南京理工大学

NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

电子工程与光电技术学院

实验报告

课程名称：DSP 应用技术

实验名称：任意信号发生器

班　　级：9151040G02

姓　　名：傅　超

学　　号：9151040G0216

指导老师：李彧晟

2018 年 11 月 22 日

目　录

1 实验目的

1、熟悉 DSP 硬件开发平台

2、熟悉 DSP 集成开发环境（CCS）

3、掌握 TMS320F2812 的存储器配置表

4、学习 TMS320F2812 的编程开发

5、熟悉代码调试的基本方法

2 实验仪器

计算机，C2000 DSP 教学实验箱，XDS510 USB 仿真器，示波器

3 实验内容

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 平台实现任意波的产生，通过示波器观察结果。

4 实验准备

在 DSP2000 实验平台上实现任意波形的产生，可通过 DSP 实时运算得到相应波形的数据，随后通过 DAC 完成模拟输出。在该实验中，我们利用 DSP 的运算能力，首先计算出波形的数值信息，存储到相应的数据空间中，通过查表的方式读取该波形的数值并写入到 DAC 端口，实现任意波形的生成。由此可得程序流程如图 10.1 所示。

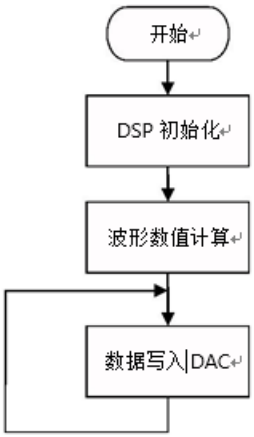


图 1：任意波形发生程序流程

10.4.2 数据的定标

TMS320C28xx 是定点 DSP 芯片，采用定点数进行数值的运算，其操作数一般采用整型或长整型数据。数据最大表示范围取决于 DSP 芯片给定的字长，字长越长，所能表示数的范围就越大，精度也越高。数据以 2 进制补码格式表征，最高位是符号位，其余 15 位表示数值的大小。

而在实际中，数值的大小、数据的运算都会带来小数，用定点数格式表示小数，确定小数点的位置，称之为数据的定标。数据的定标一般有 Q 表示法，即 Q15 表示在定点数格式中有 15 位小数。由此 16 位定点数有 16 种 Q 表示形式，对应了 16 种十进制数据范围。例如 16 位定点的 Q0 表示没有小数，数据范围[-32768, 32767]；Q4 表示有 4 位小数，数据范围[-2048, 2047.9375]；Q15 表示有 15 位小数，数据范围[-1, 0.9999695]。可见，不同 Q 所表示的数据范围和精度都有所不同，精度与范围是一对矛盾，在实际定点算法中，为了达到最佳性能，必须对数据进行合理的定标。

浮点数 X_F 与定点数 X_D 的转换关系可表示为： 定点数 $XD=\lfloor X_F \times 2^Q \rfloor$

浮点数 $X_F = X_D \times 2^{-Q}$

在程序中，根据数据的动态范围来确定 Q 值，分析程序中的数据可能的绝对值最大值|max|，使下式成立：$2^{n-1} < |max| < 2^n$，则 Q＝15－n。

10.4.3 相关实验硬件资源

TMS320F2812 内部采用哈佛结构总线，与 TMS320F24xx 以及 TMS320F206 系

列 DSP 不同，内部的程序空间、数据空间采用统一的编址方式。其次 TMS320F2812 支持 32 位格式的数据访问，32 位的数据访问必须从偶地址开始。

由于 TMS320C28xx 绝大部分指令采用 32 位，因此，当程序存放到程序空间时，必须分配到偶数地址空间。

除了 TMS320F2812 片上集成的存储器，在 DSP2000 实验箱上还扩展了 RAM、FIFO、双端口存储器等资源，供实验者使用。其地址分配如下表 10.1 所示。

地址范围	存储器	等待时间	备注
0x08,0000 ~ 0x08,0FFF	双端口 RAM	至少 2 等待	占 ZONE2
0x10,0000 ~ 0x13,FFFF	SRAM	至少 2 等待	占 ZONE6
0x14,0000 ~ 0x14,FFFF	FIFO	至少 4 等待	占 ZONE6

表 1：外扩存储器地址映射

DSP 处理器的外部接口（XINTF）负责完成对外扩设备的连接管理。TMS320F2812 的 XINTF 映射到 5 个独立的存储空间，分别是 ZONE0、ZONE1、ZONE2、ZONE6 和 ZONE7。

每一个空间都有一个内部的片选信号，并可以通过编程来独立的配置访问等待、选择、建立以及保持时间，以实现 TMS320F2812 与各种外部存储器或设备的无缝连接。

实验箱上的 DAC1 采用的是 AD768，位宽 16bit，数据以无符号数表示，转换速度 30ns，通过 OUT3 端口输出，在 TMS320F2812 的地址映射为 0x2900（只写）。即 DSP 只要将数字信号写到该端口，DAC1 自动完成模拟的转换。

实验箱中的 8 个数码显示管为共阴级显示管，即只要对相应的显示位写 1，就可点亮该位，写 0，则熄灭该显示位。

若要使 LED 显示字符，可先往相应的端口写入字符对应的码字，而后往 LED 数据更新端口写任意数，即可刷新 LED 显示的字符。比如 LED1 显示字符“A”，先往端口 0x2700 写入字符 0x77，随后往端口 0x2C00 写入 0x00。

5 实验步骤

1、设备检查

检查仿真器、C2000 DSP 实验箱、计算机之间的连接是否正确，打开计算机和实验箱电源。

2、启动集成开发环境

点击桌面 CCS 2（C2000）快捷方式，进入集成开发环境 CCS。

3、新建工程

新建一个 DSP 工程，编辑源程序、配置命令等相关文件，并在工程中添加这些程序文件。

要求产生一个线性调频信号，其数学表达式如下所示：

$$s(t) = \cos(\pi Kt2\ ) \quad (10.1)$$

其中调制斜率 K 为 39062，t 为持续时间是[-0.0128, 0.0128]，在采样时间内共 1024 个采样点，即有 1024 个离散数值。

4. 建立工程（Build）

建立工程（build），若出错，则根据错误提示，修改源程序文件或者配置命令文件，直至编译链接正确，生成可执行的.out 文件。

5. 加载程序

在主菜单下，选择“File → Load Program”，将程序下载到 DSP 内部。

6. 调试程序

在程序中的“波形数值计算”子模块后设置断点，运行程序后 PC 指针会停留在此处，打开图形显示功能，查看存储空间中保存的时域波形，是否为线性调频信号。如果不是，则重新修改程序，直至正确为止。

7. 运行程序

重新全速运行程序。连接 C2000 实验箱 OUT3 输出口至示波器，调节示波器，观察线性调频信号的输出。

6 实验结果

6.1 基本要求

1、利用数码显示管，在 DSP 初始化子模块后添加语句或者编写子程序，使之能够显示实验日期。为了让程序清晰有条理，我单独编写了一个 LED 显示子程序，在子程序中 LED 的文字可以通过提前定义每个数字对应的十六进制数字，这样在程序中可以直接调用，方便后期查看。

```
void showLED(){
    /* 初始化LED */
    // LED8
    *LED8 = CHAR_0;
    // LED7
    *(LED8+0x100) = CHAR_2;
    // LED6
    *(LED8+0x200) = CHAR_1;
    // LED5
    *(LED8+0x300) = CHAR_1;
    // LED4
    *(LED8+0x400) = CHAR_8;
    // LED3
    *(LED8+0x500) = CHAR_1;
    // LED2
    *(LED8+0x600) = CHAR_0;
    // LED1
    *(LED8+0x700) = CHAR_2;
    // WIRTE DATA TO LED
    *(LEDWR) = 0XFF;
}
```

```
#define CHAR_NULL   0x00
#define CHAR_L      0x38
#define CHAR_1      0x06
#define CHAR_2      0x5B
#define CHAR_8      0x7F
#define CHAR_0      0x3F
#define CHAR_DOT    0x80
#define CHAR_DH     0x08
#define CHAR_C      0x39
```

图 2：LED 数码管显示程序截图

在主程序中调用之后，LED 显示子程序即可在实验箱的数码管上看到实验当天的日期：20181120，从而成功实现了显示日期的功能，实验结果如下图所示。

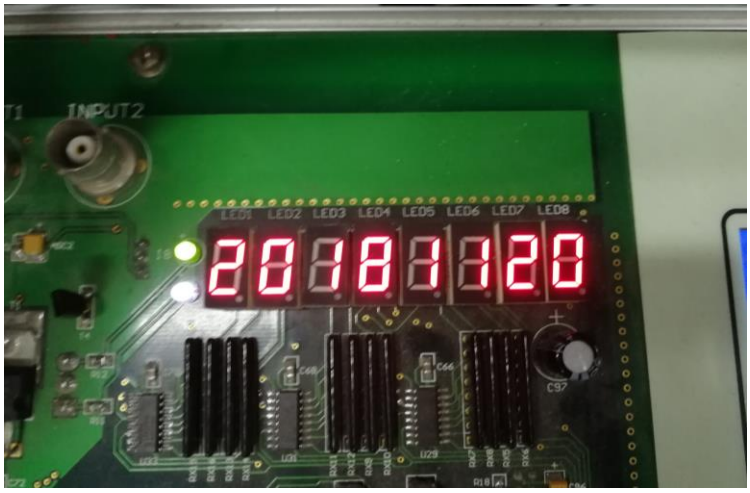
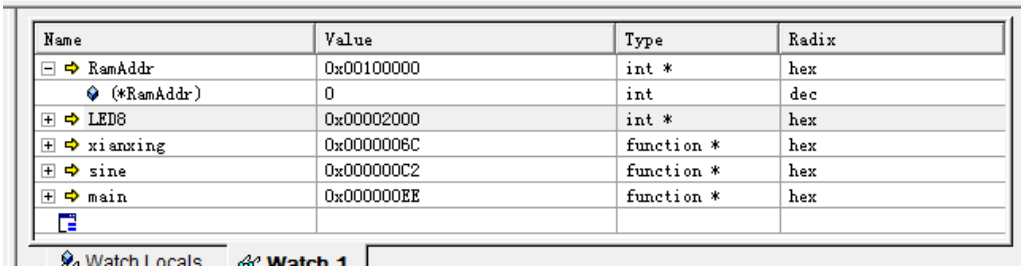


图 3：LED 数码管显示效果图

3、记录实验中个子程序包括主程序的入口实际地址，与 memory 比较，指出分别位于什么类型的存储器中。

通过将主程序和各个子程序的名称加入到变量窗口中可以看到主程序和不同的子程序对应的地址，其中主程序的地址为 0x000000EE，线性调频信号子程序的地址为 0x0000006C，正弦信号子程序的地址为 0x000000C2。



Name	Value	Type	Radix
RamAddr	0x00100000	int *	hex
(*RamAddr)	0	int	dec
LED8	0x00002000	int *	hex
xianxing	0x0000006C	function *	hex
sine	0x000000C2	function *	hex
main	0x000000EE	function *	hex

Watch Locals Watch 1

图 4：各个子程序入口地址

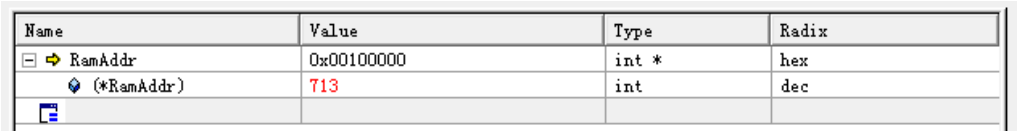
通过查看.map 的文件内容，可以看到以上程序的地址均在 PAGE 0 的 PROG 中，即内部存储空间数据存储空间。

```
     name          origin    length     used    unused   attr    fill
---------------------- --------  ---------  --------  --------  ----  --------
PAGE 0:
  PROG             00000040  003fffc0  000004f1  003ffacf  RWIX
```

图 5：PAGE0 地址起点和长度

4、指出波形数据保存的空间地址，并以图形方式显示线性调频信号的波形。

从图中可以看到波形的数据存储地址为 0x00100000，存储有 1024 个整型数据。



Name	Value	Type	Radix
RamAddr	0x00100000	int *	hex
(*RamAddr)	713	int	dec

图 6：RamAddr 地址截图

线性调频信号产生程序如下所示。

```
void xianxing(int i){
    for(i=-512;i<512;i++)
        *(RamAddr+i+512) = (int)((cos(Pi*K*(i*32768/fs)*(i*32768/fs)/32768/32768)*2048));
}
```

图 7：线性调频信号产生程序截图

利用图形工具，将图形的起始地址设为 RamAddr，即波形数据的起始存储地址，

在图中显示 1024 个点的线性调频信号波形，图形如下：

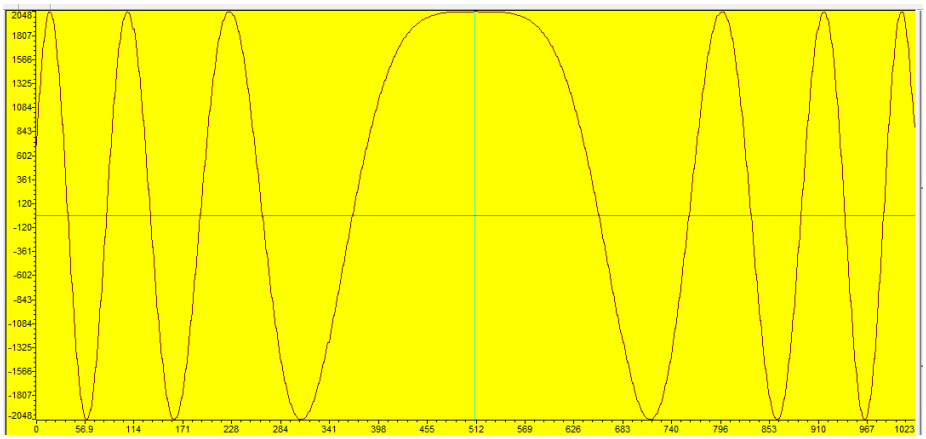


图 8：产生的线性调频信号仿真图

在示波器中可以看到该线性调频信号，图形如下所示。

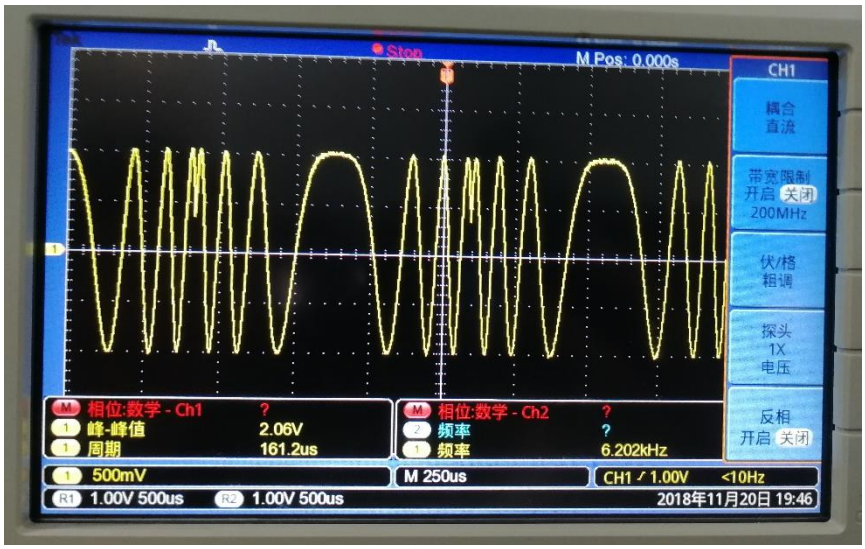


图 9：产生的线性调频信号实际输出图

6.2 实验提高

1、查看除.text、.data、.bss 段之外，还有哪些有实际长度的段，查找相关资料，指出其含义与作用。

```
.cinit    0   00000501   0000002e
              00000501   00000014   RamGen.obj (.cinit)
              00000515   0000000e   rts2800_ml.lib : exit.obj (.cinit)
              00000523   0000000a                  : _lock.obj (.cinit)
              0000052d   00000002   --HOLE-- [fill = 0]

.reset    0   00000530   00000002
              00000530   00000002   rts2800_ml.lib : boot.obj (.reset)

.bss      1   00000000   00000000   UNINITIALIZED

.stack    1   00000000   00000400   UNINITIALIZED
              00000000   00000400   --HOLE--
```

图 10：.map 其他字段的地址长度截图

从图中可以看到，除了.text、.data、.bss 字段之外，还有.stack 字段有长度，通过查阅资料可以了解到，这个是系统堆栈保留的空间，用于和函数传递变量或为局部变量分配空间；此外，还有.cinit 字段，这是全局变量和静态变量的 C 初始化记录，包含未用 const 声明的外部（extern）或静态（static）数据表。

2. 在保持源文件功能正确的前提下，仅修改.cmd 配置命令文件，改变段的地址分配，链接工程后，执行程序，以下为修改前后的地址分配图。

```
/*****************************************************
*   sinewave.cmd - Sample linker command file for F28xx devic
*
*   Description: This file is a sample F2812 linker command f
*                be used for linking programs built with the
*                C Compiler. Use it as a guideline; you may m
*                the allocation scheme according to the size
*                program and the memory layout of your target
*****************************************************/

MEMORY
{
   PAGE 0 : BOOT(R)      : origin = 0x3f8000, length = 0x80
   PAGE 0 : PROG(R)      : origin = 0x3f9000, length = 0x1f80
   PAGE 0 : RESET(R)     : origin = 0x3fffc0, length = 0x2

   PAGE 1 : MORAM(RW)    : origin = 0x000000, length = 0x400
   PAGE 1 : M1RAM(RW)    : origin = 0x000800, length = 0x400
   PAGE 1 : L0L1RAM(RW)  : origin = 0x008000, length = 0x2000
}
```

```
                     name           origin    length     used     unused   attr    fill
                  ----------------  --------  ---------  --------  --------  ----  -------
PAGE 0:
  BOOT                             003f8000  00000080  00000000  00000080  R
  PROG                             003f9000  00001f80  00000c22  00001358  R
  RESET                            003fffc0  00000002  00000000  00000002  R

PAGE 1:
  MORAM                            00000000  00000400  00000400  00000000  RW
  M1RAM                            00000800  00000400  00000400  00000000  RW
  L0L1RAM                          00008000  00002000  000007fc  00001804  RW


SECTION ALLOCATION MAP

 output                                                attributes/
 section   page    origin      length       input sections
 --------  ----  ----------  ----------   ----------------
 .pinit     0    003f9000    00000000     UNINITIALIZED

 .text      0    003f9000    00000adc
                   003f9000      00000232    rts2800_ml.lib : lowlev.obj (.text)
                   003f9232      000001f8                   : trgdrv.obj (.text)
```

图 11：修改. cmd 文件后的前后对比图

3、在不修改波形数值计算子模块前提下，即保持波形数值表中的数据，依照 DDS 原理，修改程序，调整线性调频信号的输出周期。

为了不改变波形数值表中的数据，所以在采样数据存储的程序不需要改变，而在最后从波形数据表中传输到 DAC 时，可以通过设定一个步进，每隔表中的几个点输出，就可以做到改变输出线性调频信号的频率，即改变线性调频信号的输出周期。

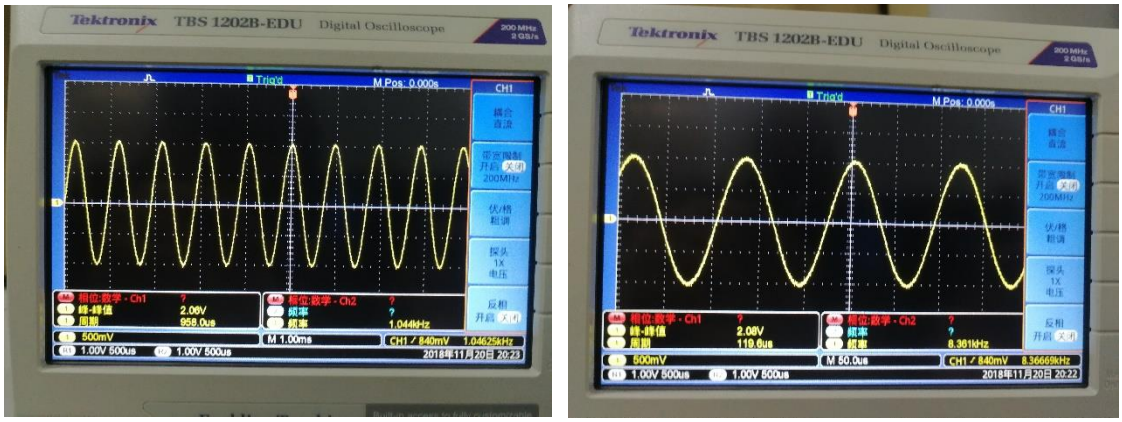


图 12：改变不同频率控制字后的示波器输出图

对于正弦波也是同理，从以上图中可以看到，通过改变程序中的步进可以实现改变输出波形的频率大小。

具体的实现程序如下。

```
while(1)
{     for(i=0;i<1024/1;i++)
      {
      //tmp = (int)(*(RamAddr+i))<<2;
      *(DAC1Addr) = (unsigned int)((*(RamAddr+1*i))<<2) + 0x8000;
      }
}
```

图 13：频率控制字程序截图

7 实验感悟

7.1 实验中遇到的问题与解决方案

7.1.1 数据定标

在本实验中，由于实验要求的时间范围为[-0.0128,0.0128]，所以在最先编写程序的时候没有考虑小数的问题，从而导致了最终的输出结果与预想的不一致，后经过查阅资料发现，由于实验中采用的是定点 DSP 芯片，所以采样点的数值需要通过同数据定标来来换算，后经过文献中的相关转换方法，对小数进行定标，最终 0.0128 定标后为 32768，再进行调试，最终成功。

7.1.2 线性调频信号的输出示波器波形混乱

在写完程序进行软件模拟绘图时发现，输出的线性调频信号和预想的不一致，正常的波形占一小部分，之后全是混乱的波形，后经过思考发现，在输出先行调频信号时，采样的数据存储地址编写有误，原先写得是*(RamAddr+i)，但是 i 是从-512 开始的，最终输出的波形又是从 RamAddr 开始的，所以这就导致了我最终输出的波形仅为我实际采样的数据的一部分，而没有数据的那一部分地址自然就输出了乱码，最终将采样的数据存储地址改为*(RamAddr+i+512)，获得的波形则为正常。

7.1.3 图形工具画出的波形错误

使用 CCS 中的图形工具，绘制出的图像波形前没有数据，波形后有杂乱的波形，经过研究发现是在绘制图象时将 16 位的数据误认为 32 位的数据，从而导致了图像错误，最终将图像数据选择为 16 位符号数，即绘制出了正确的图像。

7.2 实验的收获与感受

经过了第一次实验，我对于整个实验箱的操作以及 CCS 软件的操作有了大致的了解，所以在进行本次实验的时候我能够快速地了解程序并能够快速地运行。因为之前有过相似地编程经验，所以我在设计本次实验程序的时候，将每一个要求都写成一个子程序，比如将 LED 显示程序封装成一个子程序，将正弦信号、线性调频信号都封装成一个个子程序，然后在实际中可以根据需要调用相关的程序，这样可以使主程序的结构更整齐更容易读懂，也方便后期修改调用。

在本次实验中我还学会了如何使用数码管显示想要的数字，由于之前在微机实验中有接触过如何使用数码管，所以在本次实验中我通过阅读例程可以很快地了解如何调用数码管显示自己当天实验地日期。

由于 DDS 实验在之前的实验中有所接触，所以对于其原理还是有所了解的，但是由于当时的基础知识不够扎实，所以做实验的时候也没有很透彻地理解原理，在本次实验中，我在上次实验的基础上对于频率控制字有了更进一步的了解，并且能够在现有的例程中添加频率控制字，从而实现能够改变该变量来改变最后输出波形

的频率大小。

虽然偶尔会有吐槽总是做类似的实验，但是在每一次实验中真的会有新的收获，对于实验的态度也有所改观，其实 真正提高自己能力的是在实验中遇到问题并解决问题的时候，没有什么实验可以一次成功，只有在不断地改正，不断地踩坑中才能有真正地成长。之前做实验有一种急功近利的情况，一味地追求实验速度，对于中间遇到的问题也丝毫不考虑，认为只要结果对就可以，最终导致了每一次实验做完一段时间后就完全遗忘了。

通过翻阅之后的实验可以发现，这五个实验都是互相传承的，每一个实验都是后面实验的基础，所以这也让我们在做前面实验的时候踏踏实实，不能搪塞糊弄，否则后面的实验会越来越难做。这从侧面也反映出，我们在做实验前需要提前大致地了解后面的实验，这有助于我们对于整个实验体系的认知，以及明确每一个实验的重点。