



# 信息与软件工程学院

## 综合设计 II 中期报告

课程名称：\_\_\_\_\_ 综合设计 II

课题名称：\_\_\_\_\_ 输入正弦信号频率和相位估计

指导教师：\_\_\_\_\_ 邓建华

所在系别：\_\_\_\_\_ 软件工程(信息工程)

执行学期：\_\_\_\_\_ 大三上

学生信息：

序号	学号	姓名
1（组长）	2015220104001	刘恒利
2	2015220104004	何劲帆
3	2015220104010	胡震
4	2015220104014	唐浩
5	2015220104024	楚蕃源
6	2015220104027	滑文博



## 目 录

第一章 综合设计的进展情况 .....	1
1.1 针对工程问题的方案设计 .....	1
1.2 针对工程问题的推理分析 .....	3
1.3 针对工程问题的具体实现 .....	9
1.4 知识技能学习情况 .....	13
第二章 存在问题与解决方案 .....	15
2.1 存在的主要问题 .....	15
2.2 解决方案 .....	17
第三章 前期任务完成度与后续实施计划 .....	19
参考文献 .....	21



## 第一章 综合设计的进展情况

### 1.1 针对工程问题的方案设计

在 DM6437EVM 开发板进行对正弦信号进行捕获并进行参数估计，主要分为两个步骤。第一是要将音频输入口的输入信号进行采集，第二是要将采集得到的采样数据进行处理和分析，进一步得到最终的测量数据。

对于 DSP 的音频信号的采集，大多采用 McBSP 多通道同步缓冲串口来实现，音频信号经过分帧输入到波形缓存中，转换为一串数字信号；对于采样信号的检测和估算，可以通过相位差分法和最小二乘拟合法进行计算。

针对两个步骤的设计，相应地，系统分为两段实现，前一段为正弦信号的波形采样，后一段为采样数据的 FFT 变换和参数估算，中间需要通过整型数值到浮点数值的转换完成复数结构的建立进行连接。

小组使用集成开发环境 CCS5.5 平台用于开发，CCS 支持 TI 的微控制器和嵌入式处理器产品系列，包含一整套用于开发和调试嵌入式应用的工具。包含了用于优化的 C/C++ 编译器、源码编辑器、项目构建环境、调试器、描述器以及多种其他功能。

平台搭建于 CCS5.5 上，能够通过计算效率不太高的软仿真实现算法在开发板上的计算过程，组员开发测试能够不受开发板数量限制。

针对使用目标开发板对数字信号进行处理的目标，小组选择现在 MATLAB 中用简易的代码验证实现相位差分法以及最小二乘法拟合，验证算法的正确性。使其作为理论指导，编写对应 C 代码，然后在 CCS 中进行实际代码的开发。



## 1.2 针对工程问题的推理分析

此部分对于 CCS 平台进行分析，将可能使用到的硬件资源信息进行分析，总结出了以下可用信息，便于后续环境搭建提供支持；同时，对于相位差分法与最小二乘拟合法进行数学分析，总结出了各自的特点以及适用情况。

DM6437 使用 TI TLV320AIC33 立体声编码器(codec)来做音频信号的输入输出。音频模拟信号通过 microphone 或 line 输入，经该编码器采样转换成数字信号送 DSP 处理。

立体声编码器使用两个串行通道通信，一个用于 codec 内部寄存器的配置，另一个发送和接收数字音频信号。控制通道仅在配置 codec 时使用，因此在音频数据传输过程中，该通道处于闲置状态。缺省配置是使用 McBSP 作为双向数据通道，但也可以选择 McASP 来驱动数据通道。

### 1.2.1 McBSP 多通道缓冲串行口：

McBSP 是在标准串行接口的基础之上对功能进行扩展，因此，具有与标准串行接口相同的基本功能。它可以和其他 DSP 器件、编码器等其他串口器件通信。

McBSP 包括一个数据通道和一个控制通道，通过 7 个引脚与外部设备连接。数据发送引脚 DX 负责数据的发送，数据接收引脚 DR 负责数据的接收，发送时钟引脚 CLKX，接收时钟引脚 CLKR，发送帧同步引脚 FSX 和接收帧同步引脚 FSR 提供串行时钟和控制信号。

McBSP 具有以下特点：1. 全双工通信；2. 拥有两级缓冲发送和三级缓冲接收数据寄存器，允许连续数据流传输；3. 串行字长度可选，包括 8、12、16、20、24 和 32 位；

McBSP 可以通过两种方式进行数据传输：

- 1) EDMA 方式：McBSP 发送事件通知 EDMA 通道进行传输，有接收事件 REVT 和输出事件 XEVT，它们都与固定 EDMA 通道进行了绑定；
- 2) CPU 方式：McBSP 通过中断方式通知 CPU 进行数据的传输，有接收中断 RINT 和输出中断 XINT，中断的设定由 SPCR.RINTM 和 SPCR.XINTM 控制，CPU 也可以通过轮询方式来控制数据的传输。

课程设计中需要使用到两个数据引脚，使用的串口资源 DRR 和 DXR 分别完成音频信号的输入和输出，每采集到一次音频信号，就会通过 XINT 完成中断信号的发出，此时数据通过 16bit 总线被 CPU 调用。

### 1.2.2 DSP/BIOS 实时多任务操作系统内核:

DSP/BIOS 是一个简易的实时嵌入式操作系统，主要面向实时调度与同步、主机/目标系统通信，以及实时监测等应用，具有实时操作系统的诸多功能，如任务的调度管理、任务间的同步和通信、内存管理、实时时钟管理、中断服务管理、外设驱动程序管理等。

项目中 DSP/BIOS 使用的主要模块有以下三部分：HWI 硬件中断管理、LOG 时间记录显示和 SWI 软件中断管理中断。处理任务一般要划分为两个部分：一个是控制部分，花时间少，放在 HWI 函数中；另一部分是处理部分，放在 SWI 函数或任务中处理。

DSP/BIOS 具体模块的使用方式如下：

- 1) Log 模块：标准输入输出模块，占用内存少执行速度快。输出到 log 监视窗口，优先级很低。

参数：bufseg：日志缓冲区的存储段名称；buflen：日志缓冲区的大小（以字为单位）；logtype：说明日志类型，循环或者固定；fixed（固定）：只存储其最先接收的信息，当消息缓冲区满时就会拒绝接收新的信息；circular（循环）：当消息缓冲区满时，新日志会自动覆盖原有的日志；datatype：如使用 LOG\_printf 函数来打印输出日志信息时，请选择“printf”类型。如使用 LOG\_event 函数记录日志信息，请选择“raw data”。

API： LOG\_enable(&trace);

LOG\_printf(&trace, "Hello DSP/BIOS %d.", 0);

配置细节如下图所示：

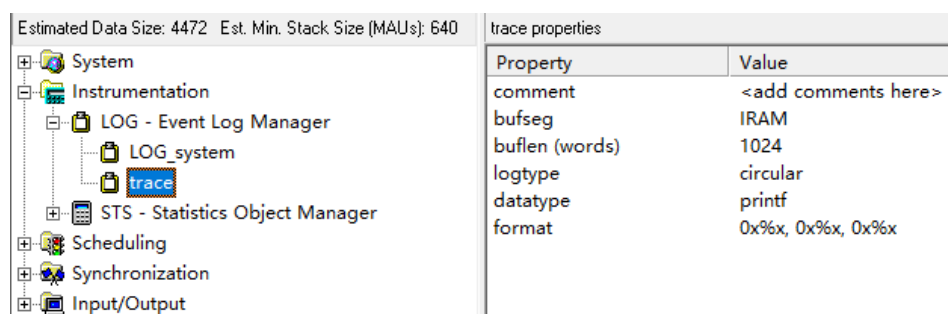


图 1 Log 模块配置



- 2) SWI 模块：一旦软中断邮箱的值满足一定条件时，触发软件中断，触发后一次执行完成（除非被硬件中断打断），执行完后软中断邮箱恢复到复位值，软中断中绝不会有死循环。

参数：function 相关函数名，priority 优先级，mailbox 软中断邮箱

API：void SWI\_post(&estimate\_SWI) // 软中段调用

void SWI\_andn(swi, mask); // 邮箱值与 mask 进行与非操作，当值为 0 时触发软中断

void SWI\_or(swi, mask); // 邮箱值与 mask 进行或非操作，当值为 0 时触发软中断

void SWI\_inc(swi); // Increment SWI's mailbox value and post the SWI

void SWI\_dec(swi); // Decrement SWI's mailbox value and post if mailbox becomes 0

时限 100us 以上，SWI 允许 HWI 将一些非关键处理在低优先级上延迟执行，这样可以减少在中断服务程序中的驻留时间每个 SWI 对象有一个邮箱，可以决定是否触发该中断。

用于触发 SWI 的 API 函数可以对邮箱值作不同的操作，软中断触发后一次执行完成（除非被硬件中断打断），执行完后软中断邮箱恢复到复位值。

配置细节如下图所示：

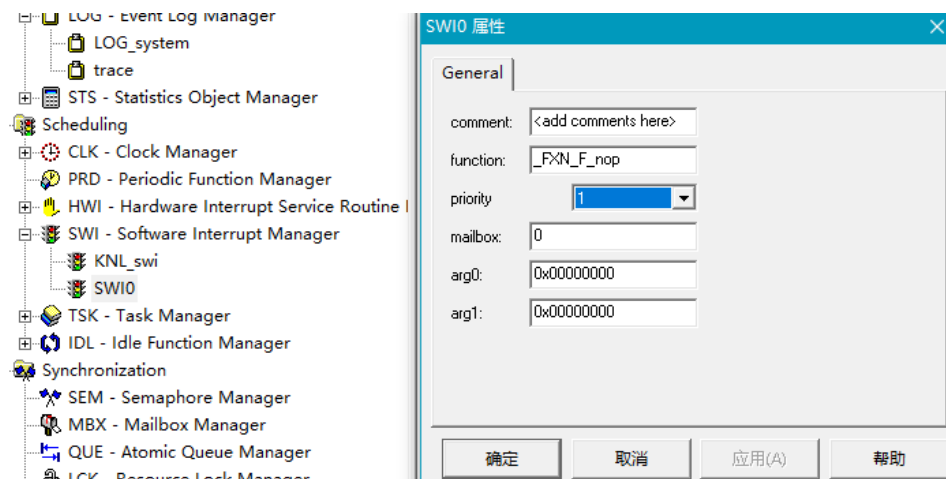


图 2 SWI 模块配置

- 3) HWI 模块：硬中断中包含 INT4 到 INT15 可自定义用户中断，其中 INT4 优先级最高。

参数：interrupt selection number 中断事件号，function 中断相应函数名，monitor 监视，Use Dispatcher 调度在调用 HWI 函数的前后自动调用 HWI\_enter 和 HWI\_exit，

## Interrupt Mask 中断掩字

频率可达 200KHz (5us)，处理时限在 2us~100us，包括 CLK 函数。配置细节如下图所示：

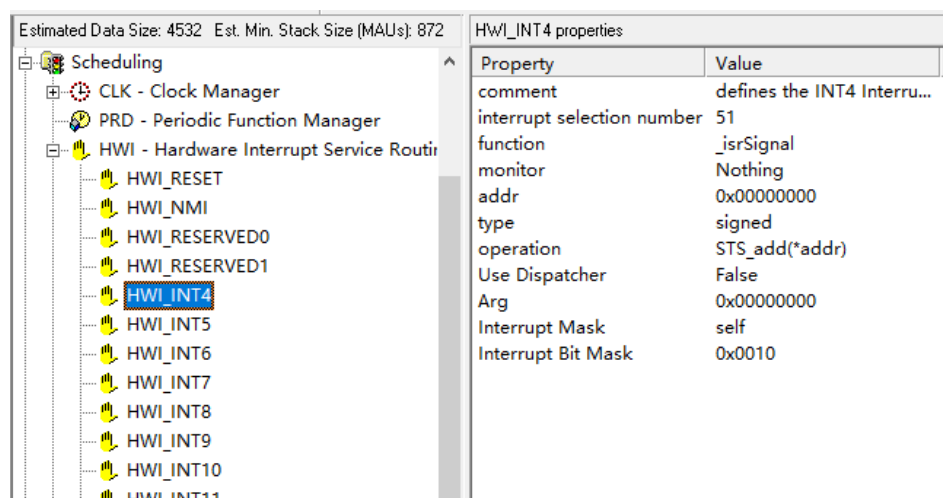


图 3 HWI 模块配置

## 1.2.3 相位差分法分析：

相位差分法：将一个采样信号序列分为两段，分别做 DFT 变换，然后找出两段 DFT 变换序列的谱线峰值位置的相差，用它来估计信号的频率与相位。其运行流程如下：

- 1) 对  $S1[n]$  和  $S2[n]$  等长子序列进行傅立叶变换，频率分辨率为  $\Delta f$ ；
- 2) 分别检测到  $S1[k]$  和  $S2[k]$  的最大谱线，计算得到频率粗测值  $f_k = k_0 * \Delta f$  和幅度估计值  $\hat{a}$ ；
- 3) 由检测到的两条最大谱线计算出相位差  $\Delta\phi = \phi_2 - \phi_1$ 、频率偏差  $f\delta = \Delta f * \Delta\phi / 2\pi$ ；
- 4) 计算出频率估计值  $f_0 = f_k + f\delta$  和初相位  $\phi_0 = \phi_1 - (N-2) * \Delta\phi / 2N$ ；

相位差分法具有以下特点，以及在进行编程实现时需要注意的一些问题：

- 1) 可以采用 FFT 快速算法，运算速度较快，适合实时信号的处理，但是 DFT 频率分辨率和频率估计精度取决于信号的测量时间长度，信号测量时间成为了限制因素之一。
- 2) DFT 对正弦信号有显著的信噪比增益，但是当信号频率不是 DFT 频率分辨率的整数倍时，由于频谱泄露使得信号幅度的估计值产生较大偏差，而且该偏差不会随 DFT 长度增加而减小。
- 3) 由于栅栏效应当频率为 DFT 分辨率一半的时候，频谱内就会有两条幅度相当的谱线，此时在噪声存在的情况下，方法就会失效。

总结：具有一定信噪比增益，运算量小、实时性好，幅度估计精准度不足，对采样有一些要求；

#### 1.2.4 最小二乘法拟合分析：

最小二乘拟合法：通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。

其运行流程如下：

##### 四参数正弦曲线拟合 `curvefit_4para()`：

- 1) 设定停止条件方差范围  $he$ ，用圆周计点法获取每周期点数  $m$  和周期数  $p$ ；
- 2) 获得频率估计值  $f0=fs/m$  极其收敛区间界  $\Delta f=fs/n$ ，确定迭代边界  $fl$ 、 $fr$ ，中值频率  $fm$ 、 $ft$ ；
- 3) 在  $fl$ 、 $fr$ 、 $fm$  和  $ft$  上执行三参数正弦曲线拟合 `curvefit()`，获得四组拟合参数；
- 4) 进行残差的比较，确定下次迭代参数，判定拟合残差大小；
- 5) 当残差平方和满足停止条件即满足测量要求。

##### 三参数正弦曲线拟合 `curvefit()`：

理想正弦信号： $y(t) = E1 * \cos(2\pi ft) + E2 * \sin(2\pi ft) + Q$

有效方差： $\varepsilon = \sum_{i=1}^n [y_i - A_1 \cos(2\pi ft_i) - B_1 \sin(2\pi ft_i) - C]^2$

- 1) 根据序列时刻  $t_i$ ，计算  $\alpha_i = \cos(2\pi ft_i)$  和  $\beta_i = \sin(2\pi ft_i)$ ，和其平均值  $\alpha$ ， $\beta$ ， $y$ ；
- 2) 代入方程计算  $AN$ 、 $AD$ 、 $BN$  和  $BD$  四个参数；
- 3) 参数  $A1=AN/AD$ 、 $B1=BN/BD$ 、 $C=y-A1*\alpha-B1*\beta$  则为理想正弦信号的拟合参数，使得有效方差最小。

最小二乘拟合法具有以下特点，以及在进行编程实现时需要注意的一些问题：

- 1) 计算量较大，幅度值和频率值估计较相位值精确；
- 2) 可手动设置迭代条件调节精度，但会带来更多的计算耗时；
- 3) 三参数拟合法在频率值给定的条件下，计算的幅度值几乎没有误差，误差来源于拟合算式；

- 4) 四参数拟合法的误差来源于迭代中值频率与实际频率的差值，依赖于迭代条件；
- 5) 算法计算时间与迭代次数有关，更优的 首次迭代中值频率的计算方法 可使迭代次数有效减少；

#### 1.2.5 两种估算方法的选择：

基于最小二乘法拟合与相位差两种方法的分析，由于在 DSP 上采用相位差算法能够减少估算的时间，且根据相位差得出的频率估计值更为精确，计算过程短且思路清晰，便于工程实现等特点，小组选用相位差方法完成采样后续的估算过程。

进行工程中的相位差分法估算，首先需要将时域正弦信号中频率、幅值和相位在频谱中对应的表现形式找到，根据频谱特点来推断正弦参数的具体数值。相位差分法的思路主要是首先直接通过 FFT 参数测值，然后使用相位差分对粗测估值进行矫正。

## 1.3 针对工程问题的具体实现

### 1.3.1 相位差分法 MATLAB 实现：

验证相位差分法需要首先生成一个采样信号的序列，代码第一段部分定义了一系列信号参数与采样参数，由此来构建一个  $N$  点的正弦信号采样序列以进行后续操作。如果最后运算结果与开始定义的信号参数接近则表示相位差分法的初步实现。

相位差分法的实现思路是，首先生成一段  $N$  点信号，分为前后两部分分别作 FFT 变换，通过谱峰获取两段信号的频率值和幅度值，同时能够计算出两个谱峰  $k1$  和  $k2$  所在的相位。计算相位差后可以获得误差的纠正值。

代码 1 相位差分法实现

```
f=122; %信号频率 A=20; %信号幅度 s=pi/6; %信号相位
t=22; %采样时间 N=1024; %采样点数 Ts=N/t; %采样频率
deltaf=Ts/N; %频率分辨率

x1=linspace(0,t/2,N/2); %第一段采样时刻
x2=linspace(t/2,t,N/2); %第二段采样时刻
y1=A*sin(2*pi*f*x1/Ts+s); y2=A*sin(2*pi*f*x2/Ts+s); %采样值
z1=fft(y1,N/2); z2=fft(y2,N/2); %生成两段频谱

[Ak1,k1]=max(abs(z1)); %求模取最大值
fk1=(k1-1)*fs/N; %得到频率粗测值
a1=Ak1*2/N; %获取幅度估计值
ang1=angle(z1)/pi*180; %获取相位谱
phi1=ang1(k1); %获取相位估计值
phi2=ang1(k2); %获取另一段信号的相位估计值

deltaphi=phi2-phi1; %得到相位差
ff=deltaphi*deltaf/2/pi; %得到频率偏差
f0=fk1+ff; %得到频率估计值
phi0=(3*N-2)*phi1/(2*N)-(N-2)*phi2/(2*N); %得到初相估计值
```

### 1.3.2 四参数拟合 MATLAB 实现：

四参数拟合中同样定义了一系列信号参数与采样参数，构建了一个正弦信号采样序列，经过首次迭代判断后进入循环的迭代过程进行逐步逼近。

前段会通过配置的信号频率和采样频率模拟生成一段采样点，然后通过初步地周期计数方法计算出频率的估计值，确定第一步迭代的频率参数，通过一次迭代调用三参数拟合法返回的估计数值，确定下一次迭代的频率参数，当计算的结果方差达到定义的精度标准时，则迭代结束，返回估计值。

代码 2 四参数拟合法实现

```
f=122; %信号频率    A=20; %信号幅度    s=pi/6; %信号相位
t=22; %采样时间    N=1024; %采样点数    Ts=N/t; %采样频率
fs=1/Ts; deltaf=Ts/N; %频率分辨率
x=linspace(0,t,N); %采样时刻    y=A*sin(2*pi*f*x/Ts +s); %采样值

%初步计算频率，用于确定迭代参数
he=0.01; %迭代条件    m=f*t; %周期计点
p=N/m; %周期点数    f0=fs/m; %频率估计值
%确定迭代参数
f1=f0-fs/N;          fr=f0+fs/N;
fm=f1+0.618*(fr-f1); ft=fr-0.618*(fr-f1);
%循环迭代
while (1)
[AL, setaL, CL, omgL]=square_fitting(f1,x,y,N);
[AR, setaR, CR, omgR]=square_fitting(fr,x,y,N);
[AM, setaM, CM, omgM]=square_fitting(fm,x,y,N);
[AT, setaT, CT, omgT]=square_fitting(ft,x,y,N);
%计算下次迭代参数
if(omgM<omgT)    omg=omgM; f1=ft; ft=fm; fm=f1+0.618*(fr-f1);
else            omg=omgT; fr=fm; fm=ft; ft=fr-0.618*(fr-f1);
end
%计算参数估计结果
if (abs((omgM-omgT)/omgT/k)<he)
    if(omg==omgM) A0=AM; f0=fm; C0=CM;
    else A0=AT; f0=ft; C0=CT;
    end break;
end end
```

### 1.3.3 参数拟合迭代过程 MATLAB 实现：

三参数拟合中主要是对每一次迭代的参数进行计算，然后返回估算结果和方差，用于计算下一次迭代条件等，是一个受调用的函数。

计算的过程为：先生成两个正余弦数组，然后求得两者均值，用于之后四个拟合参数的计算，四个参数计算完毕后即可获得一次迭代下的幅度估计值，相位估计值和计算方差。

代码 3 三参数拟合迭代过程实现

```
function [A0,seta,C,omg]=square_fitting(f,x,y,n)
alpha=cos(2*pi*f*x); %alpha 数组  beta=sin(2*pi*f*x); %beta 数组
avgalpha=mean(alpha(:)); %求均值 avgbeta=mean(beta(:)); %求均值
avgy=mean(y(:));
%计算拟合参数
An=(sum(y.*alpha)-avgy*sum(alpha))/(sum(alpha.*beta)-avgbeta.....;
Ad=(sum(alpha.*alpha)-avgalpha*sum(alpha))/(sum(alpha.*beta).....;
Bn=(sum(y.*alpha)-avgy*sum(alpha))/(sum(alpha.*alpha)-avgalpha.....;
Bd=(sum(beta.*alpha)-avgbeta*sum(alpha))/(sum(alpha.*alpha).....;
%一次拟合下的幅度和方差计算值
A1=An/Ad;      B1=Bn/Bd;
C=avgy-A1*avgalpha-B1*avgbeta;
A0=sqrt(A1^2+B1^2);
sigmn=sum((y-A1*alpha-B1*beta-C).^2);
omg=sqrt(sigmn/n);
%一次拟合下的相位估计值
if(A1>=0) seta=atan2(-B1,A1);
else      seta=atan2(-B1,A1)+pi;
end; end
```

### 1.3.4 输入信号的中断采样输出

LAB3 中将输入的信号经过一次缓存后进行处理，将左右声道数据分离然后经过滤波器合成输出，与实验中对信号的采样的实现原理有相似之处，可以进行适当的改造和删减，从而达到对于音频信号的 16bit 采集。

main 函数中首先对 EVM6437 开发板进行初始化和基本配置，将采样值存储数组初始化；isrAudio 为一次硬件中断执行的操作，将 McBSP1 串口中的 16 位传给全局变量 buffer 保存；采样点数达到 N 之后触发软件中断 estimate 进行估算。配合 DSP/BIOS 中的 HWI 和 SWI 进行配置，可以实现定

时采样后的处理。

#### 代码 4 输入信号采样实现

```

/*  main 函数对 6437 进行初始化和基本配置，将采样值存储数组初始化
   isrAudio 为一次硬件中断执行的操作，将 MCBSP1 串口中的 16 位传给 buffer 保存
   采样点数达到 N 之后触发软件中断 estimate 进行估算 */
void isrAudio(void);           //硬件中断
void estimate(void);           //软件中断
void curvefitpara(&phase, &freq, &A); //相位差分

// === MAIN - init and return to begin BIOS Scheduler environment...=====
void main(void){               // main: setup prior to BIOS
    LOG_printf(&trace, "EVM6437 started!");
    EVMDM6437_init();          // init EVM6437 HW
    initCodec();                // init McBSP1; s/u AIC via I2C

    for (i = 0; i < N; i++) buffer[i] = 0;    // clear delay line
    ICR = 0x10;                  // clear int4 (precaution)
    IER |= 0x10;                 // enable int4 as CPU interrupt
    MCBSP1_SPCR = 0x00010001;    // start McBSP
}

// ===== isrAudio() - Serial Port interrupt Service Routine: BIOS HWI
void isrAudio(void){            // get, process, send data: IPO
    static short dataIn;        //采样缓存
    short mark=0;               //采样点计数
    dataIn=MCBSP1_DRR_16BIT & 0x0ff;    //获得采样值
    if(mark<N){
        buffer[mark]=dataIn;    //存储采样值
        mark++;                 //已采样点增加
    }
    if(mark==N) SWI_post(&estimate_SWI); //软件中断
}

void estimate(void){            //软件中断
    short i=0;                  //将采样值打印
    for(i=0;i<N;i++) LOG_printf(&trace,"value=%d",buffer[i]);
    //curvefitpara(&phase, &freq, &A); //参数估计
}

```



## 1.4 知识技能学习情况

在工程的开发过程中小组成员学习了 MATLAB 的相关函数，使用 MATLAB 实现了最小二乘法拟合的实现和相位差分法的实现。在 MATLAB 的具体使用中掌握了特殊变量与常数、基本数学函数、基本矩阵和矩阵操作和绘图函数等常用函数与指令，完成了采样数据的模拟生成与傅立叶变换等基本过程，最终实现了测量方法在 MATLAB 上的编写。

在进行工程的实验前小组对于 DM6437 的基本信息进行了初步了解，查阅了相关的功能框图和硬件描述，为后期的编程设计提供参考。

了解了 DSP/BIOS 的基本知识，SWI, HWI, Log 模块的使用，McBSP 串口的使用方式等。对于 DSP 的 FFT 实现思路，进行了相关文档的查阅，查找到了混合基 FFT 的常规 C 语言实现的相关代码：

```
void CooleyTukeyFft16x16(int N, short wn[], short x[], short y[]);
```

它要求正常顺序的输入  $x[]$ ，产生正常顺序的输出  $y[]$ 。输入，输出和旋转因子  $w_n[]$  都是复数，实部和虚部存在数组中相邻的单元，实部在偶数单元，虚部在奇数单元。所有数据都是 16 位的 Q.15 格式的小数。

根据 FFT 的常规 C 语言实现的思路找到了 TI 提供的 C64x+ DSP 算法库中包含的多种 FFT 优化实现。

具体命名规则以及可用函数如下：

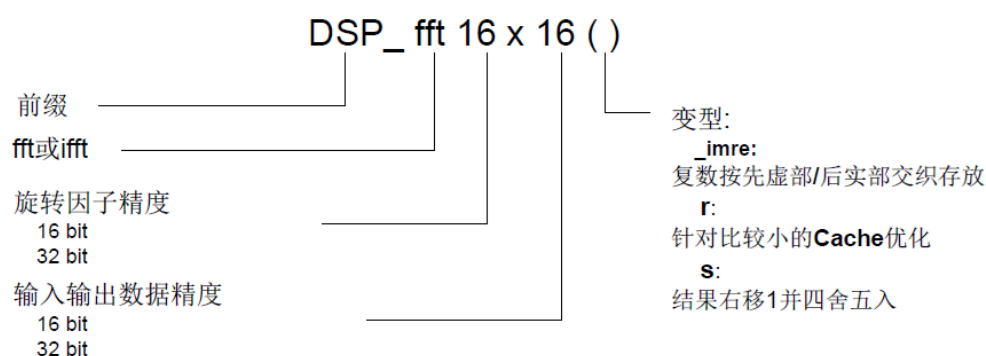


图 4 DSP 快速傅立叶变换库函数

表 1 DSP 快速傅立叶变换库函数表

函数	说明
void DSP_fft16x16(short *w, int nx, short *x, short *y)	16 bits 复输入输出数据, 复数按先实部/后虚部交织存放除最后一级外, 每级的结果右移 1 并四舍五入
void DSP_fft16x16_imre(short *w, int nx, short *x, short *y)	16 bits 复输入输出数据, 复数按先虚部/后实部交织存放除最后一级外, 每级的结果右移 1 并四舍五入
void DSP_ifft16x16(short *w, int nx, short *x, short *y)	
void DSP_fft16x32(short *w, int nx, int *x, int *y)	
void DSP_fft32x32(int *w, int nx, int *x, int *y)	

## 第二章 存在问题与解决方案

### 2.1 存在的主要问题

1. 对于 LAB3 示例工程进行更改后出现疑似重定义的问题，编译时出现：

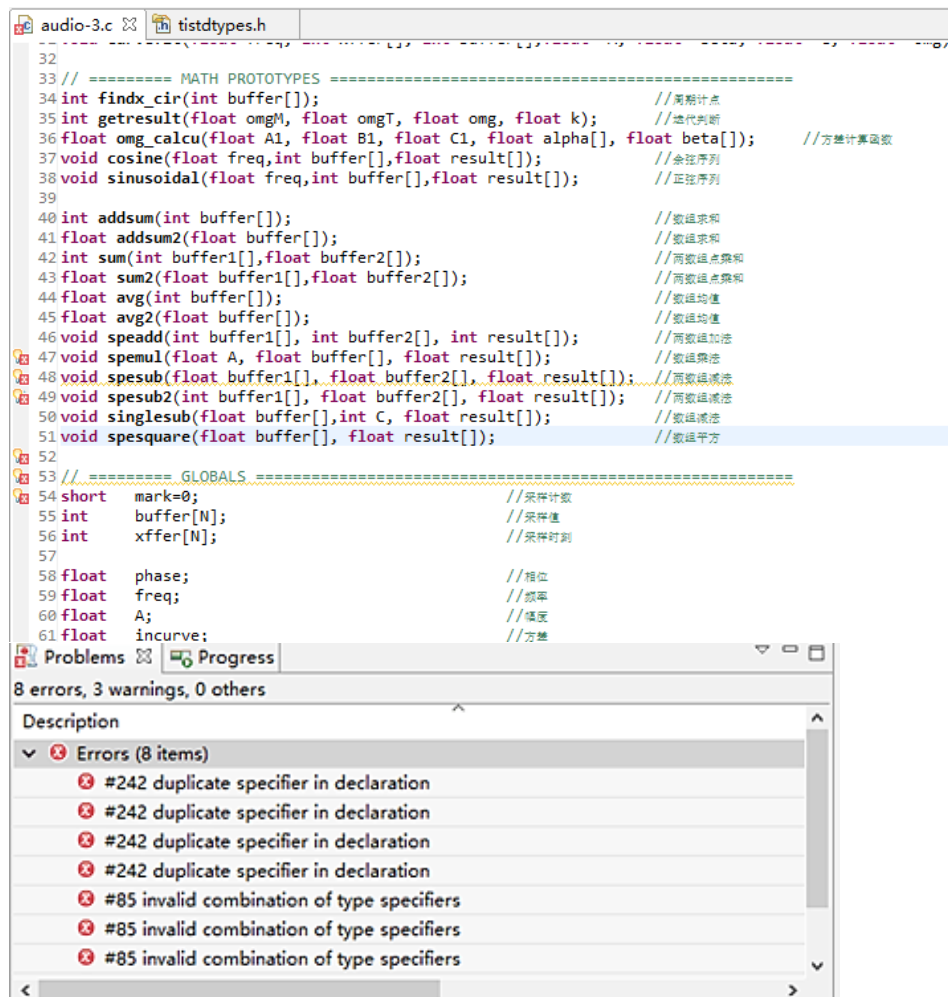


图 5 项目重定义报错

错误定位的位置在 tistdtype.h 头文件中如下位置：

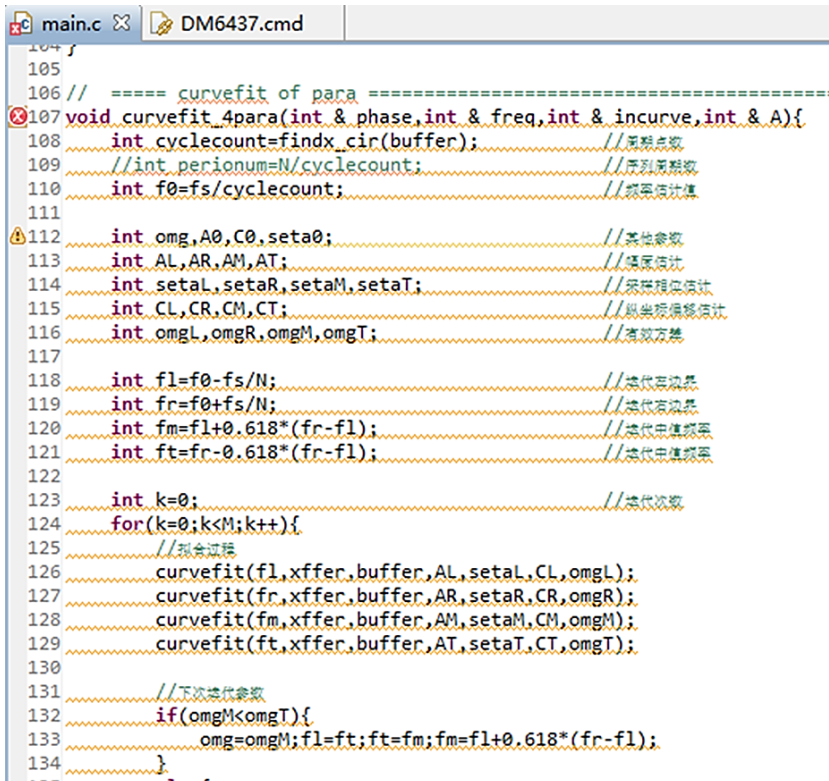
```

44 /* Handle the 6x ISA */
45 #if defined(_TMS320C6X)
46     /* Unsigned integer definitions (32bit, 16bit, 8bit) follow... */
47     typedef unsigned int      Uint32;
48     typedef unsigned short    Uint16;
49     typedef unsigned char     Uint8;
50
51     /* Signed integer definitions (32bit, 16bit, 8bit) follow... */
52     typedef int               Int32;
53     typedef short             Int16;
54     typedef char              Int8;
55
56

```

图 6 重定义错误指向位置

2. CCS 平台不能识别引用传参的语法：



```

104 }
105
106 // ===== curvefit of para =====
107 void curvefit_4para(int & phase, int & freq, int & incurve, int & A){
108     int cyclecount=findx_cir(buffer); // 周期点数
109     int perionum=N/cyclecount; // 周期数
110     int f0=fs/cyclecount; // 频率估计值
111
112     int omg,A0,C0,se0; // 基频参数
113     int AL,AR,AM,AT; // 谐波估计
114     int setal,setar,setam,setat; // 谐波相位估计
115     int CL,CR,CM,CT; // 谐波幅值估计
116     int omgL,omgR,omgM,omgT; // 谐波频率
117
118     int fl=f0-fs/N; // 迭代高边界
119     int fr=f0+fs/N; // 迭代低边界
120     int fm=fl+0.618*(fr-fl); // 迭代中值频率
121     int ft=fr-0.618*(fr-fl); // 迭代中值频率
122
123     int k=0; // 迭代次数
124     for(k=0;k<M;k++){
125         // 拟合过程
126         curvefit(fl,xfer,buffer,AL,setal,CL,omgL);
127         curvefit(fr,xfer,buffer,AR,setar,CR,omgR);
128         curvefit(fm,xfer,buffer,AM,setam,CM,omgM);
129         curvefit(ft,xfer,buffer,AT,setat,CT,omgT);
130
131         // 下次迭代参数
132         if(omgM<omgT){
133             omg=omgM;fl=ft;ft=fm;fm=fl+0.618*(fr-fl);
134         }
135     }
136 }

```

图 7 函数引用传参作物

## 2.2 解决方案

对于疑似重定义的问题则需要重新搭建工程逐函数进行排查，引用传参的问题需要后续改为指针传参来解决。其他算法上存在的错误等问题将在后续的调试过程中进行逐一排查解决。



### 第三章 前期任务完成度与后续实施计划

目前小组已经完成了在 DSP/BIOS 中对于软件中断以及硬件中断的配置，以及在 MATLAB 中实现了相位差分以及法最小二乘法的拟合，验证了两种方法实现的有效性，并进行了对比。

小组前期的工作主要围绕着两种方法的仿真实现展开，在 MATLAB 的编写环节中也出现了生成序列错误和测量结果无效等问题，无法进行后续的验证环节和 C 语言实现环节。对存在的错误进行分析消耗掉了一部分时间。但总体上进度进展顺利，后续的工作将在 CCS 平台上具体实现。

前期的工作中团队的主要分工如下：

胡震和楚蕃源负责资料收集，分析相位差分法和曲线拟合法，分析原理绘制控制流图等，提供改进建议等；唐浩和滑文博在 CCS 平台上对于官方提供的工程示例进行测试，并提供可用 API 及 DSP/BIOS 使用建议等；刘恒利和何劲帆在 MATLAB 上对与两种方法进行编写和仿真提供结果。

对于后续任务的展开，主要分为两个部分，首先进行音频信号的采样输出，通过硬软件中断的方式，首先了解采样得到的数据是什么样的；第二是在 C 语言方式下实现相位差分法，验证其正确性。两个部分实现完成后进行整合就能实现最终的测量功能要求。





## 参考文献

- [1] 张涛, 杨炳恒, 樊向党. 基于 FFT 的正弦信号幅值估计研究[J]. 兵器装备工程学报, 2016, 37(7):90-93.
- [2] 张晓威, 孟凡明. 正弦信号幅值和初相位估计的问题研究[J]. 计算机工程与应用, 2013, 49(5):216-219.