DSP 应用实验报告

实验十二：FIR 滤波器的 DSP 实现

院　　系：电子工程与光电技术学院

专　　业：电子信息工程

姓　　名：赵婧萱

学　　号：920104330118

指导老师：李彧晟

2023 年 5 月 29 日

目录

实验十二：FIR 滤波器的 DSP 实现

12.1 实验目的

1.巩固数字 FIR 滤波器的概念

2.理解定点 DSP 中数的定标、有限字长、溢出等概念

3.理解算法实现中实时的概念

4.掌握 DSP 开发过程以及基本调试方法

12.2 实验仪器

计算机，TMS320F28335 DSP 教学实验箱，XDS510 USB 仿真器，示波器，信号源

12.3 实验内容

针对 FIR 算法，设计滤波器系数，完成数据的定标，查看滤波器特性曲线。建立工程，编写 DSP 的主程序，并对工程进行编译、链接，利用现有 DSP 平台实现 FIR 滤波器算法，通过信号源、示波器理解滤波器特性，验证实现与理论设计的一致性。

12.4 实验步骤

12.4.1 滤波器系数求解

本实验在实验 11 的基础上，对采集到的数据进行滤波处理，嵌入 FIR 运算。我们采用 Matlab 软件设计滤波器系数。利用自带函数 fir1（），输入相关参数即可得到滤波器系数。其完整命令如下：

$$h = \mathrm{fir1}(n, Wn, 'ftype', window)$$

其中，n 为滤波器阶数，Wn 为归一化截止频率（此处指与采样频率一半进行

归一化),Wn 对应在幅频特性曲线上-6dB 点的频率数值,ftype 为滤波器类型，分为低通、带通、高通、带阻形式，window 为使用的窗函数，可以为 hamming（汉明窗）、hanning（汉宁窗）、chebwin（切比雪夫窗）等形式；h 为产生的滤波器系数。

根据实验要求，我们小组需要设计带通滤波器。设定相关参数，最终设计为一个 48 阶、频率范围为 2kHz 到 5kHz 的带通滤波器，采样频率设定为 40kHz。Matlab 设计程序如下：

```
clc;
clear all;
close all;
h=fir1(48,[0.10,0.25],'bandpass');
figure;
freqz(h);
```

对产生的滤波器系数用 freqz 命令查看其幅频、相频特性曲线。操作后如**图 1** 所示：

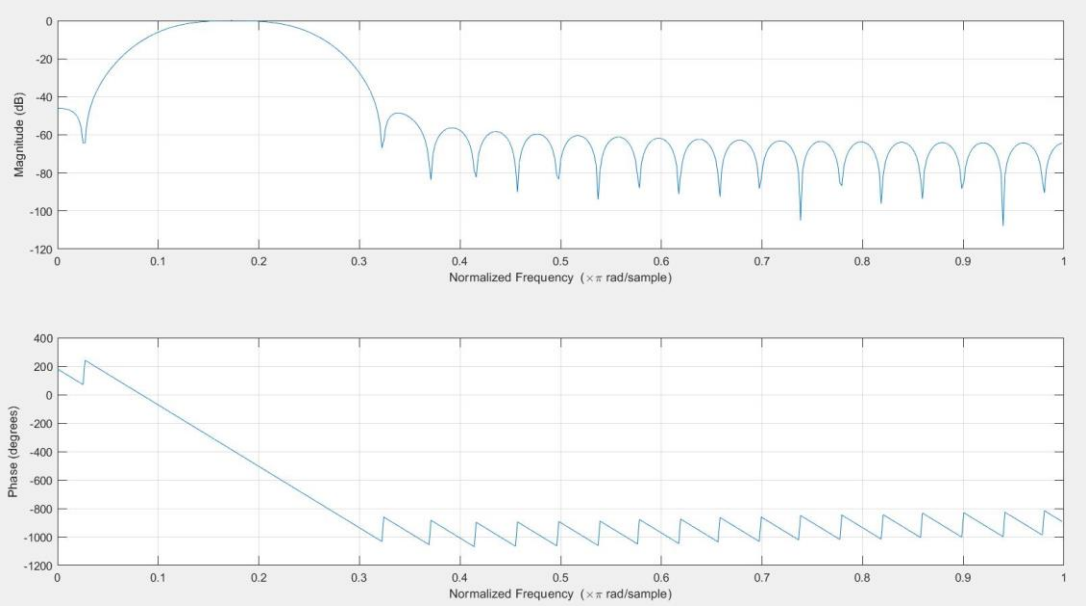


图 1：带**通**滤波器的幅频/**相**频特性曲线

同时，在工作区得到滤波器各阶系数，如**表 1** 所示：

表 1：滤波器各阶滤波器系数

h	值	h	值	h	值	h	值	h	值
1	-0.00101	13	0.00842	25	0.14998	37	0.00842	49	-0.00101
2	-0.00176	14	0.01764	26	0.12620	38	0.00120		
3	-0.00220	15	0.02098	27	0.06457	39	-0.00047		
4	-0.00177	16	0.01008	28	-0.01043	40	0.00226		
5	0.00000	17	-0.01801	29	-0.07101	41	0.00586		
6	0.00298	18	-0.05653	30	-0.09833	42	0.00738		
7	0.00603	19	-0.08955	31	-0.08955	43	0.00603		
8	0.00738	20	-0.09833	32	-0.05653	44	0.00298		
9	0.00586	21	-0.07101	33	-0.01801	45	0.00000		
10	0.00226	22	-0.01043	34	0.01008	46	-0.00177		
11	-0.00047	23	0.06457	35	0.02098	47	-0.00220		
12	0.00120	24	0.12620	36	0.01764	48	-0.00176		

将所得的滤波系数保存下来，为后续数据的定标修改做准备。

12.4.2 数据定标

取定标 Q=15，即将数据乘$2^{15} = 32768$，之后取整，其余的数值也做此操作。对应 Matlab 的命令如下：

```
H1=2^15*h;
H2=floor(H1);
hold on;
freqz(H2);
```

式中，floor()函数功能为“向下取整”。通过上述系列操作得到如**表 2** 所示的定标后的滤波器系数；

表 2：数据定标后的滤波器系数

h	值	h	值	h	值	h	值	h	值
1	-34	13	275	25	4914	37	275	49	-34
2	-58	14	578	26	4135	38	39		
3	-73	15	687	27	2115	39	-16		
4	-59	16	330	28	-342	40	74		
5	0	17	-591	29	-2327	41	192		
6	97	18	-1853	30	-3223	42	241		
7	197	19	-2935	31	-2935	43	197		
8	241	20	-3223	32	-1853	44	97		
9	192	21	-2327	33	-591	45	0		
10	74	22	-342	34	330	46	-59		
11	-16	23	2115	35	687	47	-73		
12	39	24	4135	36	578	48	-58		

利用 freqz（）函数作出定标后系数的幅频、相频特性曲线，同时利用

Matlab 命令“hold on”将定标前后的特性曲线展示在同一张绘图上以便于对照。具体实现效果如**图 2** 所示，其中红色标线为数据定标后，由于数据扩大并且采用舍入运算截断小数，在某些特征点上会产生误差，在后续实际测试阶段可能会影响性能。

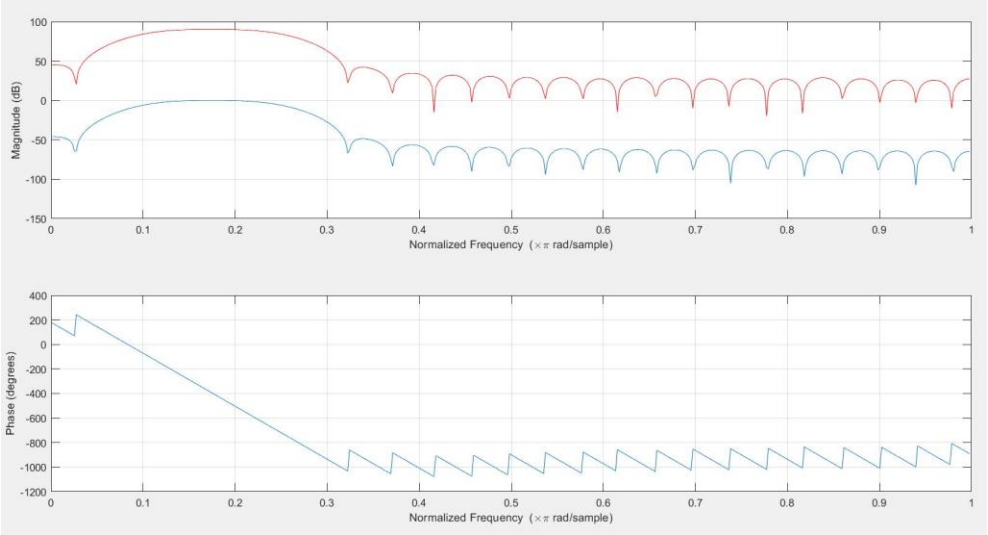


图 2：**数据定**标后的幅频相频特性曲线

由此，我们完成了 h 系数从浮点数到定点数的转换。利用 Matlab 软件，设置函数对设计的 FIR 滤波器进行仿真验证。实现代码如下图所示：

```
fs =40e3;
Ts=1/fs;
t=0:Ts:200*Ts;
y=zeros(4,length(t));
y_filter=zeros(4,length(t));
y_filter_fft=zeros(4,length(t));
i=1;
figure();
for f=1.5:1:6.5
    y(i,:)=cos(2*pi*f*1e3*t);
    y_filter(i,:)=filter(h,1,y(i,:));
    subplot(2,3,i);
    plot(t,y_filter(i,:));
    axis([0 max(t) -2 2]);
    title(sprintf("%.1fkHz下输出波形",f));
    i=i+1;
end
figure();
```

```
f=1;
for i=1:6
    y_filter_fft(i,:)=fftshift(abs(fft(y_filter(i,:)))/(length((fft(y_filter(i,:)))/2));
    subplot(2,3,i);
    plot([-fs/2:fs/length(y_filter(i,:)):fs/2-(fs/length(y_filter_fft(i,:)))],y_filter_fft(i,:));
    axis([-fs/2 fs/2 0 1.2]);
    title(sprintf('输出频谱%d',i));
    f=f+0.5;
end
```

图 3：Matlab 验证滤波器功能

仿真后为得到特定频率下输出波形图，如下列**图 4、图 5** 所示：

1. 输入波形（频率分别对应 1.5、2.5、3.5、4.5、5.5、6.5kHz）

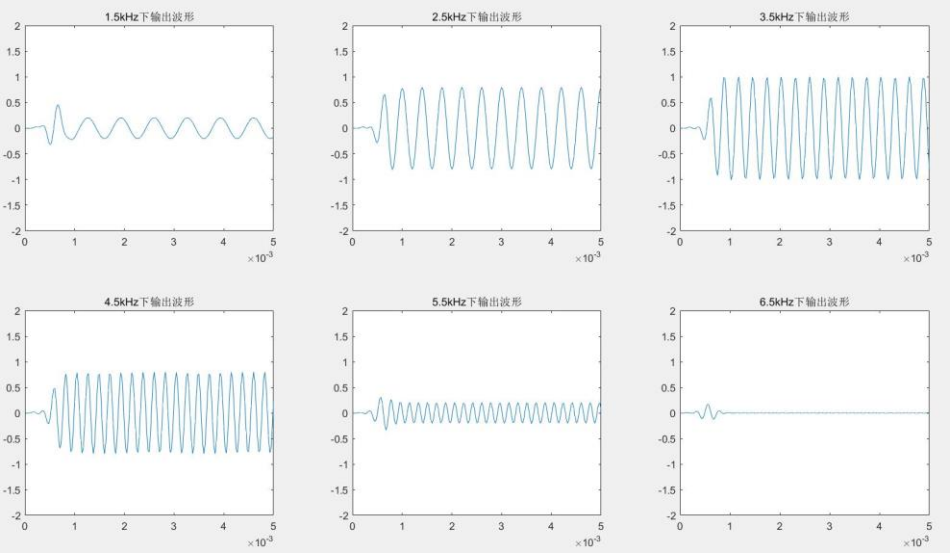


图 4：Matlab **仿真**输入波形

2. 输出滤波波形（频率分别对应 1.5、2.5、3.5、4.5、5.5、6.5kHz）

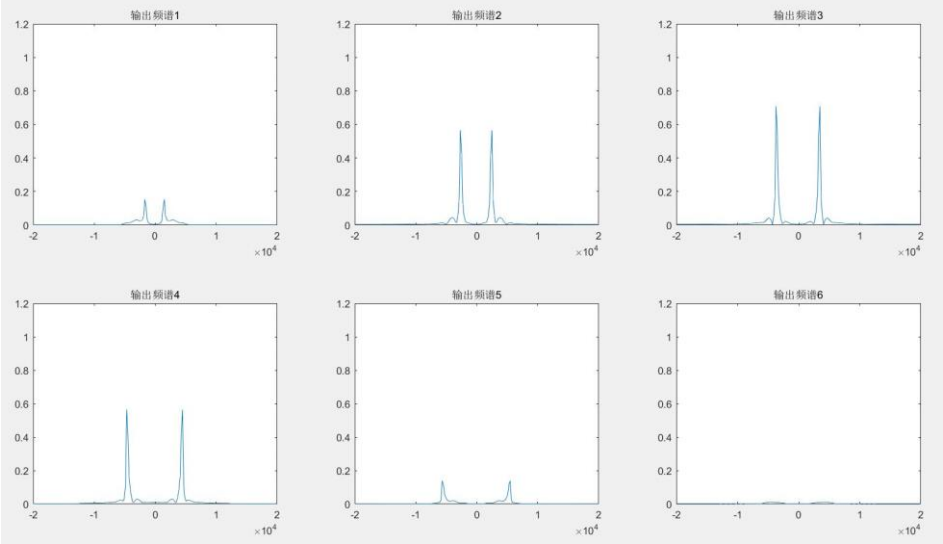


图 5：Matlab **仿真**输出波形

仿真模拟的结果如下：在 2kHz 到 5kHz 的通带内，正弦波输入时，可以得到正常的输出信号，随着输入频率逐步逼近截止频率，输出波形的幅度逐渐变小，超出一定范围之后，模拟输出波形的频率不再符合输入且幅度极小，将该仿真结果保留，作为后续实际验证的参照。

12.4.3 FIR 计算模块设置

有限长单位冲激响应滤波器（FIR）差分方程表示为：

$$y[n] = \sum_{k=0}^{N} h[k]x[n-k]$$

其中，h 为滤波器系数、x 为输入的数字信号、y 为 FIR 滤波器的计算输出、N 为滤波器阶数。由此可得，一个 N 阶滤波器计算，需要 N+1 个滤波器系数，N+1 个数字输入；每得到一个 y 的值，需要 N+1 次乘法以及 N 次加法。另外，N 阶滤波器需要保存当前的 N+1 个输入信号数值。

程序中，由于涉及到 16 位有符号、无符号数相乘等运算，涉及到字长的变化。x 为定点数（16 位，低 4bit 无效）、h 为浮点数（16bit），二者相乘为 32bit；为保证相乘运算两方均为同类型，我们将 h 改为定点数（即数据定标方式），则运算出 y 为定点数，输出给 DA 之前需截取 16 位定点数；同时考虑到数据动态范围，y 的位数不能限定于 32bit，于是我们设定 x 为 unsigned int 类型，y 为 long long int 类型，xn 数组为 long int 类型；计算模块功能如图所示：

```
42
43 long int xn[49]={0};
44 Uint32 x;
45 long long int y;
46 int h[49]={-34,-58,-73,-59,0,97,197,241,192,74,-16,39,275,578,687,330,-591,
47         -1853,-2935,-3223,-2327,-342,2115,4135,4914,4135,2115,-342,-2327,
48         -3223,-2935,-1853,-591,330,687,578,275, 39,-16,74,192,241,197,97,0,-59,-73,-58,-34};
49
50 void main(void)
```

```
366     //滤波
367     for (y=0,k=0 ;k<49;k++)
368         y=y+xn[48-k]*h[k];
```

图 6：FIR **差分方程**计算功能

12.4.4 FIR 算法实现

为实现滤波器输出的实时更新，在每一个采样信号到来时，需要系统知晓该信号前 48 个信号的数值。构建数组 xn 保存最新的 N 个采样数，每次调整完需要删除最旧的数据，将输入信号前移（即数组后移一位），使最新采样点数据存至最后，数组始终未按时间顺序排列的最新数据，实现该功能的代码如**图 7** 所示：

```
357 interrupt void epwm1_timer_adc_isr(void)     //中断函数
358 {
359     //读取ADC数据并更新xn数组
360     x= (AdcRegs.ADCRESULT1 & 0xFFF0);
361     int k;
362     for (ConvCount=0;ConvCount<48;ConvCount++){
363     xn [ConvCount]=xn [ConvCount+1];
364     }
365     xn [48]=x;
```

图 7：输入信号更新实现代码

将上述两部分代码写入 `interrupt void epwm1_timer_adc_isr(void)`，即完成 FIR 滤波器代码编写。

12.5 算法功能验证

设置信号源输出为峰值 1V、偏移 500mVpp 的正弦波，运行程序，逐步提高信号源输出频率，观察示波器的输出波形。如**图 8-图 25** 所示：

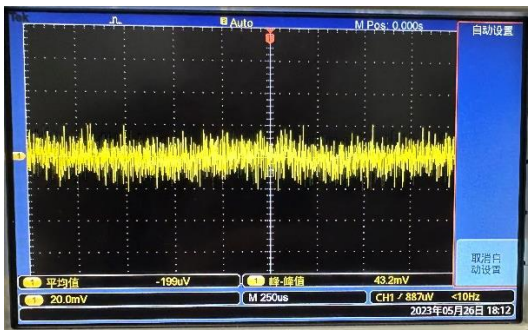


图 8：1000Hz 时的输出波形

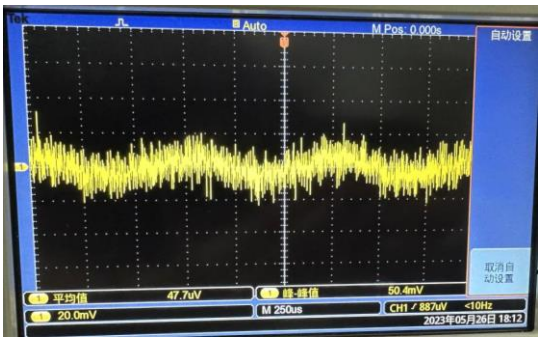


图 9：1300Hz 时的输出波形

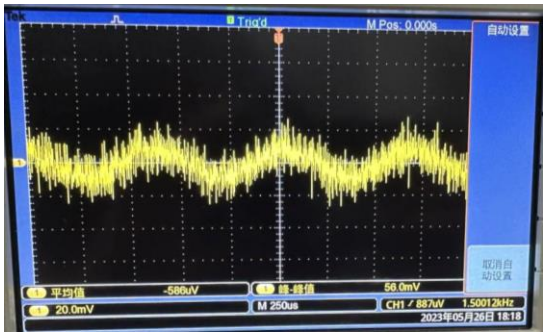


图 10：1500Hz 时的输出波形

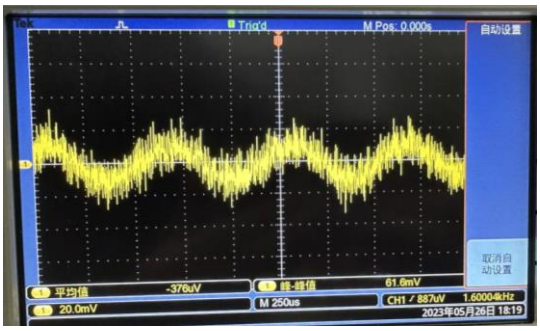


图 11：1600Hz 时的输出波形

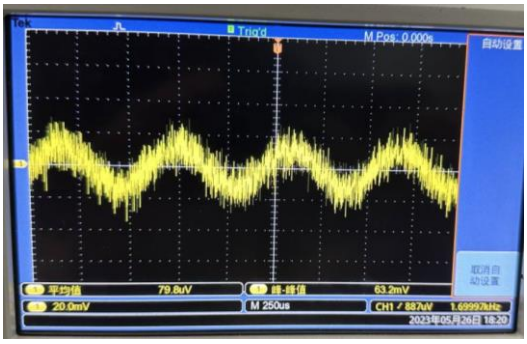


图 12：1700Hz 时的输出波形

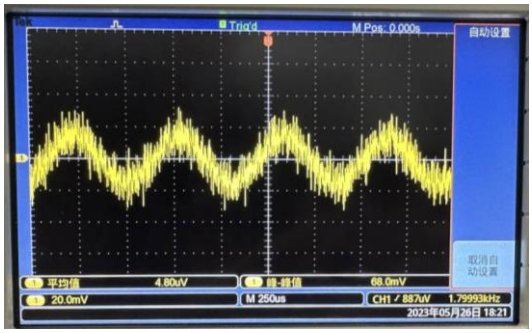


图 13：1800Hz 时的输出波形

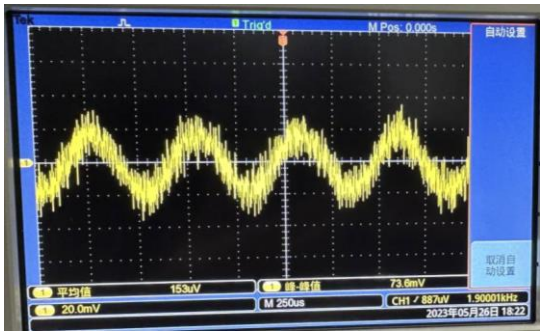


图 14：1900Hz 时的输出波形

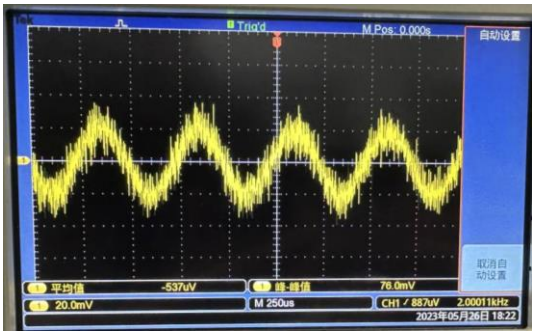


图 15：2000Hz 时的输出波形

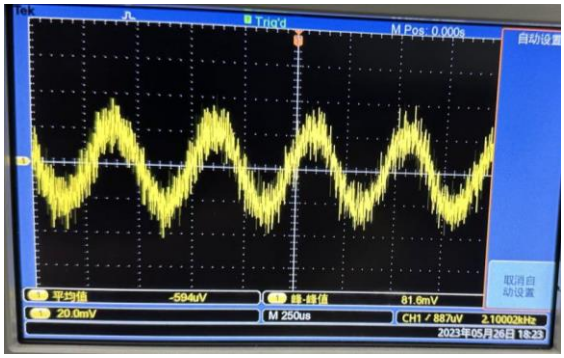


图 16：2100Hz 时的输出波形

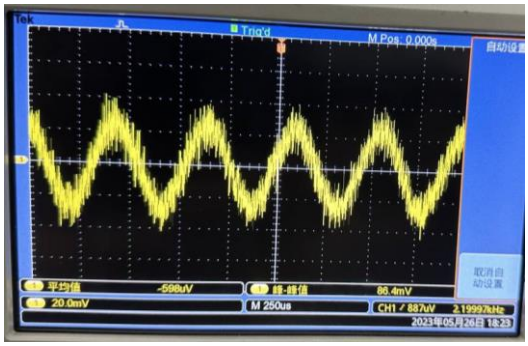


图 17：2200Hz 时的输出波形

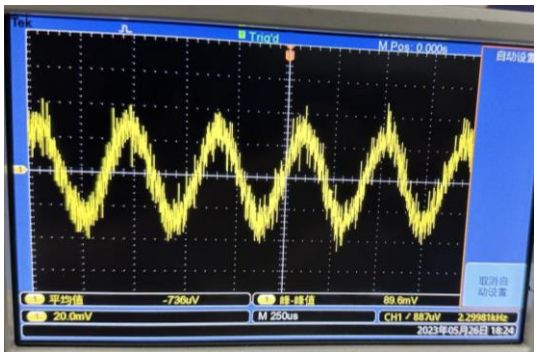


图 18：2300Hz 时的输出波形

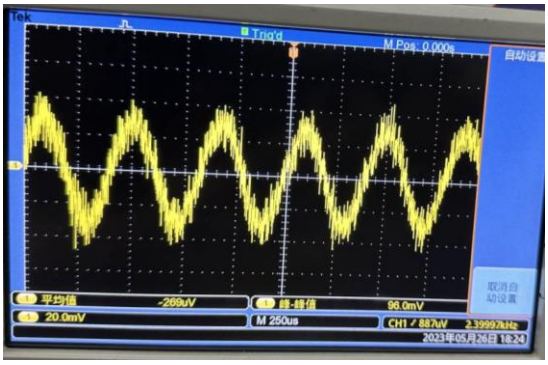


图 19：2400Hz 时的输出波形

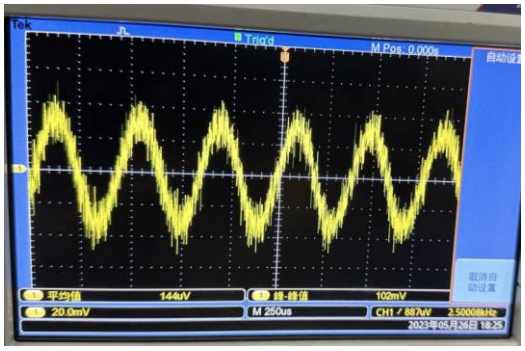


图 20：2500Hz 时的输出波形

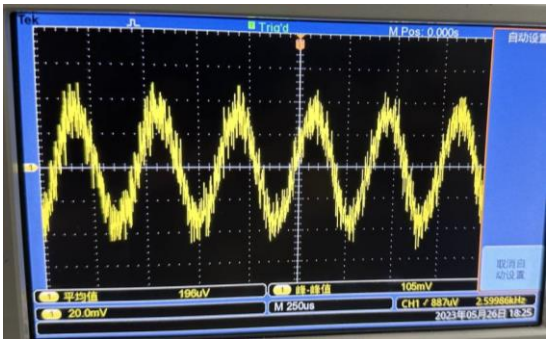


图 21：2600Hz 时的输出波形

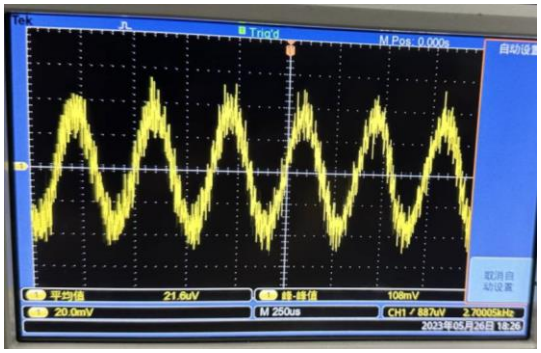


图 22：2700Hz 时的输出波形

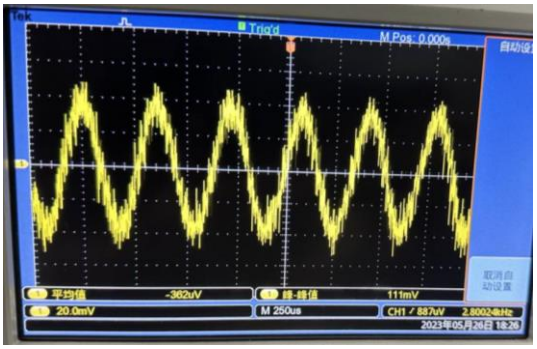


图 23：2800Hz 时的输出波形

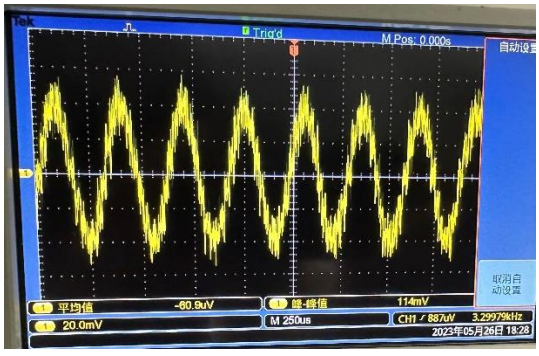


图 24：3300Hz 时的输出波形

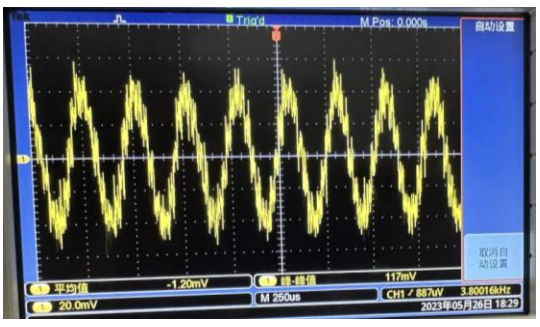


图 25：3800Hz 时的输出波形

可以看到随着输入频率的升高，示波器输出幅度逐渐增大，达到 2.5kHz 后，输出幅度趋于稳定区间，直到 4.5kHz 开始衰减，输出波形逐渐趋于直线；随着输入频率的继续增大，输出频率也不再正确反映输入信号，此时可以认为超出 5kHz 的信号被滤除。综合以上的功能验证，我们可以看出测试结果基本符合设计的带通滤波器幅频特性曲线。

衰减部分的具体输出波形如**图 26-图 33** 所示：

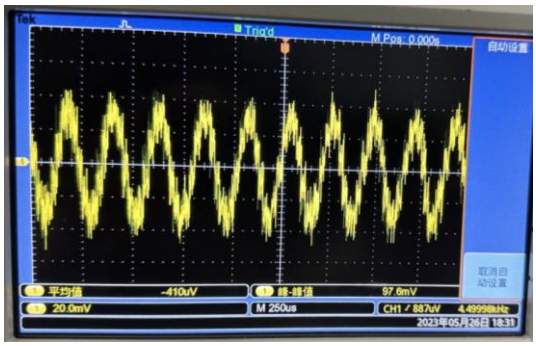


图 26：4500Hz 时的输出波形

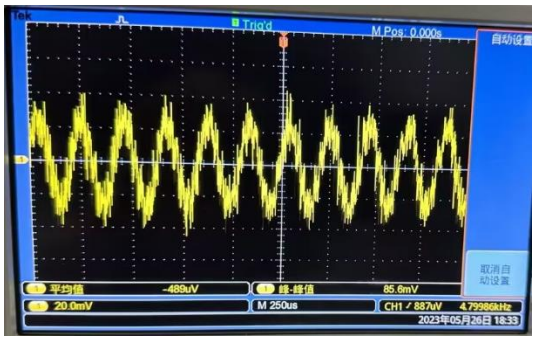


图 27：4800Hz 时的输出波形

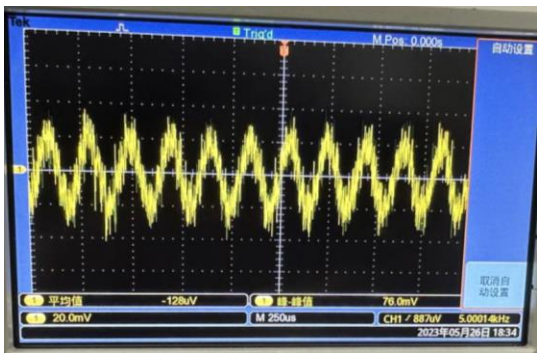


图 28：5000Hz 时的输出波形

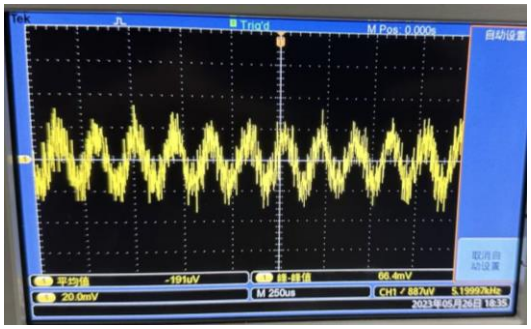


图 29：5200Hz 时的输出波形

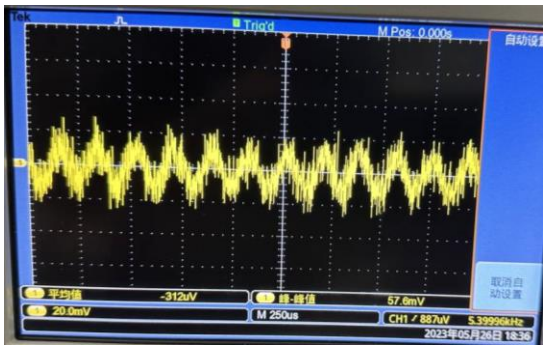


图 30：5400Hz 时的输出波形

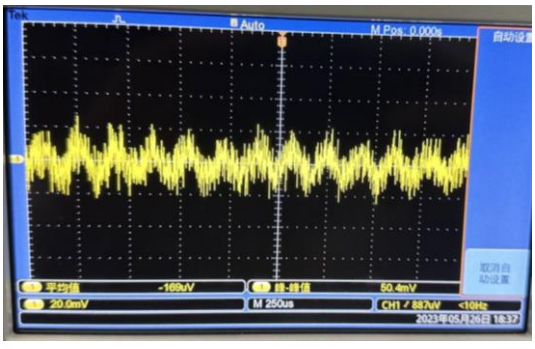


图 31：5600Hz 时的输出波形

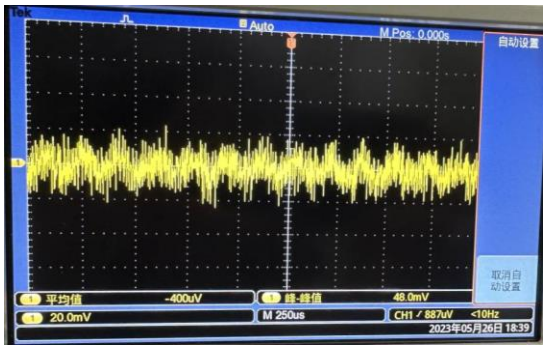


图 32：5800Hz 时的输出波形

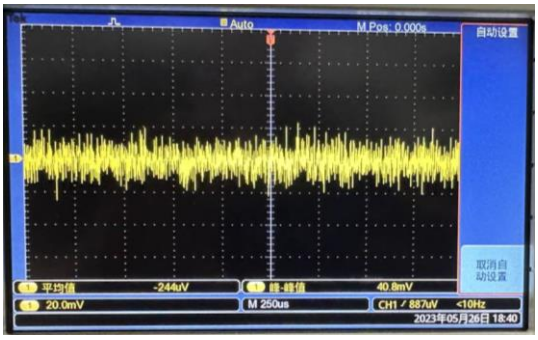


图 33：6000Hz 时的输出波形

12.6 实际幅频特性曲线

改变输入正弦信号的频率，记录下对应的峰值幅度，描点作图并与理论幅频特性曲线进行比较。具体波形见 12.5；经过整理，得到实际的幅频特性情况记录表，如**表 3** 所示：

表 3：实际幅频特性数值记录表

频率（kHz）	幅度（mV）	频率（kHz）	幅度（mV）	频率（kHz）	幅度（mV）
1	43.2	2.5	102	4.9	80.8
1.3	50.4	2.6	105	5	76
1.5	56	2.7	108	5.1	72
1.6	61.6	2.8	110	5.2	66.4
1.7	63.2	3.3	114	5.3	60.8
1.8	68	3.8	117	5.4	57.6
1.9	73.6	4.3	107	5.5	52
2	76	4.4	104	5.6	50.4
2.1	81.6	4.5	97.6	5.8	48
2.2	86.4	4.6	93.6	6	40.8
2.3	89.6	4.7	89.6		
2.4	96	4.8	85.6		

根据表格作图，得到实际的幅频特性曲线，如**图 34** 所示：

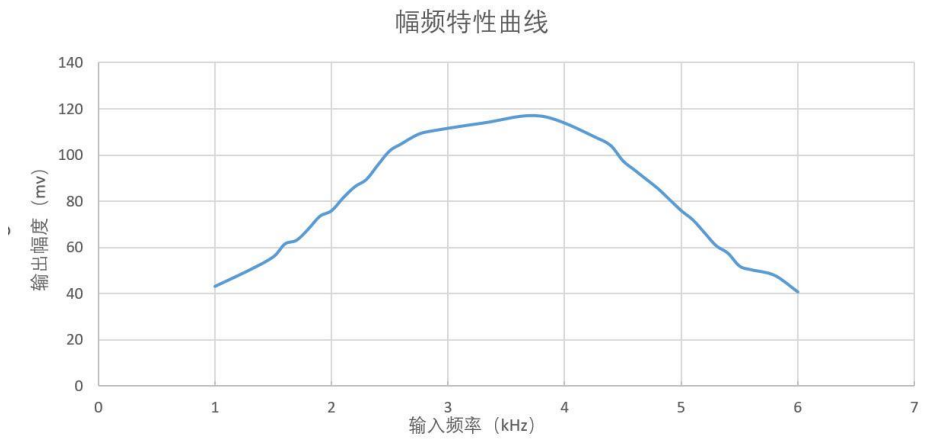


图 34：实际幅频特性曲线

可以看出在误差允许范围内，实际幅频特性曲线较为良好地贴合理论值，即表明 FIR 带通滤波器的设计较为合理。

12.7 算法实时性验证

仿照 Lab_11 实验思考的设计，修改中断部分代码，在进入中断时给 DA 高电平，中断结束后给 DA 低电平。具体代码如**图 35** 所示：

```
interrupt void epwm1_timer_adc_isr(void)    //中断函数
{
    //读取ADC数据并更新xn数组
    *Da_out=a;
    x= (AdcRegs.ADCRESULT1 & 0xFFF0);
    int k;
    for (ConvCount=0;ConvCount<48;ConvCount++){
    xn [ConvCount]=xn [ConvCount+1];
    }
    xn [48]=x;
    //滤波
    for (y=0,k=0 ;k<49;k++)
        y=y+xn[48-k]*h[k];

    //*Da_out=(y>>18)+0x8000;

// Reinitialize for the next ADC Sequence
// Reset SEQ1
   AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1;       //复位SEQ1
// Clear SOC_SEQ1
   AdcRegs.ADCTRL2.bit.SOC_SEQ1 = 0;
///Clear INT SEQ1 bit
// EPwm1Regs.ETCLR.bit.INT = 1;           //清除中断标志位

// Clear ETFLG SOCA
//   EPwm1Regs.ETFLG.bit.SOCA = 1;
  // 清除SEQ1中断标志位
   AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
// Acknowledge interrupt to PIE
   PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //PIEACK-PIE ackonwledge register   //中断应答

    //*CPLD_CNTR = 0;
    //*CPLD_LED = 0;
   *Da_out=0;
   return;
}
```

图 35：验证算法实时性代码

在示波器上显示方波状，可知本算法具有实时性；利用示波器自带光标测量采样周期和算法工作计算时间，如**图 36、图 37** 所示：

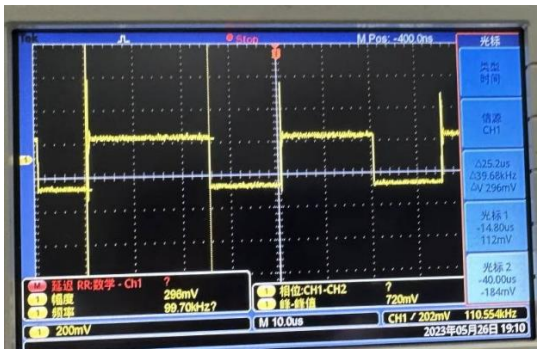


图 36：测量采样周期

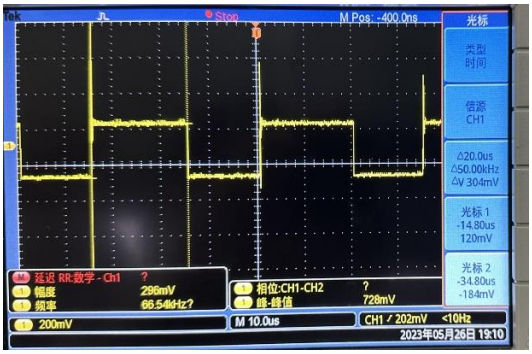


图 37：测量中断工作时间

由图可知，FIR 滤波器工作时间为 20.0us 采样周期为 25.2us，即 DSP 芯片设置的采样频率，算法工作时间小于采样时间，符合实时性要求。

12.8 实验思考和问题分析

1.指出 FIR 滤波器系数的设计参数指标

在 MATLAB 中，fir1 函数用于设计 FIR（有限脉冲响应）数字滤波器。它的使用方法如下：

h = fir1(N, Wn, filterType, windowType)

参数的含义如下：

N：滤波器的阶数（长度减 1）。较高的阶数可以提供更陡的滤波特性，但会增加计算复杂度。

Wn：归一化的截止频率或频率范围。对于低通或高通滤波器，它是一个标量，表示截止频率。对于带通或带阻滤波器，它是一个长度为 2 的向量，表示频率范围。截止频率或频率范围的取值范围是 0 到 1，其中 1 对应于采样率的一半。

filterType：滤波器类型。可以选择的滤波器类型包括：'low'（低通）、'high'（高通）、'bandpass'（带通）和'stop'（带阻）。

windowType（可选）：窗函数类型。常见的窗函数类型包括：'rect'（无窗函数）、'hamming'（Hamming 窗）、'hanning'（Hanning 窗）和'blackman'（Blackman 窗）等。

函数的输出是一个 FIR 滤波器的系数向量 h，可以将其应用于输入信号使用 filter 函数进行滤波。

在本实验中，我们设计了一个 48 阶、数字频率范围为 0.1π 到 0.25π 的带通滤波器，没有加窗（或是加矩形窗）。

2.观察各种输入信号通过数字滤波器系统之后的输出波形，解释信号失真的原因。

信号失真的原因主要有两个方面。

首先，由于 DSP 的输出字长限制，当对输入信号进行量化时，存在着量化误差。DSP 系统将模拟信号转换为数字信号时，需要将连续的模拟信号离散化表示，但是输出位数有限，无法完全表达输入信号的所有细节。因此，输出信号的位数小于滤波器输出的理论值，这导致了截取误差的产生。截取误差会引入额外的噪声成分，使得输出信号与原始信号存在差异。

另外，信号通过数字滤波器时，会受到滤波器的特性影响。例如，当信号通过低通滤波器时，滤波器会抑制通带范围外的高频信号成分，只允许通过通带范围内的低频信号成分。如果输入信号包含了通带范围外的频率分量，这些频率分量会被滤波器抑制，从而引起输出波形的失真。这种失真通常表现为波形的形状变化、频率成分的丢失或畸变。

3.由汇编语言编写的 FIR 滤波器程序，其运算时间高于 C 语言，分析 C 语言效率低的原因。

汇编语言相比 C 语言具有更高的效率，主要是因为它直接操作底层硬件资源，更加贴近计算机体系结构的操作。影响 C 语言效率底下的原因可能有以下几个：

编译器优化限制：C 语言的编译器在将代码转换为机器语言时进行了一系列优化，但优化会导致一些性能上的损失。与此相比，汇编语言代码是直接由程序员编写的，可以针对特定的硬件架构进行高度优化。

内存访问效率：C 语言在处理 FIR 滤波器时会使用数组存储滤波器系数和输入数据。但 C 语言的数组访问是通过指针和偏移量进行的，这涉及到多级寻址和间接寻址的操作，会引入额外的开销。汇编语言可以更直接地访问内存和寄存器，减少了寻址和间接寻址的开销。

控制流开销：C 语言的控制流结构通常需要使用分支指令来实现，这会引入

额外的分支预测和分支跳转开销。

4. DSP 芯片存储器字长转换关系验证

在程序中，涉及到 16 位有符号、无符号数相乘等运算，涉及到字长的变化，需要先对 DSP 芯片中字长变化的关系进行验证，主要分 3 部分：

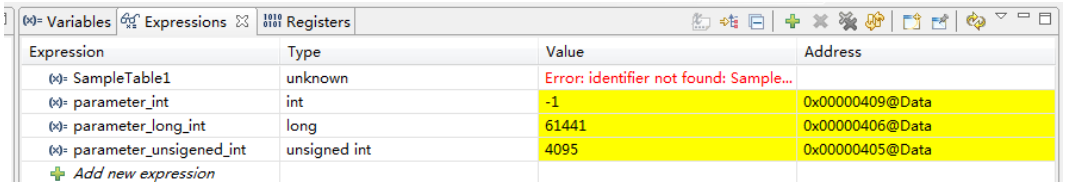
（1）验证有符号长整型变量能否直接接受有符号数与无符号数相乘的结果

在主程序中编写验证代码如下：

```
int parameter_int=0xFFFF;
long int parameter_long_int=0;
unsigned int parameter_unsigened_int=0x0FFF;

parameter_long_int=parameter_int*parameter_unsigened_int;
```

运行后，查看变量结果如图所示。则可知，这种运算方式与期望值（-4095）不一致，不可以使用这种方式来进行字长转换。



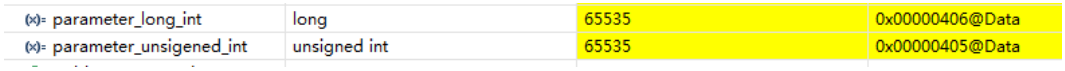
（2） 验证无符号整型变量能否直接赋值给有符号长整型变量

在主程序中编写验证代码如下：

```
int parameter_int=0xFFFF;
long int parameter_long_int=0;
unsigned int parameter_unsigened_int=0x0FFF;

parameter_long_int=parameter_unsigened_int;
```

运行后，查看变量结果如图所示。则可知，无符号整型变量直接赋值给有符号长整型变量后，值不变，可以利用这种方式实现字长转换。



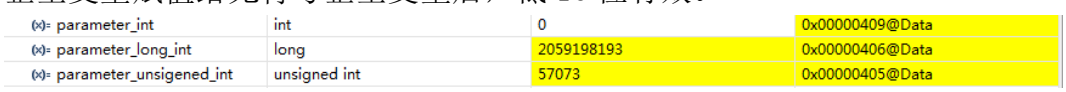
（3）验证有符号长整型变量赋值给无符号整型变量的结果

在主程序中编写验证代码如下：

```
int parameter_int=0;
long int parameter_long_int=0x7ABCDEF1;
unsigned int parameter_unsigened_int=0;

parameter_unsigened_int= parameter_long_int;
```

结果如图所示。parameter_unsigened_int=0xEDF1。则可知，有符号长整型变量赋值给无符号整型变量后，低 16 位有效。



5. 以该 FIR 滤波器系统为例，总结分析系统实时性的取决因素。

按照软硬件来区分，可以将实时性取决因素划分为运算量和芯片速度两个方面，而运算量又可以分为数据率和算法复杂程度。实时性的要求是指在下一个数据到达之前，前一个数据的处理必须完成。根据验证算法实时性的实验，采样频率越高，矩形波占空比越大，系统实时性越难以满足。另外，通过改变滤波器阶数 N 发现，N 越大，即算法复杂度越高，实时性越难满足。

6. 观察各种输入信号通过数字滤波器的波形

由于本滤波器是带通滤波器，因此 2.5kHz 的方波、三角波等波形在通过本滤波器后，会出现不同程度的失真，形状逐渐向基频的正弦波靠拢。这是因为这些波是由各种不同频率的正弦波叠加而成，因此在滤波后，只剩下了基频或基频和二次谐波部分，高次谐波部分被滤除。

7. 如何用该系统实现实时 FFT

可以调用 TI 提供相关 dsp 算法库中的 fft 库，如下图所示：

```
// 声明并初始化结构体对象
RFFT32 rfft = RFFT32_512P_DEFAULTS;

RFFT32_brev(ipcbsrc, ipcb, FFT_SIZE); // 实数FFT位反转

rfft.ipcbptr = ipcb;                  // FFT计算缓冲区
rfft.magptr  = ipcbsrc;               // 幅度输出缓冲区
rfft.winptr  = (long *)win;           // 窗口系数数组
rfft.init(&rfft);                     // 初始化旋转因子指针

rfft.calc(&rfft);                     // 计算FFT
rfft.split(&rfft);                    // 后处理以获得正确的频谱
rfft.mag(&rfft);                      // Q31格式 (abs(ipcbsrc)/2^16).^2
```

这段程序调用了“fft.h”，实现了一个基于 512 点 Q31 定点数 FFT 的频谱分析功能。首先调用函数 RFFT32_brev() 对输入缓冲区 ipcbsrc 进行实数 FFT 的位反转操作，将数据重新排列为正确的顺序。然后将结构体对象 rfft 的成员变量指向相应的缓冲区和数组：rfft.ipcbptr 指向输入缓冲区 ipcb、rfft.magptr 指向幅度输出缓冲区 ipcbsrc、rfft.winptr 指向窗口系数数组 win。之后调用 rfft.init(&rfft) 来初始化结构体对象，主要是对旋转因子指针进行初始化。最后调用 rfft.calc(&rfft) 来计算 FFT,rfft.split(&rfft) 进行后处理操作，以获取正确的频谱， rfft.mag(&rfft) 计算幅度，结果以

Q31 格式表示，即取绝对值并除以 2^16 后再平方。

我们可以将这段程序添加进已有的数据采集和输出程序，对不同频率成分的信号进行 AD 采集和转换，经过 FFT 处理，最后用 DA 输出信号频谱图。

实验过程中我们遇到了以下问题：

1.示波器输出波形显示波形较粗并且抖动剧烈。

经过进一步分析，我们发现问题出在系数定标上。我们的定标方法存在一定的问题，导致滤波器的系数范围没有正确地映射到合适的数值范围上。经过修正后的定标方法应该是在系数表中找出具有最大值的 h 系数，其值为 0.15，并将其映射到 16 位有符号数的最大值。换言之，滤波器的系数之间的差异倍数并不大，这就导致了滤波效果的不佳。

2.采样率过高，算法时间余裕不大

我们使用 40kHz 作为采样频率，采样周期就是 25us，通过算法实时性的验证实验可以看出，算法时间为 20us，如果还想对数据进行一些其他处理的话，现有采样周期的余裕不大，就要被迫降低采样率。

12.9 实验总结

本次实验由于没有示例程序，因此需要自己对实验三的程序进行修改，以实现功能要求。因此难度相比前面的实验高了很多。回顾已做的实验，实际上都在为最后的实验奠定基础，确保电脑可以正确连接 dsp 仿真器并在线调试、DA 模块输出正常、AD 模块采样正常，这些都是滤波器实验高楼大厦的一砖一瓦，让我深刻体会到模块化工程调试的思想。

本次实验有了需要借助外部软件自行设计的部分，不同的滤波器参数得到的结果也会有所不同。这就要求我们准确掌握 dsp 相关原理知识，这样既可保证后续程序的可靠性，同时也可以在一定程度上降低验证结果的复杂度。

在验证仿真滤波器功能的过程中，如何编写实现代码成为了困扰我们多时的问题。因牵扯到数据移位和各种数制之间的转换，还要检查数组越界与否，我们前后多次调整程序，不断对猜想进行调试验证，尽量高效地修改程序，最终得到了理想的输出结果。但最后滤波输出波形有较大噪声，我们在现场没有深究滤波

器系数定标的问题，而是选择先拍照记录，错失了发现问题之后再次调试的机会。

最后，感谢同组成员的相互配合。感谢助教和老师的指导！