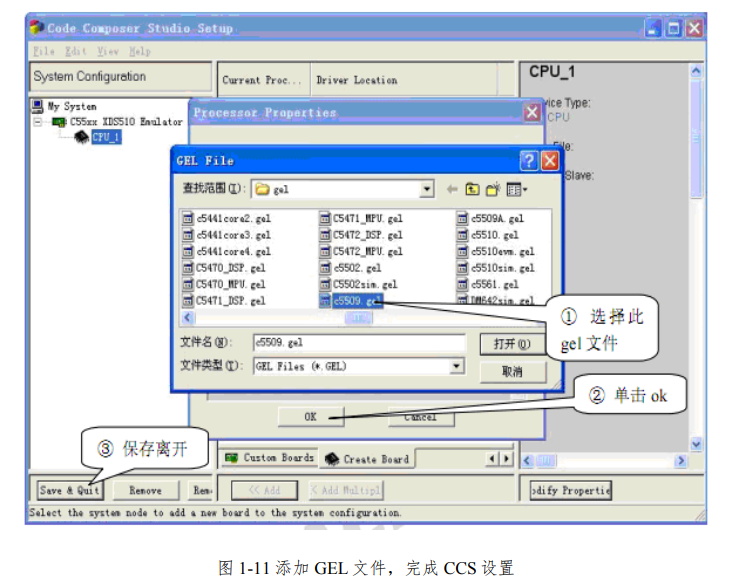
(b)在出现的窗口中，按标号顺序进行如图 1-7 设置：

(c)在出现的窗口中，按标号顺序进行如图 1-8 设置：

(d)在出现的窗口中，按标号顺序进行如图 1-9 设置：

(e)在出现的窗口中，按标号顺序进行如图 1-10 所示的选择：

(f)在出现的窗口中，按标号顺序进行如图 1-11 所示的选择：

****(g)在如图 1-12 出现的窗口中，选择“否（N）”。

以上设置完成后，CCS 已经被设置成 Emulator 的方式(用仿真器连接硬件板卡的方式)，并且指定通过 ICETEK-5100USB 仿真器连接 ICETEK–VC5509-A评估板。如果以后一直使用这一方式就不需要重新进行以上设置操作了，**在后续两个是实验中的emulator环境配置与此处相同，因此在后续两个实验中不再重复叙述。**

**(2) 启动 CCS；**

启动 Emulator（硬件仿真）方式：

(a) 首先将实验箱电源关闭。连接实验箱的外接电源线。

(b) 检查 ICETEK-5100USB 仿真器的黑色 JTAG 插头是否正确连接到 ICETEK–VC5509-A 评估板的 J1 插头上。注：仿真器的插头中有一个孔加入了封针，与J1 插头上的缺针位置应重合，保证不会插错。

(c) 检查其他连线是否符合实验要求。检查实验箱上三个拨动开关位置是否符合实验要求。

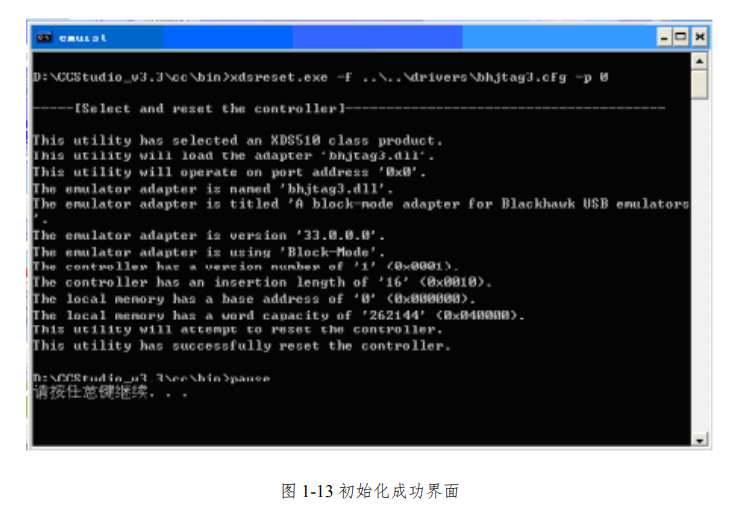
(d) 打开实验箱上电源开关（位于实验箱底板左上角），注意开关边上红色指示灯点亮。ICETEK–VC5509-A 评估板上指示灯 D5 和 D6 点亮。如果打开了ICETEK-CTR 的电源开关，ICETEK-CTR 板上指示灯 L1、L2 和 L3 点亮。如果打开了信号源电源开关，相应开关边的指示灯点亮。

(e) 用实验箱附带的 USB 信号线连接 ICETEK-5100USB 仿真器和 PC 机后面的USB 插座。注意 ICETEK-5100USB 仿真器上指示灯 Power 和 Run 灯点亮。

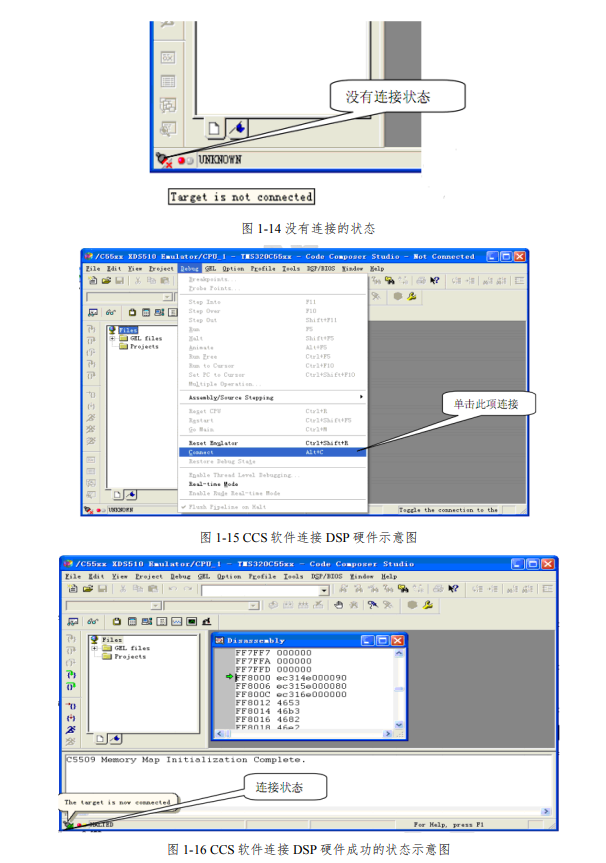


(f) 双击桌面上“emurst”仿真器初始化图标：

如果出现如图 1-13 提示窗口，表示初始化成功，按一下空格键进入下一步操作。

如果窗口中没有出现“按任意键继续…”，请关闭窗口，关闭实验箱电源，再将 USB 电缆从仿真器上拔出，返回第（2）步重试。

如果窗口中出现“The adapter returned an error.”，并提示“按任意键继续…”，表示初始化失败，请关闭窗口重试两三次，如果仍然不能初始化则关闭实验箱电源，再将 USB 电缆从仿真器上拔出，返回第（2）步重试。

(g) 双击桌面上的“CCStudio v3.3”图标 ，启动 CCS；在出现的窗口中，按标号顺序进行如图 1-14,图 1-15,图 1-16 的操作：

(h) 如果进入 CCS 提示错误，先选“Abort”，然后用“初始化 ICETEK5100USB2.0 仿真器”初始化仿真器，如提示错误，可多做几次。如仍然出错，拔掉仿真器上 USB 接头（白色方形），按一下 ICETEK–VC5509-A 评估板上S1 复位按钮，连接 USB 接头，再做“初始化 ICETEK-5100USB2.0 仿真器”。

(i) 如果遇到反复不能连接或复位仿真器、进入报错，请打开 Windows 的“任务管理器”，在“进程”卡片上的“映像名称”栏中查询 5100USB2 是否有“cc\_app.exe”，将它结束再试。

(j) 选择菜单 Debug->Reset CPU

**在后续两个实验中的Emulator实验中的环境配置操作与该实验相同，因此后续不再赘述。**

**(3) 打开工程文件；**

浏览程序：工程文件目录为 C:\ICETEK\VC5009Ae\ VC5009Ae\ Lab0501-FIR\Fir.pjt

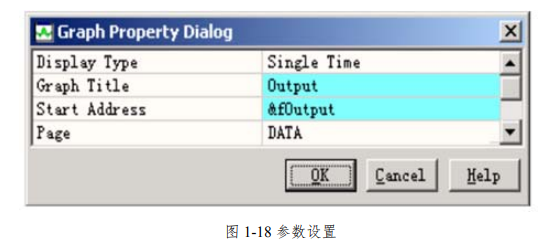
打开方法：选择 Project->Open，选择 Lab0501-FIR 文件夹下的 Fir.pjt。

**(4) 编译并下载程序；**

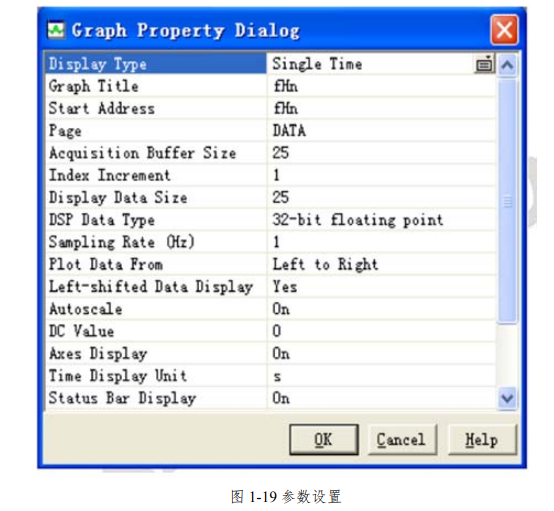
选择 Project->Rebuild All 编译，然后下载程序：选 File->Load Program，在Debug 文件夹下选择 Fir.out，并打开。

**(5) 打开观察窗口；**

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 1-17 所示：

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 1-18 所示：

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 1-19 所示：

在弹出的图形窗口中单击鼠标右键，选择“Clear Display”。

**(6) 设置断点；**

在工程管理窗中，双击 Fir.c，在有注释“break point”的语句处设置软件断点，方法：双击该语句左侧灰色控制条（出现红色圆点）。

**(7) 运行并观察结果；**

(a) 选择“Debug”菜单的“Animate”项，或按 Alt+F5 键运行程序。

(b) 观察“Input”、“Output”窗口中时域图形，并保存；观察滤波效果。

(c) 鼠标右键单击“Input”和“Output”窗口，选择“Properties…”项，设置“Display Type”为“FFT Magitude”，再单击“OK”按钮结束设置。

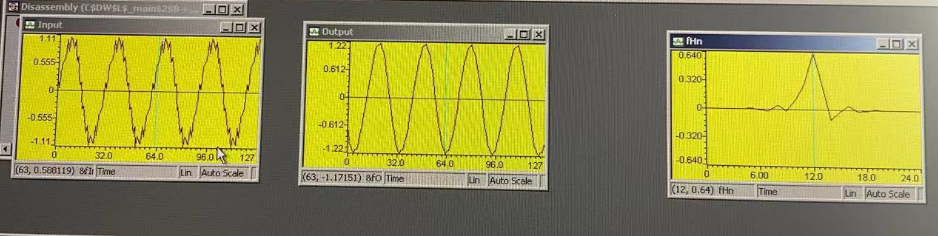
(d) 观察“Input”、“Output”窗口中频域图形，并保存；理解滤波效果。

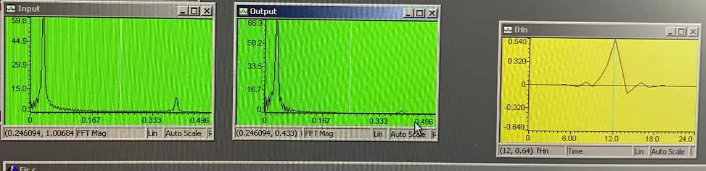
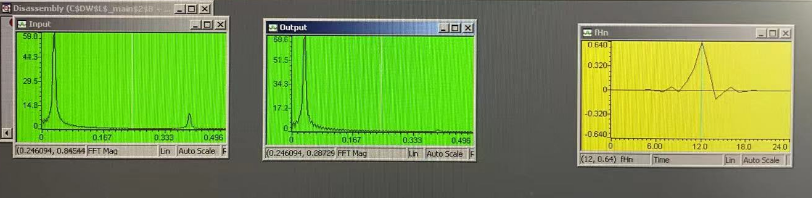
**(8) 退出 CCS。**

**四、实验结果及体会**

**1、实验结果**

经过调试可以看出FIR滤波器满足设计要求，滤波效果十分显著，如下为软件仿真Simulator时域和频域实验结果：



如下为硬件仿真Emulator时域和频域实验结果：

**2、实验体会**

在基本相同的实验结果下，在实验过程中可以看出硬件仿真相较于软件仿真建立稳定波形的时间明显较长，这可能也是日常学习实验中理想与实际的区别，实际状况中会出现较多理想情况下无法考虑到的状况，在本实验中，建立时间对最终的实验结果并无太大影响，但对于未来可能遇到的对实验结果产生影响的状况，应熟悉底层的基本原理，运用自己的知识进行处理。

**五、实验原理的认识或实验程序清单或实验结果的分析**

由上述实验结果及对底码的阅读可知，该FIR数字滤波器的设计采用的是窗函数法，且采用了布莱克曼窗，且根据指标计算窗宽应设置为25，最终所得的滤波器是一个低通滤波器。

Blackman窗函数可以提供较窄的主瓣宽度和较高的副瓣抑制，但是会引入较大的过渡带宽度。因此，选择窗函数时需要根据具体的设计要求来平衡主瓣宽度、副瓣抑制和过渡带宽度的需求。

以下附实验源代码：

#include "myapp.h"

#include "ICETEK-VC5509-EDU.h"

#include "scancode.h"

#include <math.h>

#define FIRNUMBER 25

#define SIGNAL1F 1000

#define SIGNAL2F 4500

#define SAMPLEF 10000

#define PI 3.1415926

float InputWave();

float FIR();

float fHn[FIRNUMBER]={ 0.0,0.0,0.001,-0.002,-0.002,0.01,-0.009,

-0.018,0.049,-0.02,0.11,0.28,0.64,0.28,

-0.11,-0.02,0.049,-0.018,-0.009,0.01,

-0.002,-0.002,0.001,0.0,0.0

};

float fXn[FIRNUMBER]={ 0.0 };

float fInput,fOutput;

float fSignal1,fSignal2;

float fStepSignal1,fStepSignal2;

float f2PI;

int i;

float fIn[256],fOut[256];

int nIn,nOut;

main()

{

nIn=0; nOut=0;

f2PI=2\*PI;

fSignal1=0.0;

fSignal2=PI\*0.1;

fStepSignal1=2\*PI/30;

fStepSignal2=2\*PI\*1.4;

while ( 1 )

{

fInput=InputWave();

fIn[nIn]=fInput;

nIn++; nIn%=256;

fOutput=FIR();

fOut[nOut]=fOutput;

nOut++; /\* break point \*/

if ( nOut>=256 )

{

nOut=0;

}

}

}

/\* InputWave函数产生正弦波信号fXn，包含了高频信号和低频信号，是两者的叠加，波形图上看是有毛刺、不光滑的。

fXn[0]=sin((double)fSignal1)+cos((double)fSignal2)/6.0;

这里产生的信号就是fXn，对应波形的Y轴。波形的X轴对应的则是离散的步进时间点。\*/

float InputWave()

{

for ( i=FIRNUMBER-1;i>0;i-- )

fXn[i]=fXn[i-1];

fXn[0]=sin((double)fSignal1)+cos((double)fSignal2)/6.0;

fSignal1+=fStepSignal1;

if ( fSignal1>=f2PI ) fSignal1-=f2PI;

fSignal2+=fStepSignal2;

if ( fSignal2>=f2PI ) fSignal2-=f2PI;

return(fXn[0]);

}

/\*在滤波模块FIR函数中，fSum是每个时间点的滤波输出，fSum+=(fXn[i]\*fHn[i]);是对fXn的滤波输出。\*/

float FIR()

{

float fSum;

fSum=0;

for ( i=0;i<FIRNUMBER;i++ )

{

fSum+=(fXn[i]\*fHn[i]);

}

return(fSum);

}

**实验名称：** **IIR数字滤波器的设计**

**姓名: 学号: 同组： 无 实验日期:**

**一、实验目的**

(1) 了解和熟悉 ICETECK-DSP 教学实验系统平台；

(2) 了解和熟悉 DSP 开发集成环境 Code Composer Studio 3.3 软件的设置和使用；

(3) 了解 IIR 数字滤波器的特性和设计原理；

(4) 掌握 IIR 数字滤波器的设计方法。

**二、实验内容**

使用配置好的PC机以及ICETECK-DSP实验箱，熟悉无限冲激响应数字滤波器的基础理论；了解模拟滤波器的原理；了解数字滤波器系数的确定方法；根据要求设计低通IIR数字滤波器。

**滤波器设计指标：**

低通巴特沃斯滤波器在其通带边缘 1kHz 处的增益为-3dB，12kHz 处的阻带衰减为 30dB，采样频率 25kHz。

**滤波器设计：**

确定待求通带边缘频率 fp1Hz、待求阻带边缘频率 fs1Hz 和待求阻带衰减-20logδsdB。

模拟边缘频率为：fp1=1000Hz，fs1=12000Hz

阻带边缘衰减为：-20logδs=30dB

用 Ω=2πf/fs 把由 Hz 表示的待求边缘频率转换成弧度表示的数字频率，得到 Ωp1和 Ωs1。

Ωp1=2πfp1/fs=2π1000/25000=0.08π 弧度

Ωs1=2πfs1/fs=2π12000/25000=0.96π 弧度

计算预扭曲模拟频率以避免双线性变换带来的失真。

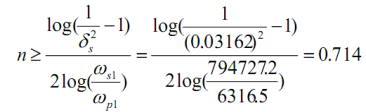
由 ω=2fs tan(Ω/2)求得ωp1 和ωs1，单位为弧度/秒。

ωp1 =2fs tan(Ωp1/2)=6316.5 弧度/秒

ωs1=2fs tan(Ωs1/2)=794727.2 弧度/秒

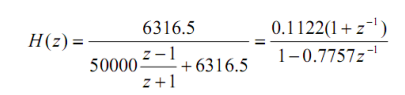
由已给定的阻带衰减-20logδs 确定阻带边缘增益 δs。

因为-20logδs=30，所以 logδs=-30/20，δs=0.03162

计算所需滤波器的阶数：

因此，一阶巴特沃斯滤波器就足以满足要求。

一阶模拟巴特沃斯滤波器的传输函数为：H(s)= ωp1 /(s+ωp1) =6316.5/(s+6316.5)

由双线性变换定义 s=2fs(z-1)/(z+1)得到数字滤波器的传输函数为：

因此，差分方程为：y[n]=0.7757y[n-1]+0.1122x[n]+0.1122x[n-1]。

**三、实验方案及流程图**

**1.软件仿真Simulator**

**(1) 设置 CCS；**

**(2) 启动 CCS；**

**(3) 打开工程文件；**

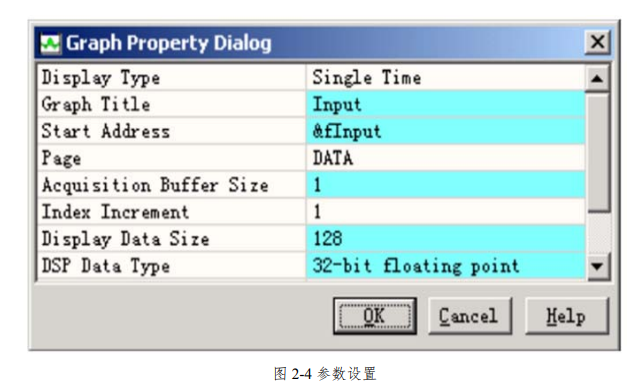
浏览程序：工程文件目录为 C:\ICETEK\VC5009Ae\VC5009Ae\ Lab0502-IIR\IIR.pjt

打开方法：选择 Project->Open，选择 Lab0502-IIR 文件夹下的 IIR.pjt，

**(4) 编译并下载程序；**

选择 Project->Rebuild All 编译，然后下载程序：选 File->Load Program，在Debug 文件夹下选择 IIR.out，并打开。

**(5) 打开观察窗口；**

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 2-4 所示：

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 2-5 所示：



**(6) 清除显示；**

在以上打开的窗口中单击鼠标右键，选择弹出式菜单中“Clear Display”。

**(7) 设置断点；**

在工程管理窗中，双击 iir.c，在程序 iir.c 中有注释“break point”的语句上设置软件断点。

方法：双击该语句左侧灰色控制条（会出现红色圆点）。

**(8) 运行并观察结果：**

(a) 选择“Debug”菜单的“Animate”项，或按 Alt+F5 键运行程序。

(b) 观察“IIR”窗口中时域图形，并保存；观察滤波效果。

(c) 切换成频域波形；鼠标右键单击“Input”和“Output”窗口，选择 “Properties…”

项，设置“Display Type”为“FFT Magitude”，再单击“OK”按钮结束设置。观察并理解滤波效果。

(d) 观察“Input”、“Output”窗口中频域图形，并保存；理解滤波效果。

**(9) 退出 CCS。**

2.硬件仿真Emulator

**(1) 设置 CCS；**

**(2) 启动 CCS；**

**(3) 打开工程文件；**

浏览程序：工程文件目录为 C:\ICETEK\VC5009Ae\VC5009Ae\ Lab0502-IIR\IIR.pjt

打开方法：选择 Project->Open，选择 Lab0502-IIR 文件夹下的 IIR.pjt，

**(4) 编译并下载程序；**

选择 Project->Rebuild All 编译，然后下载程序：选 File->Load Program，在Debug 文件夹下选择 IIR.out，并打开。

**(5) 打开观察窗口；**

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 2-16 所示：

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 2-17 所示：

**(6) 清除显示；**

在以上打开的窗口中单击鼠标右键，选择弹出式菜单中“Clear Display”。

**(7) 设置断点；**

在工程管理窗中，双击 iir.c，在程序 iir.c 中有注释“break point”的语句上设置软件断点。

方法：双击该语句左侧灰色控制条（会出现红色圆点）。

**(8) 运行并观察结果：**

(a) 选择“Debug”菜单的“Animate”项，或按 Alt+F5 键运行程序。

(b) 观察“IIR”窗口中时域图形，并保存；观察滤波效果。

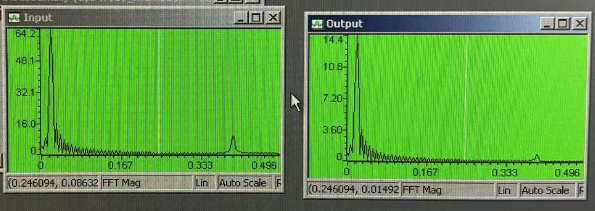
(c) 切换成频域波形；鼠标右键单击“Input”和“Output”窗口，选择“Properties…”项，设置“Display Type”为“FFT Magitude”，再单击“OK”按钮结束设置。观察并理解滤波效果。

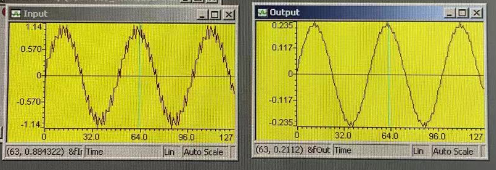
(d) 观察“Input”、“Output”窗口中频域图形，并保存；理解滤波效果。

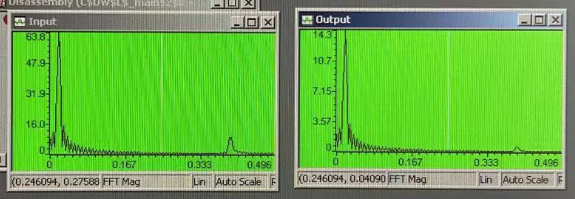
**(9) 退出 CCS。**

**四、实验结果及体会**

**1、实验结果**

经过调试可以看出IIR滤波器满足设计要求，滤波效果十分显著，如下为软件仿真Simulator时域和频域实验结果：

如下为硬件仿真Emulator时域和频域实验结果：

**2、实验体会**

在设计IIR的数字滤波器实验过程中，实验者发现，除了出现和FIR数字滤波器实验中类似的硬件仿真比软件仿真建立时间长，同样包含理想与实际的情况外，也可以直观地观察到IIR数字滤波器的波形没有FIR数字滤波器的波形光滑，这除了与输入的信号不同有关外，更深层次的原因在于——FIR滤波器是一种具有有限脉冲响应的数字滤波器，其脉冲响应在有限时间内变为零。FIR滤波器的特点是稳定性好，易于设计，且可以实现线性相位特性。FIR滤波器通常由一组延迟器和加权系数组成，没有反馈路径，因此不会引入稳定性问题。然而，FIR滤波器通常需要更高的阶数才能实现与IIR滤波器相似的性能，导致更高的计算复杂度。

IIR滤波器是一种具有无限脉冲响应的数字滤波器，其脉冲响应在有限时间内不会消失。IIR滤波器的特点是可以使用较低阶数实现与FIR滤波器相似的性能，因此通常具有更低的计算复杂度。然而，IIR滤波器容易引入稳定性问题，因为它们包含了反馈路径，这可能导致滤波器不稳定或者引入震荡。

总的来说，FIR滤波器具有稳定性好、易于设计、线性相位特性等特点，但通常需要更高的阶数；而IIR滤波器具有较低的计算复杂度，但可能存在稳定性问题。

**五、实验原理的认识或实验程序清单或实验结果的分析**

由设计指标可知，该IIR数字滤波器为巴特沃斯低通滤波器，阶数为一阶，模拟转数字采用了双线性变换法，将连续的模拟信号转为离散的数字信号。

以下附实验源代码：

#include"math.h"

#define IIRNUMBER 2

#define SIGNAL1F 1000

#define SIGNAL2F 4500

#define SAMPLEF 10000

#define PI 3.1415926

float InputWave();

float IIR();

float fBn[IIRNUMBER]={ 0.0,0.7757 };

float fAn[IIRNUMBER]={ 0.1122,0.1122 };

float fXn[IIRNUMBER]={ 0.0 };

float fYn[IIRNUMBER]={ 0.0 };

float fInput,fOutput;

float fSignal1,fSignal2;

float fStepSignal1,fStepSignal2;

float f2PI;

int i;

float fIn[256],fOut[256];

int nIn,nOut;

main()

{

nIn=0; nOut=0;

fInput=fOutput=0;

f2PI=2\*PI;

fSignal1=0.0;

fSignal2=PI\*0.1;

// fStepSignal1=2\*PI/30;

// fStepSignal2=2\*PI\*1.4;

fStepSignal1=2\*PI/50;

fStepSignal2=2\*PI/2.5;

while ( 1 )

{

fInput=InputWave();

fIn[nIn]=fInput;

nIn++; nIn%=256;

fOutput=IIR();

fOut[nOut]=fOutput;

nOut++; // break point

if ( nOut>=256 )

{

nOut=0;

}

}

}

float InputWave()

{

for ( i=IIRNUMBER-1;i>0;i-- )

{

fXn[i]=fXn[i-1];

fYn[i]=fYn[i-1];

}

fXn[0]=sin((double)fSignal1)+cos((double)fSignal2)/6.0;

fYn[0]=0.0;

fSignal1+=fStepSignal1;

if ( fSignal1>=f2PI ) fSignal1-=f2PI;

fSignal2+=fStepSignal2;

if ( fSignal2>=f2PI ) fSignal2-=f2PI;

return(fXn[0]);

}

float IIR()

{

float fSum;

fSum=0.0;

for ( i=0;i<IIRNUMBER;i++ )

{

fSum+=(fXn[i]\*fAn[i]);

fSum+=(fYn[i]\*fBn[i]);

}

return(fSum);

}

**实验名称： 快速傅里叶变换(FFT)算法**

**姓名: 学号: 同组： 无 实验日期:**

**一、实验目的**

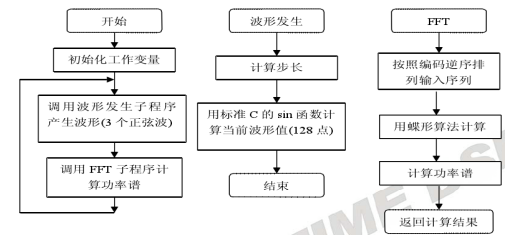
(1) 了解和熟悉 ICETECK-DSP 教学实验系统平台；

(2) 了解和熟悉 DSP 开发集成环境 Code Composer Studio 3.3 软件的设置和使用；

(3) 掌握 FFT 快速傅立叶变换的原理和方法；

(4) 熟悉 FFT 快速傅立叶变换特性。

**二、实验内容**

使用配置好的PC机以及ICETECK-DSP实验箱，了解FFT的原理，了解FFT的参数生成公式，了解FFT的程序流程图。

**三、实验方案及流程图**

**1.软件仿真Simulator**

**(1) 设置 CCS；**

**(2) 启动 CCS；**

**(3) 打开工程文件；**

浏览程序：工程文件目录为 C:\ICETEK\VC5009Ae\VC5009Ae\ Lab0503-FFT\FFT.pjt

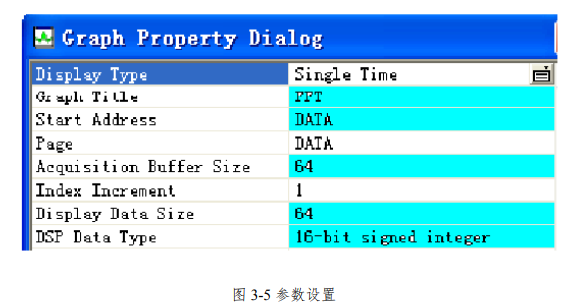
打开方法：选择 Project->Open，选择 Lab0503-FFT 文件夹下的 FFT.pjt，

**(4) 编译并下载程序；**

选择 Project->Rebuild All 编译，然后下载程序：选 File->Load Program，在Debug 文件夹下选择 FFT.out，并打开。

**(5) 打开观察窗口；**

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 3-4 所示：

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 3-5 所示：

**(6) 清除显示：**

在以上打开的窗口中单击鼠标右键，选择弹出式菜单中“Clear Display”。

**(7) 设置断点：**

在工程管理窗中，双击 FFT.c，在程序 FFT.c 中有注释“break point”的语句上设置软件断点。

方法：双击该语句左侧灰色控制条（会出现红色圆点）。

**(8) 运行并观察结果：**

(a) 选择“Debug”菜单的“Animate”项，或按 Alt+F5 键运行程序。

(b) 观察“Test Wave”窗口中时域图形；并保存；

(c) 在“Test Wave”窗口中点击右键，选择属性，切换成频域波形；鼠标右键单击“Input”和“Output”窗口，选择 “Properties…”项，设置“Display Type”为“FFT Magitude”，再单击“OK”按钮结束设置。观察频域图形。

(d) 观察“FFT”窗口中的由 CCS 计算出的正弦波的 FFT。

**(9) 退出 CCS。**

**2.硬件仿真Emulator**

**(1) 设置 CCS；**

**(2) 启动 CCS；**

**(3) 打开工程文件；**

浏览程序：工程文件目录为 C:\ICETEK\VC5009Ae\VC5009Ae\ Lab0503-FFT\FFT.pjt

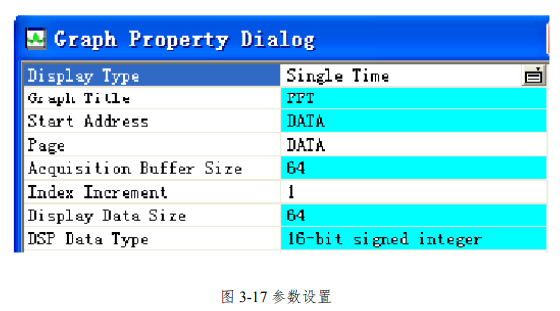
打开方法：选择 Project->Open，选择 Lab0503-FFT 文件夹下的 FFT.pjt，

**(4) 编译并下载程序；**

选择 Project->Rebuild All 编译，然后下载程序：选 File->Load Program，在Debug 文件夹下选择 FFT.out，并打开。

**(5) 打开观察窗口；**

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 3-16 所示：

选择菜单 View->Graph->Time/Frequency...进行相应设置。如图 3-17 所示：

**(6) 清除显示；**

在以上打开的窗口中单击鼠标右键，选择弹出式菜单中“Clear Display”。

**(7) 设置断点；**

在工程管理窗中，双击 FFT.c，在程序 FFT.c 中有注释“break point”的语句上设置软件断点。

方法：双击该语句左侧灰色控制条（会出现红色圆点）。

**(8) 运行并观察结果：**

(a) 选择“Debug”菜单的“Animate”项，或按 Alt+F5 键运行程序。

(b) 观察“Test Wave”窗口中时域图形；并保存；

(c) 在“Test Wave”窗口中点击右键，选择属性，切换成频域波形；鼠标右键单击“Input”和“Output”窗口，选择 “Properties…”项，设置“Display Type”为“FFT Magitude”，再单击“OK”按钮结束设置。观察频域图形。并保存。

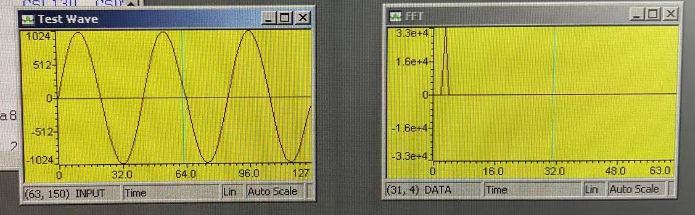
(d) 观察“FFT”窗口中的由 CCS 计算出的正弦波的 FFT。并保存。

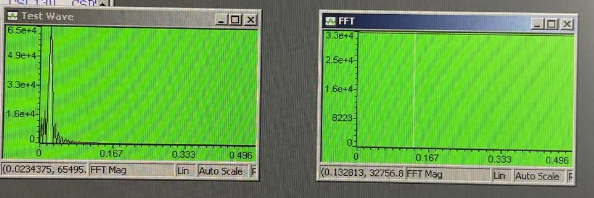
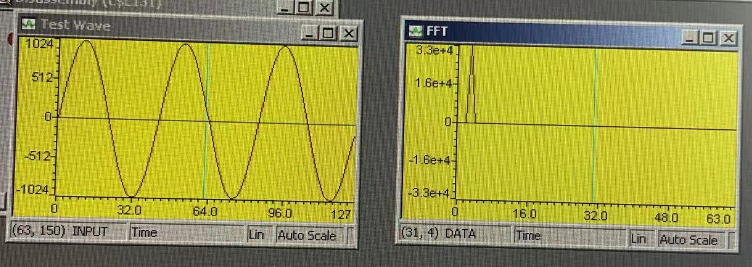
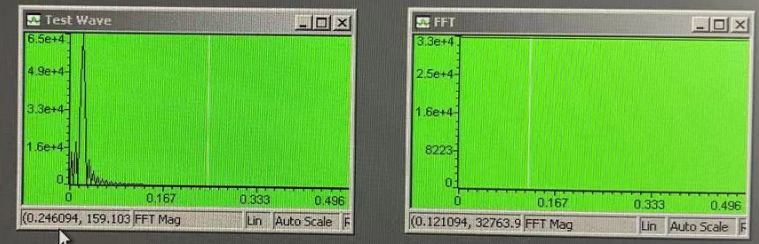
**(9) 退出 CCS。**

**四、实验结果及体会**

**1、实验结果**

根据要求构建了FFT的算法，如下为测试算法功能的软件仿真Simulator频域与时域实验结果：



如下为硬件仿真Emulator时域与频域实验结果：

**2、实验体会**

在FFT的实现算法当中，关键是要搞清楚蝶形算法的实现方法，它是一种分治算法，通过递归地将DFT（离散傅里叶变换）分解为较小规模的DFT，然后组合它们的结果来实现快速计算。

蝶形算法中的"蝶形"指的是一种计算单元，用于计算两个复数的乘积和加法。蝶形算法的基本思想是将DFT的计算过程分解为多个阶段，每个阶段包含多个蝶形计算单元，每个蝶形计算单元都可以并行计算。在每个阶段中，蝶形算法按照特定的规则对输入序列进行重新排列，然后进行蝶形计算和组合，最终得到DFT的结果。

蝶形算法通过不断地将规模较大的DFT分解为规模较小的DFT，并结合它们的结果，从而大大减少了计算的复杂度。这种算法的时间复杂度为O(N log N)，远远快于朴素的DFT算法的时间复杂度O(N^2)。

**五、实验原理的认识或实验程序清单或实验结果的分析**

实验采用的是128点的FFT算法，共有log2(128)=7级，每级有128/2=64个蝶形，所以共有7\*64=448个蝶形,且是DIT类型，输入序列码位倒置。

以下附实验源代码：

#include "math.h"

#define PI 3.1415926

#define SAMPLENUMBER 128

void InitForFFT();

void MakeWave();

void FFT();

int INPUT[SAMPLENUMBER],DATA[SAMPLENUMBER];

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

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

main()

{

int i;

InitForFFT();

MakeWave();

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

{

fWaveR[i]=INPUT[i];

fWaveI[i]=0.0f;

w[i]=0.0f;

}

FFT(fWaveR,fWaveI);

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

{

DATA[i]=w[i];

}

while ( 1 ); // break point

}

/\* FFT(fWaveR,fWaveI)则是FFT算法主体，具体包括：

1）输入序列码位倒置（注释为following code invert sequence的部分）

2）蝶型算法（注释为following code FFT的部分）

3）计算功率谱（最后一个for循环）\*/

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

{

int x0,x1,x2,x3,x4,x5,x6,xx;

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

float TR,TI,temp;

/\*\*\*\*\*\*\*\*\*\* following code invert sequence \*\*\*\*\*\*\*\*\*\*\*\*/

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

{

x0=x1=x2=x3=x4=x5=x6=0;

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;

xx=x0\*64+x1\*32+x2\*16+x3\*8+x4\*4+x5\*2+x6;

dataI[xx]=dataR[i];

}

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

{

dataR[i]=dataI[i]; dataI[i]=0;

}

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

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

{ /\* for(1) \*/

b=1; i=L-1;

while ( i>0 )

{

b=b\*2; i--;

} /\* b= 2^(L-1) \*/

for ( j=0;j<=b-1;j++ ) /\* for (2) \*/

{

p=1; i=7-L;

while ( i>0 ) /\* p=pow(2,7-L)\*j; \*/

{

p=p\*2; i--;

}

p=p\*j;

for ( k=j;k<128;k=k+2\*b ) /\* for (3) \*/

{

TR=dataR[k]; TI=dataI[k]; temp=dataR[k+b];

dataR[k]=dataR[k]+dataR[k+b]\*cos\_tab[p]+dataI[k+b]\*sin\_tab[p];

dataI[k]=dataI[k]-dataR[k+b]\*sin\_tab[p]+dataI[k+b]\*cos\_tab[p];

dataR[k+b]=TR-dataR[k+b]\*cos\_tab[p]-dataI[k+b]\*sin\_tab[p];

dataI[k+b]=TI+temp\*sin\_tab[p]-dataI[k+b]\*cos\_tab[p];

} /\* 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]);

}

} /\* END FFT \*/

/\* InitForFFT()是初始化工作变量，建立sin和cos的查找表。\*/

void InitForFFT()

{

int i;

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

{

sin\_tab[i]=sin(PI\*2\*i/SAMPLENUMBER);

cos\_tab[i]=cos(PI\*2\*i/SAMPLENUMBER);

}

}

/\* MakeWave()生成128点的正弦波形\*/

void MakeWave()

{

int i;

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

{

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

}

}