



南京理工大学

NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

# DSP 应用技术实验

## FIR 滤波器实现实验报告

作 者 : 许晓明 学 号 : 9161040G0734

同 组 人 : 李玥 学 号 : 9161040G0703

同 组 人 : 陈锦涛 学 号 : 9161040G0614

学 院 : 电子工程与光电技术学院

专 业 : 电子信息工程

班 级 : 电信 3 班

组 号 : 第二组 B4

题 目 : DSP 应用技术实验

FIR 滤波器实现实验报告

指 导 者 : 李彧晟

2019 年 11 月

# 目录

1 实验目的.....	1
2 实验仪器.....	1
2.1 实验仪器清单.....	1
2.2 硬件连接示意图.....	1
3 实验步骤及现象 .....	1
3.1 算法实现流程图.....	1
3.2 程序流程图.....	2
3.3 系统设计.....	3
3.3.1 利用设计滤波器系数.....	3
3.3.2 数据定标.....	4
3.4 设备检查并启动集成开发环境.....	5
3.5 编写 FIR 算法模块 .....	5
3.5.1 FIR 滤波器编写原理 .....	5
3.5.2 DSP 芯片存储器字长转换关系验证 .....	5
3.5.3 输入信号的更新.....	6
3.5.4 FIR 滤波器代码 .....	6
3.6 建立工程并运行、调试程序.....	7
3.7 算法实时性验证.....	7
4 实验结果及思考题回答 .....	7
4.1 设计 FIR 滤波器并仿真 .....	7
4.2 算法功能验证.....	9
4.3 实际幅频特性曲线.....	12
4.4 验证算法实时性.....	13
5 实验总结.....	15
5.1 实验中遇到的问题及解决方法.....	15
5.2 实验心得体会.....	15

## 1 实验目的

1. 巩固数字 FIR 滤波器的概念；
2. 了解 DSP 运算特点；
3. 理解算法实时性含义；
4. 熟练掌握 DSP 软件开发过程；
5. 熟练掌握 DSP 软件调试方法。

## 2 实验仪器

### 2.1 实验仪器清单

- |                              |    |
|------------------------------|----|
| 1. DSP 仿真平台（仿真器、DSP 实验箱、计算机） | 一套 |
| 2. 示波器                       | 一台 |
| 3. 信号发生器                     | 一台 |

### 2.2 硬件连接示意图

实验硬件连接大致如图 2.1 所示。

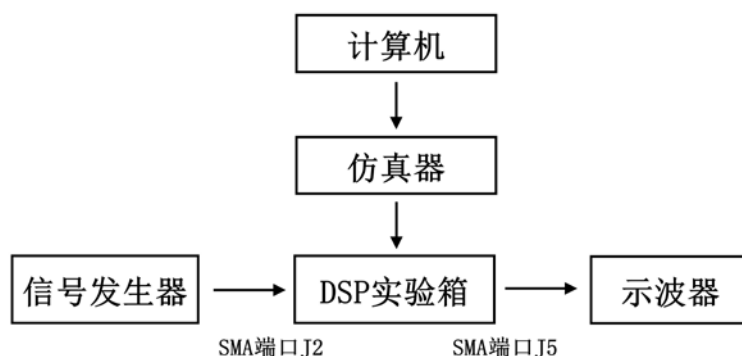


图 2.1 硬件连接示意图

## 3 实验步骤及现象

### 3.1 算法实现流程图

结合实验要求，实验之前首先必须对 FIR 滤波器的设计、实现算法有所了解，必要时通过计算机算法仿真，理解 FIR 滤波器特性。

根据 FIR 滤波器算法，编写 C 源程序，实现算法功能。并验证 DSP 实现时算法的正确性以及精度的要求。这种算法功能上的仿真可以利用 CCS 集成开发环境中数据 IO 来模拟信号的输入，完成验证算法精度与功能的正确。

验证了算法的功能正确之后，可以将程序下载到 DSP 上运行，观察现象。更为重要的是在硬件平台上验证系统的实时性，以及评估资源的使用情况。若满足实时性要求，则测试各项指标，应该与原理设计相吻合。如果实现与理论不一致，则首先检查算法的实时性，以及资源使用是否冲突等原因，对程序进行用是否冲突等原因，对程序进行优化后再次编译链接，重新验证直至正确。算法的优化有时会贯穿于整个设计之中。于是，程序流程图如图 3.1 所示。

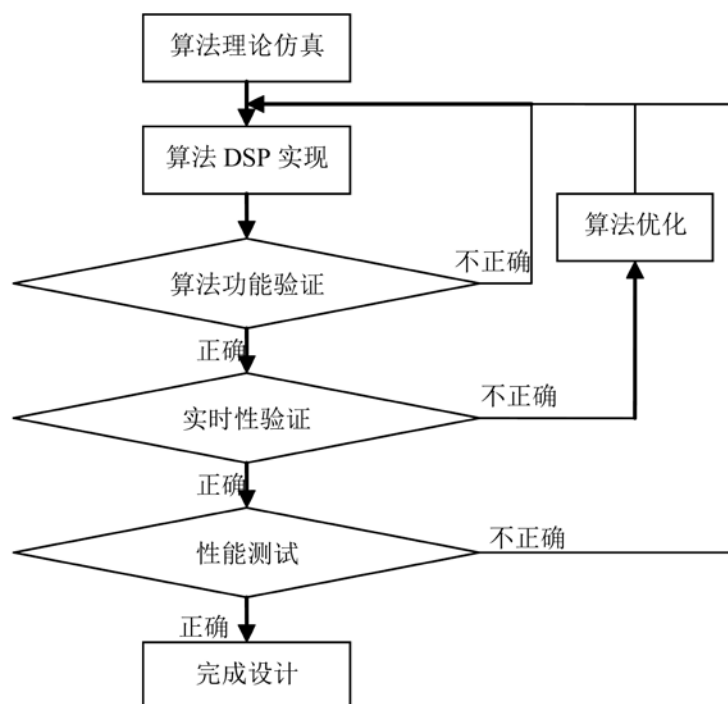


图 3.1 算法实现流程图

### 3.2 程序流程图

FIR 滤波器处理数据时，每到达一个新数据，就必须进行一次计算，更新输出。因此，和 DSP 的数据采集实验类似，用 DSP 实现实时的 FIR 信号处理算法必须依赖于 ADC、DSP 以及 DAC 三大基本部件。充分利用 DSP 片上 ADC 外设，实现模拟信号的采样，并由 DSP 完成 FIR 核心算法，由实验箱中 DAC(AD9747)来完成数字到模拟的还原。在数据采集实验基础上，我们对程序流程稍加改动，就可实现完整数字 FIR 滤波器功能。程序流程如图 3.2 所示。

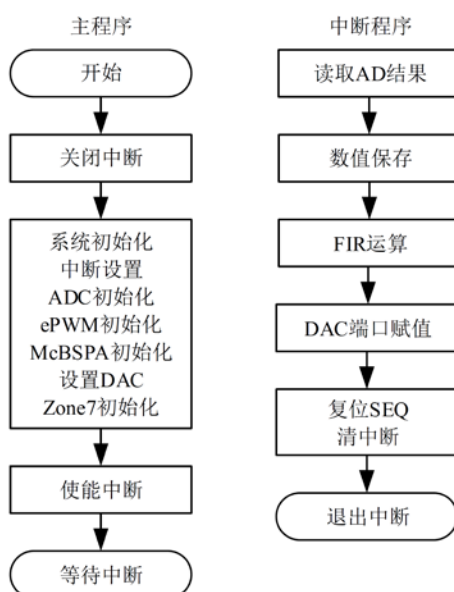


图 3.2 程序流程图

### 3.3 系统设计

#### 3.3.1 利用设计滤波器系数

利用 MATLAB 设计滤波器系数，设计的滤波器为 48 阶，其幅频、相频特性曲线如图 3.3 所示，系数矩阵如表 3.1 所示。

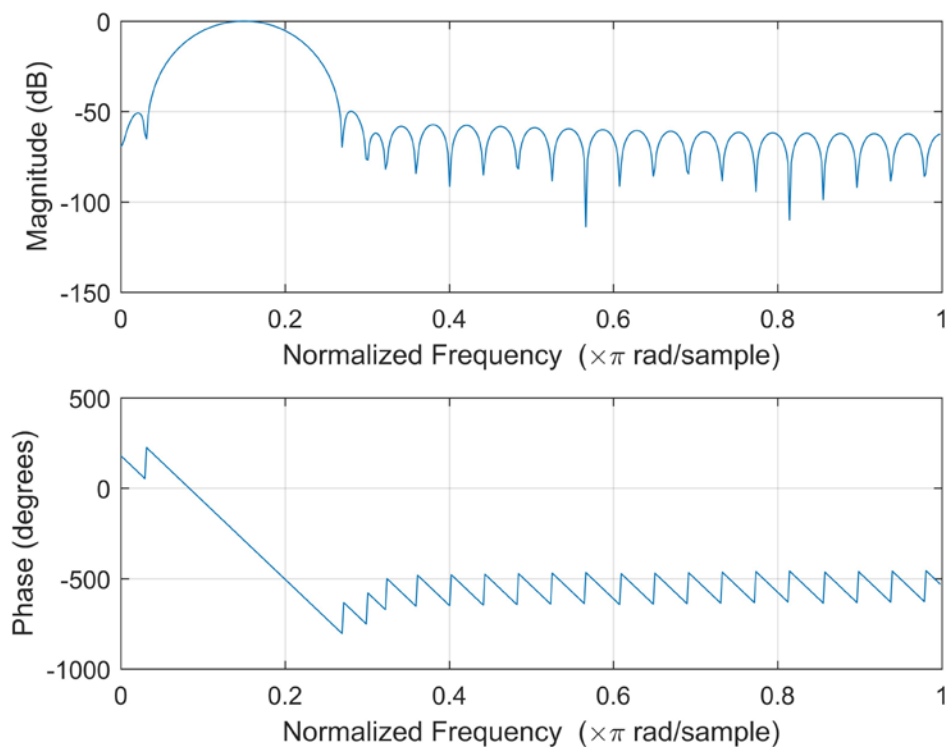


图 3.3 MATLAB 设计的滤波器的幅频相频特性曲线

表 3.1 MATLAB 设计的滤波器的 h 系数

h	值	h	值	h	值	h	值
0	-0.00042	13	0.016914	26	0.061816	39	0.008389
1	0.000179	14	0	27	0.015797	40	0.002434
2	0.000546	15	-0.02467	28	-0.02946	41	-0.00075
3	0.000528	16	-0.05121	29	-0.06258	42	-0.0015
4	0	17	-0.07129	30	-0.07673	43	-0.00089
5	-0.00089	18	-0.07673	31	-0.07129	44	0
6	-0.0015	19	-0.06258	32	-0.05121	45	0.000528
7	-0.00075	20	-0.02946	33	-0.02467	46	0.000546
8	0.002434	21	0.015797	34	0	47	0.000179
9	0.008389	22	0.061816	35	0.016914	48	-0.00042
10	0.015997	23	0.096004	36	0.023942		
11	0.022467	24	0.10862	37	0.022467		
12	0.023942	25	0.096004	38	0.015997		

### 3.3.2 数据定标

$h$  系数中最大的值为 0.108619773528739，将它映射到 16 位有符号数的最大值 32767，即乘 301666 并取整，其余的值也做此操作，得到表 3. 2 的系数，完成浮点到定点的转换。作出定标后系数的幅频相频特性曲线如图 3. 4 所示，发现满足设计要求。

表 3. 2 数据定标后的滤波器系数

h	值	h	值	h	值	h	值
0	-126	13	5102	26	18648	39	2531
1	54	14	0	27	4765	40	734
2	165	15	-7442	28	-8889	41	-227
3	159	16	-15448	29	-18877	42	-452
4	0	17	-21505	30	-23146	43	-268
5	-268	18	-23146	31	-21505	44	0
6	-452	19	-18877	32	-15448	45	159
7	-227	20	-8889	33	-7442	46	165
8	734	21	4765	34	0	47	54
9	2531	22	18648	35	5102	48	-126
10	4826	23	28961	36	7223		
11	6778	24	32767	37	6778		
12	7223	25	28961	38	4826		

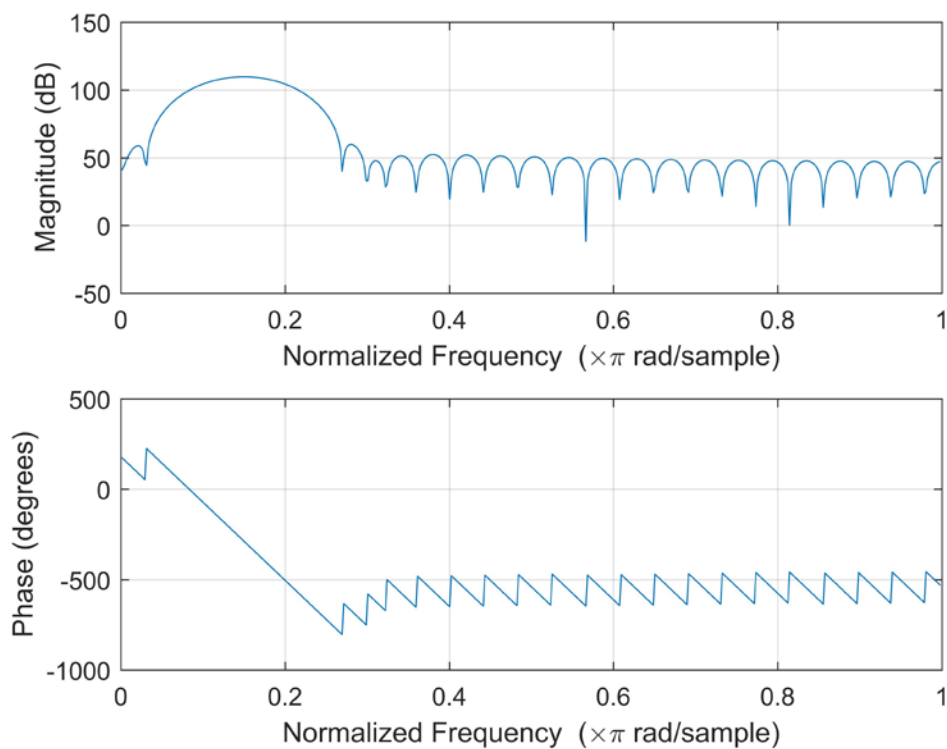


图 3. 4 数据定标后的滤波器的幅频相频特性曲线

### 3.4 设备检查并启动集成开发环境

检查仿真器、C2000 DSP 实验箱、计算机之间的连接是否正确，确认无误后，打开计算机和实验箱电源，并进入集成开发环境 CCS。

### 3.5 编写 FIR 算法模块

#### 3.5.1 FIR 滤波器编写原理

有限长的单位冲击响应滤波器（FIR）差分方程可表示为：

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

其中， $h$ 是滤波器系数， $x$ 为输入的数字信号， $y$ 为 FIR 滤波器计算输出。 $N$ 为滤波器阶数。由此可得，一个  $N$  阶的滤波器计算，需要  $N+1$  个滤波器系数， $N+1$  个数字输入，每得到一个  $y$  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，以及事先设计的  $N+1$  个滤波器系数。

#### 3.5.2 DSP 芯片存储器字长转换关系验证

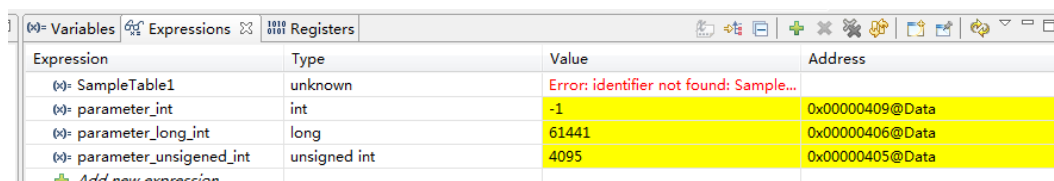
在程序中，涉及到 16 位有符号、无符号数相乘等运算，涉及到字长的变化，需要先对 DSP 芯片中字长变化的关系进行验证，主要分 3 部分：

##### 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：

```
69
70     int parameter_int=0xFFFF;
71     long int parameter_long_int=0;
72     unsigned int parameter_unsigned_int=0x0FFF;
73
74     parameter_long_int=parameter_int*parameter_unsigned_int;
```

运行后，查看变量结果如图 3.5 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。



Expression	Type	Value	Address
SampleTable1	unknown	Error: identifier not found: Sample...	
parameter_int	int	-1	0x00000409@Data
parameter_long_int	long	61441	0x00000406@Data
parameter_unsigned_int	unsigned int	4095	0x00000405@Data

图 3.5 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

##### 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：

```
70     int parameter_int=0xFFFF;
71     long int parameter_long_int=0;
72     unsigned int parameter_unsigned_int=0x0FFF;
73
74     parameter_long_int=parameter_unsigned_int;
```

运行后，查看变量结果如图 3.6 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。

(x) parameter_long_int	long	65535	0x00000406@Data
(x) parameter_unsigned_int	unsigned int	65535	0x00000405@Data

图 3.6 验证无符号整型变量能否直接赋值给有符号长整型变量

### 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：

```

70     int parameter_int=0;
71     long int parameter_long_int=0x7ABCDEF1;
72     unsigned int parameter_unsigned_int=0;
73
74     parameter_unsigned_int= parameter_long_int;

```

运行后，查看变量结果如图 3.7 所示。parameter\_unsigned\_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。

(x) parameter_int	int	0	0x00000409@Data
(x) parameter_long_int	long	2059198193	0x00000406@Data
(x) parameter_unsigned_int	unsigned int	57073	0x00000405@Data

图 3.7 验证有符号长整型变量赋值给无符号整型变量的结果

#### 3.5.3 输入信号的更新

为实现滤波器输出的实时更新，每一个采样信号到来时，需要得知这个信号之前的 48 个信号，共 49 个信号。因此需要对输入信号进行前移，并将最新采样的数据存到最后，使得 49 个数据始终为按时间顺序排列的最新数据，实现的代码如下：

```

366     x= (AdcRegs.ADCRESULT1 & 0xFFF0);
367     int k;
368     for(ConvCount=0;ConvCount<48;ConvCount++)
369     {
370         xn[ConvCount]=xn[ConvCount+1];
371     }
372     xn[48]=x;

```

通过这段代码，前 48 个数据循环前移，并将最新的数据放入数组的最后一个位置。

#### 3.5.4 FIR 滤波器代码

综合以上内容，FIR 滤波器代码如下：



```

366 x= (AdcRegs.ADCRESULT1 & 0xFFF0);
367 int k;
368 for(ConvCount=0;ConvCount<48;ConvCount++)
369 {
370     xn[ConvCount]=xn[ConvCount+1];
371 }
372 xn[48]=x;
373
374 for(k=0;k<49;k++)
375 {
376     y=y+xn[48-k]*h[k];
377 }
378 y1=y;
379 /* y=0;
380 if(k<1024)
381 { y_result[k]=y1;
382 k++;
383 }
384 else k=0;*/
385 *Da_out=(y1>>18)+0x8000;

```

### 3.6 建立工程并运行、调试程序

连接信号发生器至教学实验箱 SMA 输入端口 J2、教学实验箱 SMA 输出端口 J5 至示波器，编译链接工程并进入调试调试界面，运行程序后，查看输出是否正确，验证算法正确性。

### 3.7 算法实时性验证

测量采样频率与 FIR 算法的核心执行时间，判断该系统是否实时。若非实时，则优化程序甚至修改算法，直至满足实时要求。

## 4 实验结果及思考题回答

### 4.1 设计 FIR 滤波器并仿真

完成 FIR 滤波器系数的设计并仿真。

设计的 FIR 滤波器系数见表 3.2，幅频相频特性见图 3.4。

利用 MATLAB 对设计的 FIR 滤波器进行仿真验证，得到的仿真结果，如图 4.1 到图 4.2 所示，说明可以衰减 1kHz 到 2kHz 之外的信号。

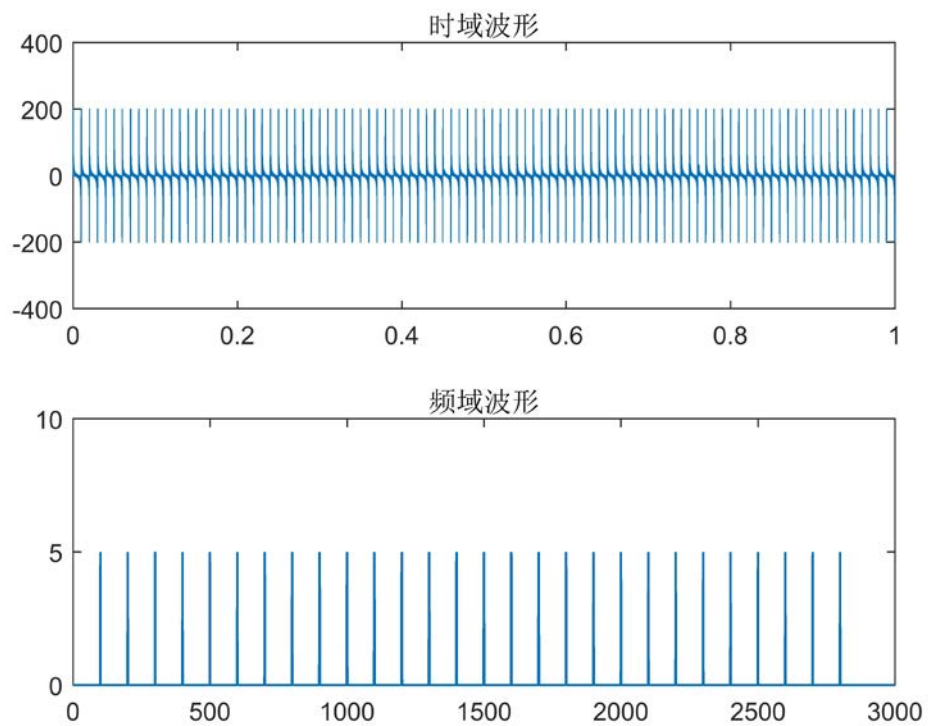


图 4.1 通过 fir 滤波器前的信号

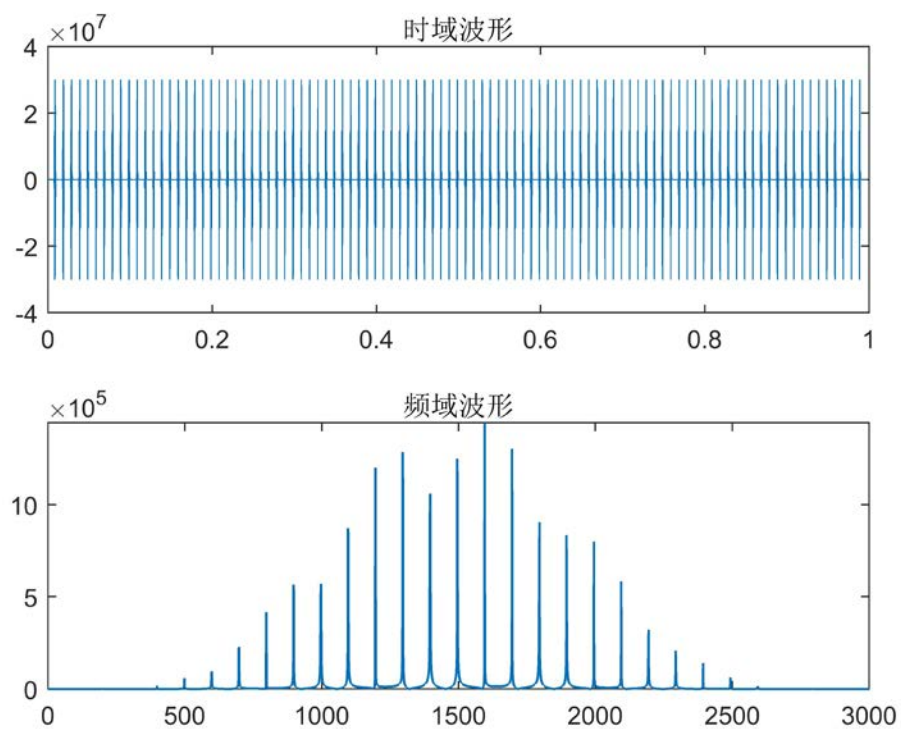


图 4.2 通过 fir 滤波器后的信号

验证系统的实时性，测试采样周期以及计算时间

## 4.2 算法功能验证

调试程序，实现 FIR 功能，利用硬件验证。

通过信号发生器输入不同频率的正弦波，观察示波器上结果，如图 4.3 到图 4.24 所示，可观察到在 1kHz 到 2kHz 之外的信号衰减，而 1kHz 到 2kHz 之内的信号可以通过。

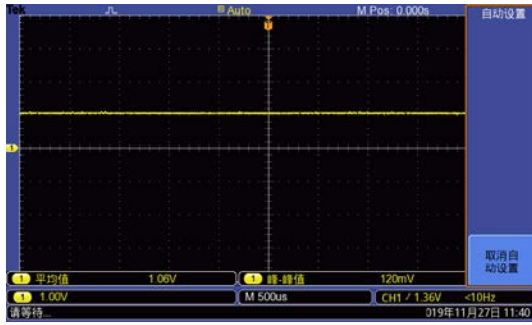


图 4.3 600Hz 时的输出波形

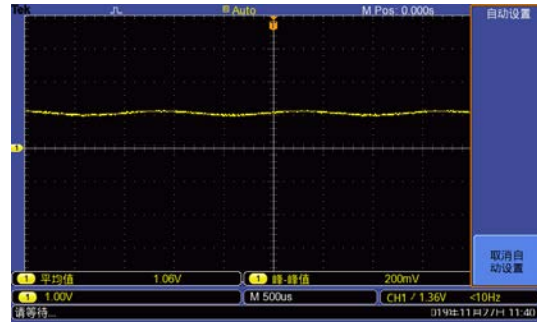


图 4.4 700Hz 时的输出波形

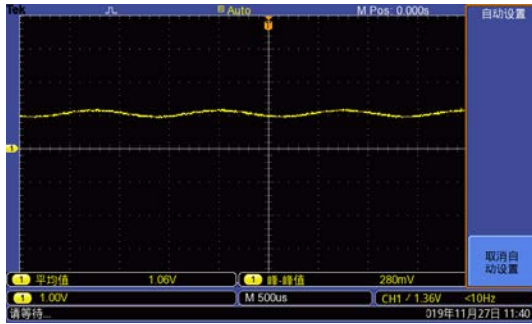


图 4.5 800Hz 时的输出波形

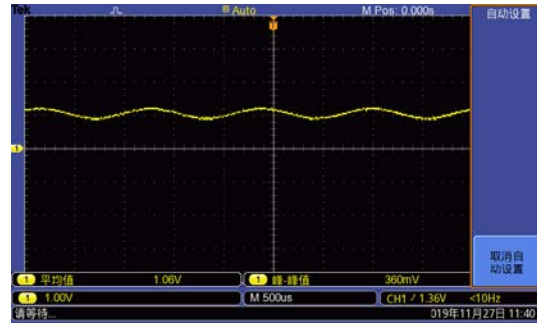


图 4.6 900Hz 时的输出波形

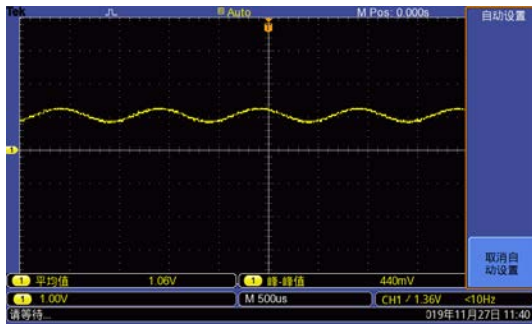


图 4.7 1000Hz 时的输出波形

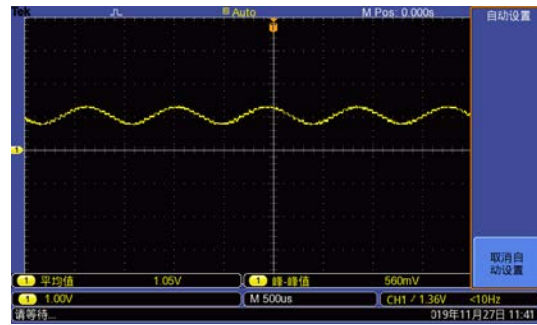


图 4.8 1100Hz 时的输出波形

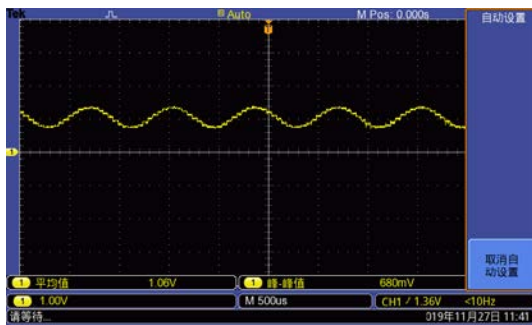


图 4.9 1200Hz 时的输出波形

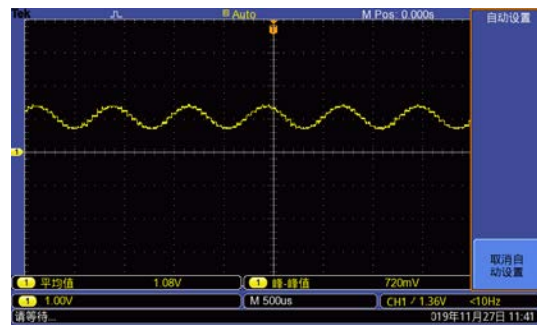


图 4.10 1300Hz 时的输出波形

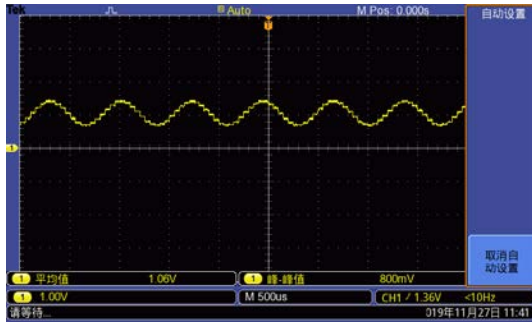


图 4.11 1400Hz 时的输出波形

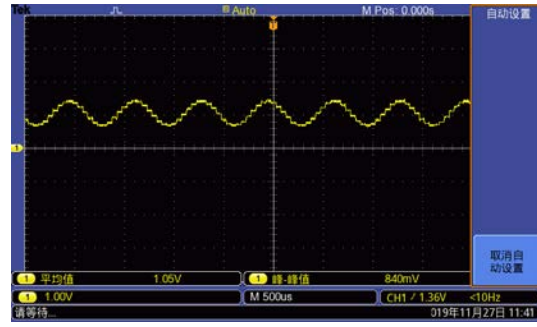


图 4.12 1500Hz 时的输出波形

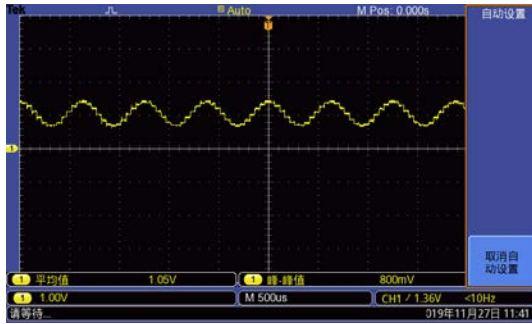


图 4.13 1600Hz 时的输出波形

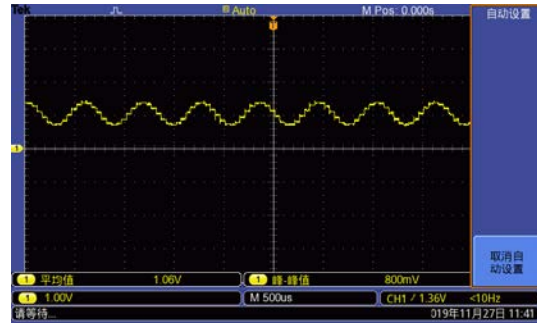


图 4.14 1700Hz 时的输出波形

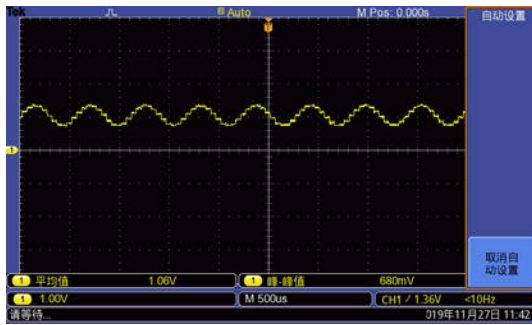


图 4.15 1800Hz 时的输出波形

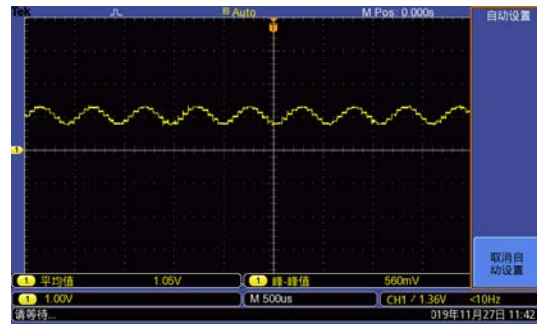


图 4.16 1900Hz 时的输出波形

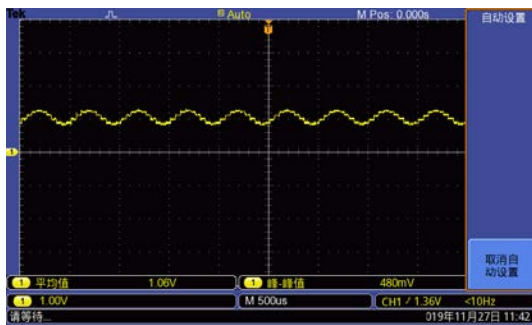


图 4.17 2000Hz 时的输出波形

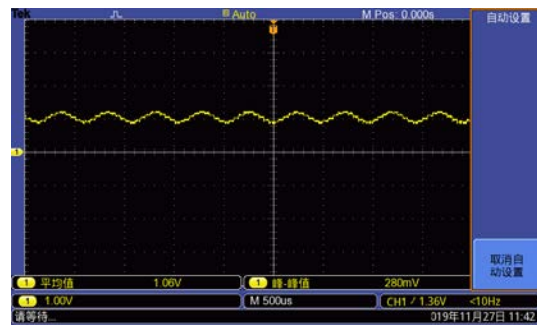


图 4.18 2100Hz 时的输出波形

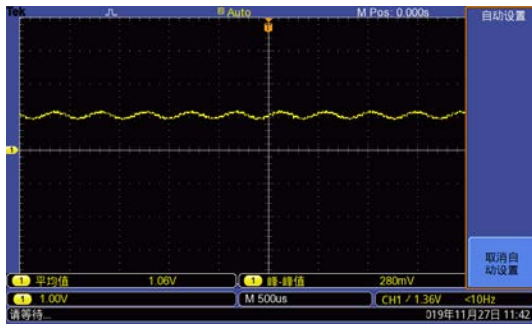


图 4.19 2200Hz 时的输出波形

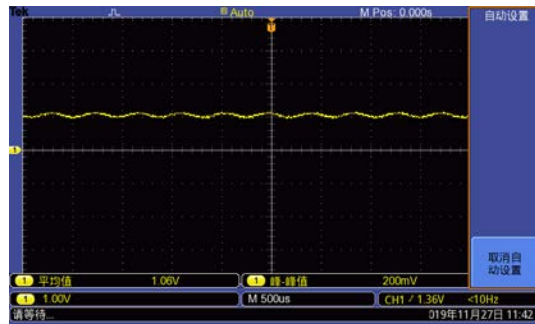


图 4.20 2300Hz 时的输出波形

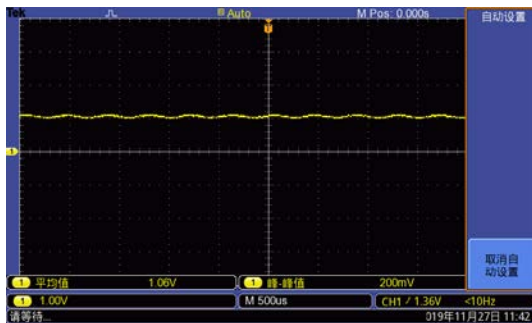


图 4.21 2400Hz 时的输出波形

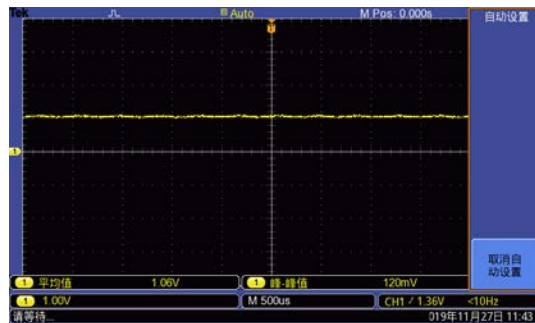


图 4.22 2500Hz 时的输出波形

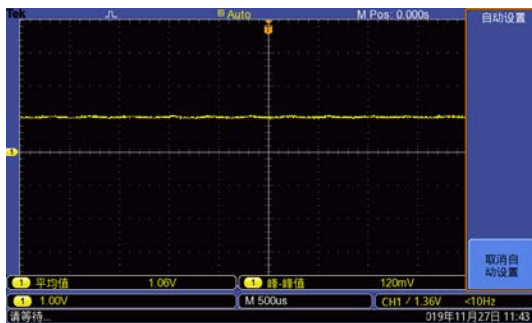


图 4.23 2600Hz 时的输出波形

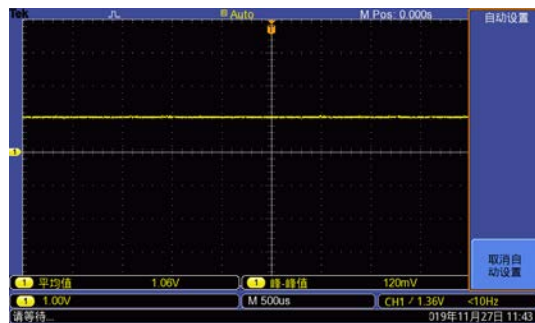


图 4.24 2700Hz 时的输出波形

### 4.3 实际幅频特性曲线

改变输入正弦信号频率，记录对应的幅度，描点作图，与理论幅频曲线比较

如图 4.3 到图 4.24 所示，整理可得到表 4.1，绘图，得到实际的幅频特性曲线如图 4.25 所示。

表 4.1 实际幅频情况记录表

频率(Hz)	幅度(mv)	频率(Hz)	幅度(mv)
600	120	1700	800
700	200	1800	680
800	280	1900	560
900	360	2000	480
1000	440	2100	280
1100	560	2200	280
1200	680	2300	200
1300	720	2400	200

1400	800	2500	120
1500	840	2600	120
1600	800	2700	120

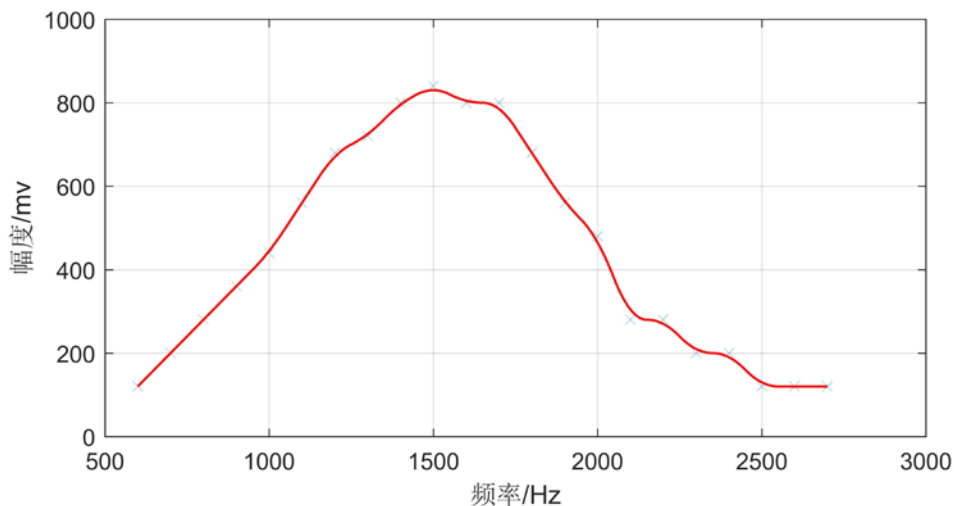


图 4.25 实际幅频特性曲线

#### 4.4 验证算法实时性

修改代码，测量算法运行时间与采样时间的关系，验证算法实时性。

在进入中断时给 DA 高电平，中断结束时给 DA 低电平，代码如下：

```

360 interrupt void epwm1_timer_adc_isr(void) //中断函数
361 {
362     *Da_out=10000;
363
364     //DA
365
366     x= (AdcRegs.ADCRESULT1 & 0xFFF0);
367     int k;
368     for(ConvCount=0;ConvCount<48;ConvCount++)
369     {
370         xn[ConvCount]=xn[ConvCount+1];
371     }
372     xn[48]=x;
373
374     for(k=0;k<49;k++)
375     {
376         y=y+xn[48-k]*h[k];
377     }
378     y1=y;
379     /* y=0;
380     if(k<1024)
381     { y_result[k]=y1;
382        k++;
383     }
384     else k=0;*/
385     /*Da_out=(y1>>18)+0x8000;
386
387
388     // *Da_out=yn;
389     /*Da_out=AdcRegs.ADCRESULT1;
390
391

```



```

388 // *Da_out=yn;
389 // *Da_out=AdcRegs.ADCRESULT1;
390
391
392 // Reinitialize for the next ADC Sequence
393 // Reset SEQ1
394 AdcRegs.ADCCTRL2.bit.RST_SEQ1 = 1; //复位SEQ1
395 // Clear INT SEQ1 bit
396 // EPwm1Regs.ETCLR.bit.INT = 1; //清除中断标志位
397 // 清除SEQ1中断标志位
398 AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;
399 // Acknowledge interrupt to PIE
400 PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //PIEACK-PIE acknowledge register
401
402 *Da_out=0;
403 return;
404 }

```

可得到图 4. 26，可知算法可以实时性。进一步用光标测量各时间，如图 4. 27 到图 4. 28 所示，可知 FIR 运行所需时间为 20 微秒，采样时间为 50 微秒，算法时间小于采样时间且有一定裕余，满足实时性要求。

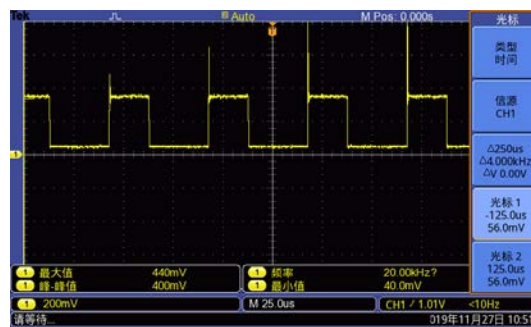


图 4. 26 验证算法实时性

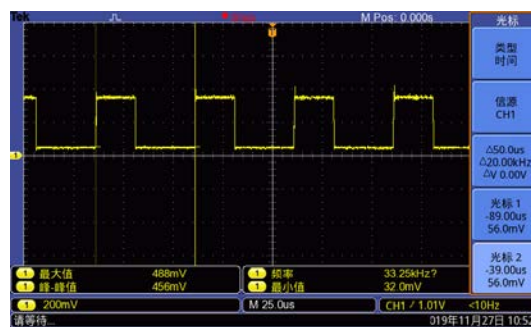


图 4. 27 测算采样时间

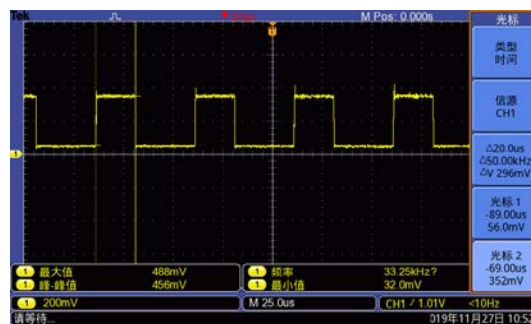


图 4. 28 测算程序运行时间



## 5 实验总结

### 5.1 实验中遇到的问题及解决方法

#### 1. 程序运行后示波器无输出

在第一次编写代码时，运行后发现示波器上无输出。屏蔽各段代码分别调试后发现是采样信号前移部分的代码出错。进一步检查后发现，是没有设置 ConvCount 变量自增，使得循环无法停止造成的，修改代码后，可以正确运行。

#### 2. 当输入在 1kHz 到 2kHz 之间时，示波器上的输出波形不正确

在修改程序代码运行后，发现在某个区间内的示波器输出波形不正确，如图 5.1、图 5.2 所示。经过老师的指点，得知是因为数据溢出，将输出给 DA 的数据多右移 2 位后，波形正常。



图 5.1 错误波形 1



图 5.2 错误波形 2

### 5.2 实验心得体会

这次实验中，没有现成的范例程序，根据需求，在之前的实验代码上独立完成 FIR 滤波器的编写。

在这次实验中，我首次深刻感受到“实际是检验真理的唯一标准”的准确性。在第四次实验课上时，我始终纠结于无符号数与有符号数计算的结果，因为无符号数的最高位不表示正负，最大值是 65535，但有符号数的最高位表示正负，最大值是 32767。由于在滤波器的程序中，对滤波器系数定标后，系数最大值是 32767，又要考虑相乘后给 DA 多少位等等问题。经过思考，在第五次实验课程中，我决定直接在系统上验证，看看字长转换该用什么方式进行。这比“凭空思考”有效率得多，而且能得到正确的结论。

在验证功能时，老师指出我们的滤波器某个区间内溢出。但事实上，我们将相乘的值给长整型变量并右移 16 位后给 DA，理论上应该是所有数据均被接收，仍然出现溢出，说明是 32 位的长整型也不够存储相乘之后的结果，于是我将该存储变量类型修改为 long long int，并右移 18 位，此时在示波器上出现了正确的波形，而波形的幅度较小。

在这次实验中，由于既要验证采样频率、又要验证算法实时性，因此采用了与上次实验不同的验证方案，即在进入中断程序的一开始给 DA 高电平，中断中的程序照常执行、但不赋值给 DA，中断程序的最后一条语句给 DA 低电平。在这种方案下，产生的方波周期就是采样频率的周期；而通过查看高电平的持续时间是否足够短，可以验证系统实时性。