实验一：常用指令实验

一、 实验目的

1. 熟悉 DSP 开发系统的连接
2. 了解 DSP 开发系统的组成和结构和应用系统构成
3. 熟悉常用 C54X 系列指令的用法(程序寻址，寄存器，I/O 口，定时器，中断控制)。

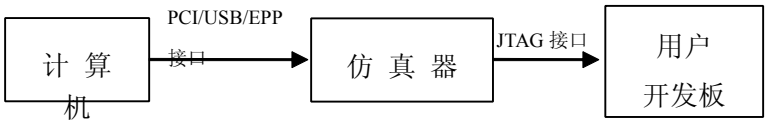
二、 实验设备

计算机，CCS 2.0 版软件，DSP 仿真器，实验箱。

三、 实验操作方法

1、系统连接

进行 DSP 实验之前，先必须连接好仿真器、实验箱及计算机，连接方法如下所示：



在硬件安装完成后，接通仿真器电源或启动计算机，此时，仿真盒上的“红色小灯”应点亮，否则 DSP 开发系统与计算机连接有问题。

2、运行 CCS 程序

先实验箱上电，然后启动 CCS，此时仿真器上的“绿色小灯”应点亮，并且 CCS 正常启动，表明系统连接正常；否则仿真器的连接、JTAG 接口或 CCS 相关设置存在问题，掉电，检查仿真器的连接、JTAG 接口连接，或检查 CCS 相关设置是否正确。

四、 实验步骤与内容

（一） 简单指令程序运行实验

1、实验使用资源

实验通过实验箱上的 XF 指示灯观察程序运行结果

2、实验过程

启动 CCS 2.0，并加载“exp01.out”；加载完毕后，单击“Run”运行程序；

实验结果：可见 XF 灯以一定频率闪烁；单击“Halt”暂停程序运行，则 XF 灯停止闪烁，如再单击“Run”，则“XF”灯又开始闪烁；

关闭所有窗口，本实验完毕。

源程序查看：用下拉菜单中 Project/Open，打开“Exp01.pjt”，双击“Source”，双击“exp01.asm”可查看源程序。

源程序注释如下：

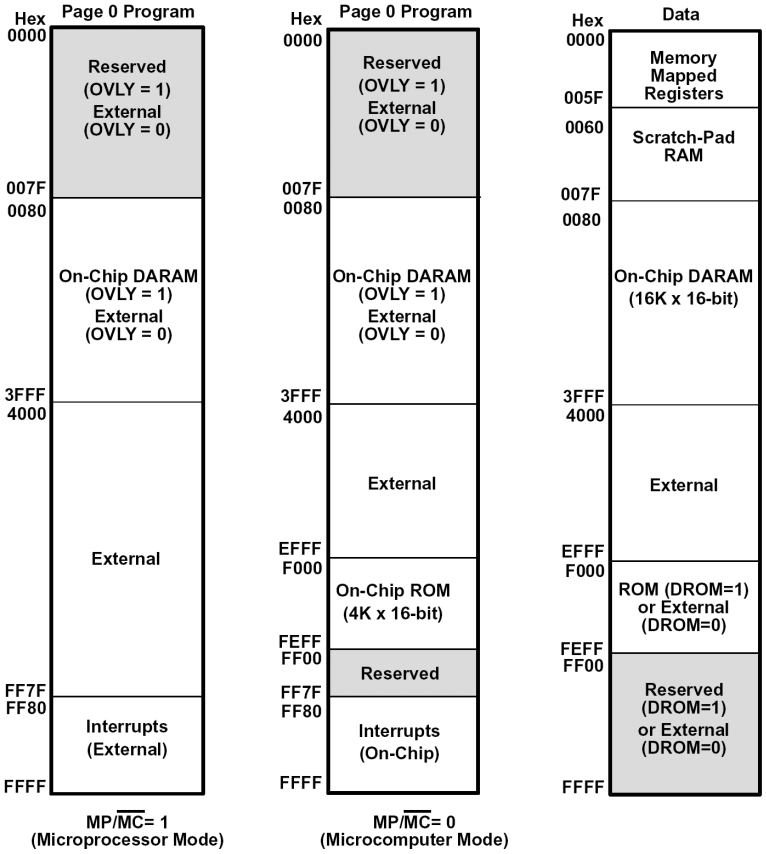
源程序：	注释：
;File Name:exp01.asm	
;the program is compiled at no autoinitialization mode	
.mmregs	定义存储器映像寄存器
.global _main	全局符号，可在外部定义
_main:	
stm #3000h,sp	;设置堆栈指针寄存器的值为 3000h
ssbx xf	;置位状态寄存器 xf
call delay	;调用 delay 函数
rsbx xf	;复位状态寄存器 xf
call delay	;调用 delay 函数
b _main	;无条件转移至_main
nop	;空指令
nop	;空指令
;delay .5 second	
delay:	
stm 270fh,ar3	;设置辅助寄存器 ar3 值为 9999
loop1:	
stm 0f9h,ar4	;设置辅助寄存器 ar4 值为 249
loop2:	
banz loop2,*ar4-	;寄存器 ar4 值减一，当其值不为 0 时跳转到 loop2
banz loop1,*ar3-	;寄存器 ar3 值减一，当其值不为 0 时跳转到 loop1
ret	;返回
nop	;空指令
nop	;空指令
;stm 2 cycles	
;banz when TRUE 4 cycles	
; FALSE 2 cycles	
;0f9h=>249d	
;270fh=>9999d	

.end	

（二）资料存储实验

1、实验使用资源

本实验指导书是以 TMS32OVC5410 为例，介绍相关的内部和外部内存资源。对于其它类型的 CPU 请参考查阅相关的资料手册。下面给出 TMS320VC5410 的内存分配表：



对于存储空间而言，映像表相对固定。值得注意的是内部寄存器与存储空间的映像关系。因此在编程应用时这些特定的空间不能作其它用途。对于程序存储空间而言，其映像表和 CPU 的工作模式有关。当 MP/MC 引脚为高电平时，CPU 工作在微处理器模式；当 MP/MC 引脚低电平时，CPU 工作在为计算机模式。具体的内存映像关系如上如所示。

内存实验主要了解内存的操作和 DSP 的内部双总线结构。并熟悉相关的指令代码和执行过程等。

2、实验过程

连接好 DSP 开发系统，运行 CCS 软件；

a) 在 CCS 的 Memory 窗口中查找 C5410 各个区段的数据存储器地址，在可以改变的存储器内容的地方,选定地址随意改变其中内容并观察结果；

b) 在 CCS 中装载实验示范程序，单步执行程序，程序中写入和读出的数据存储地址的变化；
c) 改变其它寻址方式，进行观察数据存储器地址与写入和读出数据的的变化。

本实验说明：

本实验程序将对 0x1000 开始的 8 个地址空间，填写入 0xAAAA 的数值，然后读出，并存储到 0X1008 开始的 8 个地址空间。在 CCS 中可以观察 DATA 内存空间地址 0X1000~0X100F 值的变化。

样例程序实验操作说明：

启动 CCS 2.0，并加载“exp02.out”，用“View”下拉菜单中的“Memory”查看内存单元，输入要查看的内存单元地址，本实验要查看 0x1000H~0x100FH 单元的数值变化，输入地址 0x1000H，查看 0x1000H~0x100FH 单元的初始值，单击“Run”运行程序，也可以“单步”运行程序，单击“Halt”暂停程序运行，查看 0x1000H~0x100FH 单元内数值的变化

关闭各窗口，本实验完毕。

源程序注释：

源程序：	注释：
*File Name:exp02.asm ;get some knowledge of the cmd file ;the program is compiled at no autoinitialization mode .mmregs .global _main _main: ;store data stm 1000h,ar1 rpt #07h st 0aaaah,*ar1+ ;read data then re-store stm 7h,ar3 stm 1000h,ar1 stm 1008h,ar2 loop: ld *ar1+,t st t,*ar2+ banz loop,*ar3- here: b here .end	 ;将外部内存地址 1000h 赋给 ar1 ;循环执行下一条指令 8 次 ;将 0aaaah 的值存储在 ar1 所对应内存中，且 ar1 值加 1 ;将 7h 赋给辅助寄存器 ar3 ;将地址 1000h 赋给辅助寄存器 ar1 ;将地址 1008h 赋给辅助寄存器 ar2 ;将辅助寄存器 ar1 的值赋给 t，且 ar1 值加 1 ;将 t 的值存储在 ar2 所对应内存中，且 ar2 值加 1 ;寄存器 ar3 值减 1，当其值不为 0 时跳转到 loop ;无条件转移至 here

（三）I/O 实验

1、 实验使用资源

数字量输入信号全部拓展出来，数字量输入接口主要由两个，D_Exp 与扳东开关连接，PX4 和 PX5 与电平转换芯片（74LVC245）连接，其功能分别为：

D_Exp—数字量输入扩展接口

1	2	3	4	5	6	7	8	9
I0	I1	I2	I3	I4	I5	I6	I7	VCC

电平转换扩展接口

	1	2	3	4
PX4—5V	IN0	IN1	IN2	IN3
PX5—3.3V	OUT3	OUT2	OUT1	OUT0

通过 PORTR，PORTW 指令可以实现 I/O 口的输入输出功能，如数字量采集实验。

实验说明：

实验中采用简单的一一映像关系来对 I/O 口进行验证，目的是使实验者能够对 I/O 有一目了然的认识。在本实验系统中，提供的 IO 空间分配如下：

CPU1:

0x0000 switch input (X) 8

0x0001 LED output(X) 8

CPU2:

0x0001 DAC

0x0004 Read_Key

0x0006 Write_Key

0x000F Write_LCD

0x8000 HPIC0

0x8001 HPIC1

0x8002 HPID0(AUTO)

0x8003 HPID1(AUTO)

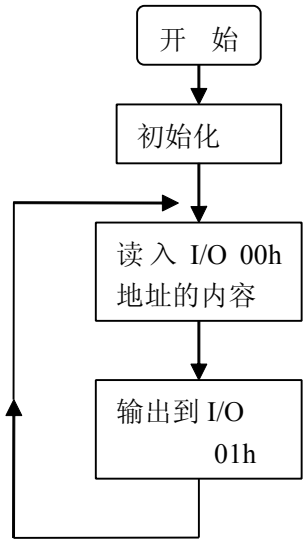
0x8004 HPIA0

0x8005 HPIA1

0x8006 HPID0(NO AUTO)

0x8007 HPID1(NO AUTO)

实验程序框图



注意： 电平转换接口主要考虑应用 3.3V 的中央处理器时，系统的电平兼容问题，用来保护 CPU 不受损坏。系统采用 74LVC245 电平兼容转换器件。

2、 实验过程

运行 CCS 程序，装载示范程序，调整 K0～K7 的开关，观察 LP1~LP7 LED 亮灭的变化，以及输入和输出状态是否一致。（**注意：输出为 0 时点亮灯**）

例程序实验操作说明

启动 CCS 2.0，并加载“exp03.out”，单击“Run”运行程序，任意调整 K0~K7 开关，可以观察到对应 LP0~LP7 灯“亮”或“灭”；单击“Halt”，暂停持续运行，开关将对灯失去控制。

关闭所有窗口，本实验完毕。

源程序查看：

用下拉菜单中 Project/Open，打开“Exp03.pjt”，双击“Source”，双击“exp03.asm”可查看源程序。

代码如下：

源程序：	注释：
;File Name :exp03.asm	
;learn how to operate the I/O ports	
;get some knowledge of the rts.lib file	
;in the I/O space 0x0000=>8 switches	
; 0x0001=>8 LEDs	
.mmregs	
.global _main	
.text	
_main:	
stm 3100h,sp	;设置堆栈指针寄存器的值为 3000h
stm 1000h,ar1	;设置辅助寄存器 ar1 值为 1000h
portr 00h,*ar1	;从 00h 端口读数据传入 ar1 所指向的内存空

	间，读按键
nop	;空指令
nop	;空指令
portw *ar1,01h	;将 ar1 所指向的内存空间的值赋给 01h 端口，控制 led 灯
nop	;空指令
nop	;空指令
b_main	;无条件转移至_main，实现按键控制
nop	;空指令
nop	;空指令
.end	

（四）定时器实验

1、实验使用资源

定时器实验时要用到 C54 芯片的定时器控制寄存器，定时器时间常数寄存器，定时器中断响应，寄存器定义详见 C54 芯片资料。C54 的定时器是一个 20 位的减法计数器，可以被特定的状态位实现停止、重新启动、重设置或禁止，可以使用该定时器产生周期性的 CPU 中断，控制定时器中断频率的两个寄存器是定时周期寄存器 PRD 和定时减法寄存器 TDDR

定时器实验通过 LED（LP1~LP7）来显示。在本系统中，时钟频率为 20MHZ，令 PRD = 0x4e1f，这样得到每 1/1000 秒中断一次，通过累计 1000 次，就能定时 1 秒钟。

2、实验过程

调入样例程序，装载并运行；

例程序实验操作说明

启动 CCS 2.0，并加载“exp04.out”，单击“Run”运行，可观察到 LED 灯（LP0~LP7）以一定的间隔时间不停摆动，单击“Halt”，暂停程序运行，LED 灯停止闪烁，单击“Run”，运行程序，LED 灯又开始闪烁。

关闭所有窗口，本实验完毕。

源程序：exp04.c	注释：

#include <stdio.h>
interrupt void timer();
/*extern void time();*/
extern void initial();
extern void porta();
extern void portb();
int flag=0;
interrupt void timer()
{ *(int *)0x300=*(int *)0x300+1
if(*(int *)0x300==0x3e8)
	;定时不断加 1
	;当定时器数值达到 0x3e8，开始定时器中断

```
    {
    *(int *)0x300=0;                    为定时器重新赋予初值
    *(int *)0x302=*(int *)0x302+1;
        if(flag==0)                     ;如果 flag 为 0，则将其变为 1，执行 porta（）
        {flag=1;
        porta();
        }
        else
        {flag=0;                        ;如果 flag 为 1，则将其变为 1，执行 portb（）
        portb();    }
    }
    return;    }
main()
{   initial();      初始化
    while(1){;} ; 无限循环   }
```

源程序：initial.asm

```
    .mmregs
    .global _initial
_initial:
    stm  300h,ar1          ;将 300h 赋值给 ar1
    st   #00h,*ar1         ;将 00h 赋值给内存地址为 300h 的空间里
    stm  302h,ar1          ;将 302h 赋值给 ar1
    st   #00h,*ar1         ;将 00h 赋值给内存地址为 302h 的空间里
    stm      200h,ar1
    st       #5555h,*ar1   ;为内存 200h 的内容赋予 5555h
    stm      201h,ar1
    st       #0aaaah,*ar1  ;为内存 201h 的内容赋予 aaaah
    stm      202h,ar1
    st       #400h,*ar1    ;为内存 202h 的内容赋予 400h
    ssbx 1,11              ;设置 ST1.INTM=1,停止所有的中断
    stm  0ffffh,ifr        ;清除所有中断的标志位，中断标志寄存器
                           （IFR）用来指明各个中断的目前状态。
    stm  00h,imr           ;  停止所有的中断 中断屏蔽寄存器（IMR）
                           在需要的时候独立地屏蔽特定的中断
    stm  410h,tcr          ;  停止定时器
    stm  4e1fh,prd         ; 设置定时器，  定时器周期计数器
    stm  420h,tcr          ;  打开定时器 定时器控制寄存器
    stm  08h,imr           ;允许定时器中断
    rsbx     1,11          ;设置 ST1.INTM=0,打开所有中断
    ret
```

源程序：port.asm	注释：
.mmregs	

.global _porta .global _portb _porta: stm 304h,ar1 st 5555h,*ar1 portw *ar1,01h ret _portb: stm 304h,ar1 st 0aaaah,*ar1 portw *ar1,01h ret	 ;将 304h 赋给辅助寄存器 ar1 ;将内存 304h 的内容赋予 5555h ;将内存 304h 的值写入 01h 端口 ;将内存 304h 的内容赋予 0aaaah ;将内存 304h 的值写入 01 端口

源程序：vectors.asm	注释：
.sect ".vectors" .ref _c_int00 .ref _timer .align 0x80 RESET: BD_c_int00 STM #200,SP stack size of 200 nmi: RETE NOP NOP NOP sint17 .space 4*16 sint18 .space 4*16 sint19 .space 4*16 sint20 .space 4*16 sint21 .space 4*16 sint22 .space 4*16 sint23 .space 4*16 sint24 .space 4*16 sint25 .space 4*16 sint26 .space 4*16	 引用函数 c_int00 引用了 c 中的函数 页边界排列 reset vector，复位中断响应 延迟分支到 C 主程序默认入口地址，c_int00 是 c 程序的入口，这里即进入 main 函数中开辟堆栈空间 中断屏蔽置为 0，响应中断，不可屏蔽中断产生时，使中断屏蔽取消，后返回。 保留出中断向量的地址空间

sint27 .space 4*16 sint28 .space 4*16 sint29 .space 4*16 sint30 .space 4*16	
int0: RETE NOP NOP NOP	中断寄存器设置 RETE 返回并允许中断外部中断产生时，直接返回。
int1: RETE NOP NOP NOP	
int2: RETE NOP NOP NOP	
tint: b _timer NOP NOP	定时器产生的时钟中断，返回到 c 中定义的 timer 程序，b 即是跳转
rint0: RETE NOP NOP NOP	同步串口 0（McBSP0）接受的中断，直接返回
xint0: RETE NOP NOP NOP	同步串口 0（McBSP0）发送的中断，直接返回
rint1: RETE NOP NOP NOP	同步串口 1（McBSP1）接受的中断，直接返回
xint1: RETE NOP NOP NOP	同步串口 1（McBSP1）发送的中断，直接返回
int3: RETE NOP NOP NOP .end	

（五）INT2 中断实验

1、实验使用资源

本实验是进行 C54 芯片的 INT2 中断练习， C54 芯片中断 INT2 是低电平单脉冲触发;实验采用导线一端连接 D_Exp—数字量输入扩展接口 I0,**经 PX4 的 IN3,到 PX5 的 OUT0 电平转换**,再与另一端连接 INT2 插孔；拨动开关 K0 一次，就产生一个低电平单脉冲；运行示范程序，观察 LP1~LP7 LED 灯的输出变化；可观察到每拨动开关 K0 一次 LP1~LP7 灯亮灭变化一次；

2、实验过程

样例程序实验操作说明

启动 CCS 2.0，并加载“exp05.out”，单击“Run”运行程序，反复拨动开关 K0，观察 LP1~LP7 LED 灯亮灭变化，单击“Halt”暂停程序运行，反复拨动开关 K0，LP1~LP7 LED 灯亮灭不变化；

关闭所有窗口，本实验完毕。

源程序查看：

用下拉菜单中 Project/Open，打开“Exp05.pjt”，双击“Source”，双击“int2.c”、“initial.asm”、“port.asm”以及“vectors.asm”可查看各源程序。

源程序：int2.c	注释：

interrupt void int2c();
extern void initial();
extern void porta();
extern void portb();
int flag=0;
main()
{
initial();
while(1){;}
}
interrupt void int2c()
{ asm("nop");
*(int *)0x300=*(int *)0x300+2
if(flag==0)
{
flag=1;
porta();
}
else
{
flag=0;
portb();
}
}
	; /*break here to show if interrupt happened*/这是一个记录外部中断的标志，通过记录 0x300h 中的值来记录

源程序：initial.asm	注释：
.mmregs .global _initial .text _initial: stm 300h,ar3 st #00h,*ar3 stm 302h,ar4 st #00h,*ar4 ssbx 1,11 stm 00h,imr stm 0ffffh,ifr stm 04h,imr rsbx 1,11 ret .end	 ;将 00h 存入地址为 300h 的内存中 ;将 00h 存入地址为 302h 的内存中 ;设置 ST1.INTM=1,停止所有的中断 ;停止所有的中断，中断屏蔽寄存器（IMR）在需要的时候独立地屏蔽特定的中断 ;清除所有中断的标志位，中断标志寄存器（IFR）用来指明各个中断的目前状态。 ;开启 int2 的外部中断 ;打开所有中断总开关

源程序：port.asm	注释：
.mmregs .global _porta .global _portb _porta: stm 304h,ar1 st 5555h,*ar1 portw *ar1,01h ret _portb: stm 304h,ar1 st 0aaaah,*ar1 portw *ar1,01h ret	 ;将 304h 赋给辅助寄存器 ar1 ;将内存 304h 的内容赋予 5555h ;将内存 304h 的值写入 01h 端口 ;将内存 304h 的内容赋予 0aaaah ;将内存 304h 的值写入 01 端口

源程序：vectors.asm	注释：
.sect ".vectors" .ref _c_int00 .ref _int2c .align 0x80 RESET: BD_c_int00	 引用函数 c_int00 页边界排列 reset vector，复位中断响应 延迟分支到 C 主程序默认入口地址，c_int00

STM #200,SP stack size of 200 nmi: RETE NOP NOP NOP sint17 .space 4*16 sint18 .space 4*16 sint19 .space 4*16 sint20 .space 4*16 sint21 .space 4*16 sint22 .space 4*16 sint23 .space 4*16 sint24 .space 4*16 sint25 .space 4*16 sint26 .space 4*16 sint27 .space 4*16 sint28 .space 4*16 sint29 .space 4*16 sint30 .space 4*16 int0: RETE NOP NOP NOP int1: RETE NOP NOP NOP int2: RETE NOP NOP NOP tint: b _timer NOP NOP rint0: RETE NOP NOP	是 c 程序的入口，这里即进入 main 函数中开辟堆栈空间 中断屏蔽置为 0，响应中断，不可屏蔽中断产生时，使中断屏蔽取消，后返回。 保留出中断向量的地址空间 中断寄存器设置 RETE 返回并允许中断外部中断产生时，直接返回。 定时器产生的时钟中断，返回到 c 中定义的 timer 程序，b 即是跳转 同步串口 0（McBSP0）接受的中断，直接返回

NOP	
xint0: RETE	同步串口 0（McBSP0）发送的中断，直接返回
NOP	
NOP	
NOP	
rint1: RETE	同步串口 1（McBSP1）接受的中断，直接返回
NOP	
NOP	
NOP	
xint1: RETE	同步串口 1（McBSP1）发送的中断，直接返回
NOP	
NOP	
NOP	
int3: RETE	
NOP	
NOP	
NOP	
.end	

实验二　A/D 采样实验

一、 实验目的

1. 掌握利用 TLV320AD50 实现 A / D 转换的技术基本原理和常用方法。
2. 学会 DSP 的多信道缓冲串口的应用方法。
3. 掌握并熟练使用 DSP 和 AD50 的接口及其操作。
4. 通过实验加深对 DSP 系统频谱混叠认识。

二、 实验设备

计算机，CCS 2.0 软件，DSP 仿真器，实验箱，示波器，连接导线。

三、 实验步骤和内容

1. 实验连线

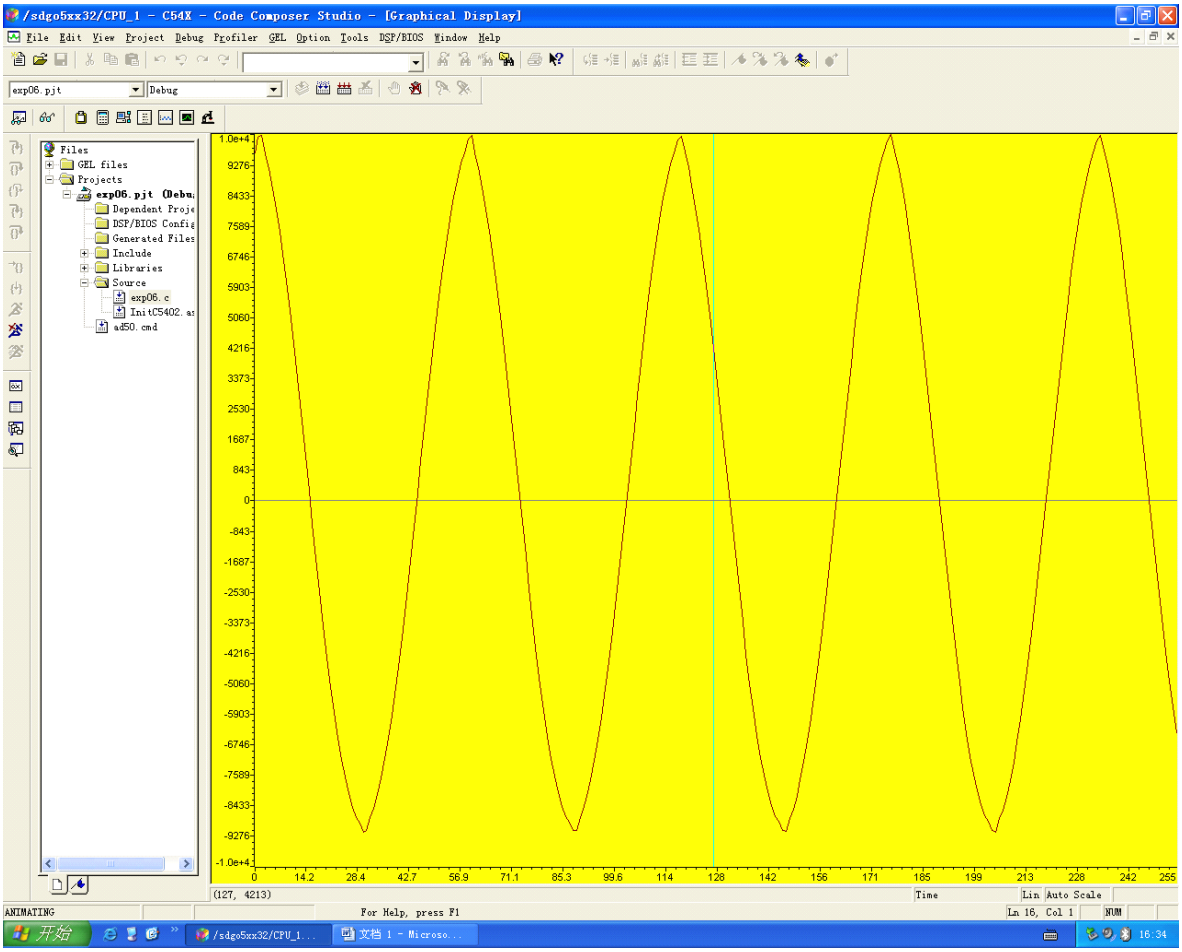
✧ 用短接块短接 SS1 的 1，2 脚，设置输出低频信号；短接 S2 的 Sin 脚，设置输出正弦波信号，这时模拟信号产生单元 SP1 输出为低频正弦波。

✧ JD 跳线断开，设置语音处理单元输入信号为交流；并用导线连接 SP1 脚和 JAD3 的 1 脚，将模拟低频正弦波信号接入语音处理单元。

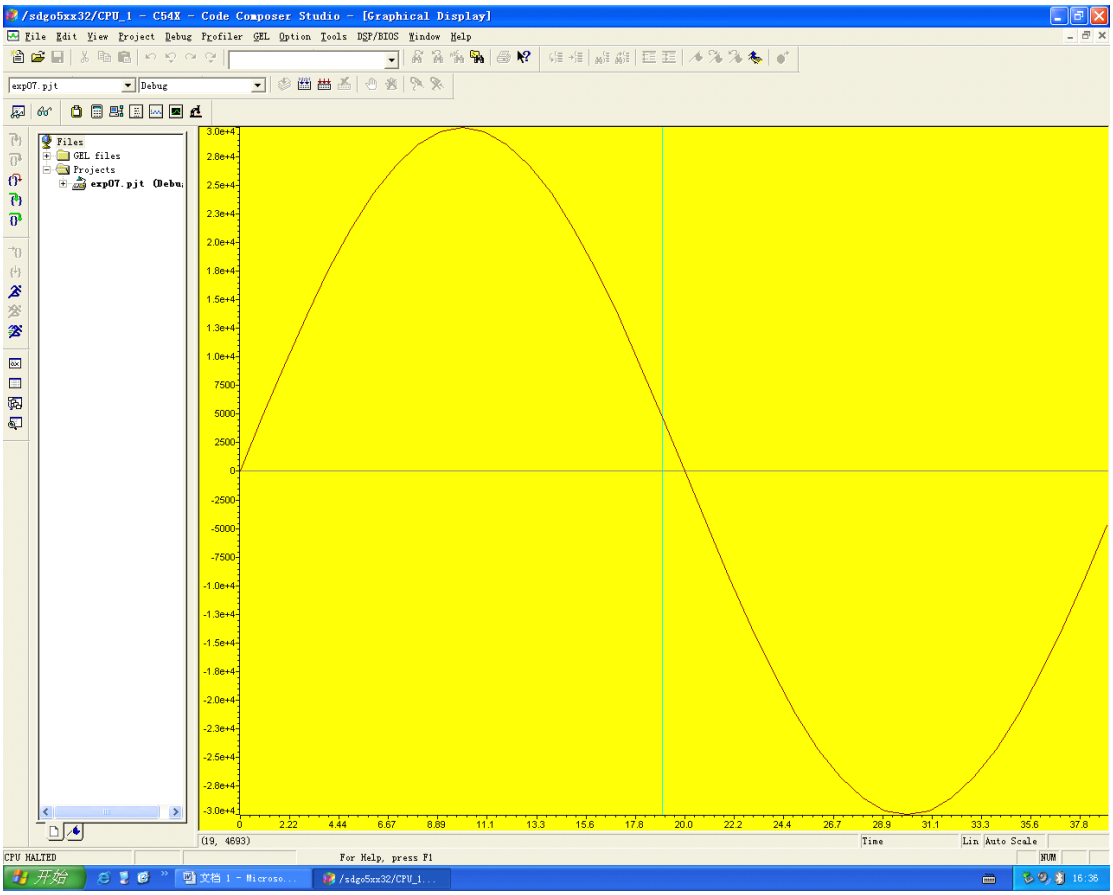
✧ 用导线连接 JAD1 的 INP 和 INPF，以及 JAD2 的 INM 和 INMF，将语音处理单元输出的差动模拟信号接入 AD50 输入端。

2. 运行 CCS 2.0 软件， 装入“exp06.pjt”工程文件，双击“exp06.pjt”及“Source”

3. 加载“exp06.out”示范程序，在“exp06.c”中“READAD50（）”处，设置断点，运行程序，通过用下拉菜单中的 View / Graph 的“Time/Frequency”打开一个图形观察窗口，调节输入信号的频率或幅值， 观察图形情况（幅值和频率），设置该图形观察窗口的参数，观察起始地址为 0x1000H，长度为 256 的内存单元内的数据，该资料为输入信号经 A/D 转换之后的数据，数据类型为 16 位整型，击“Animate”运行程序，在图形观察窗口观察 A/D 转换后的采样波形。


/sdgo5xx32/CPU_1 - C54X - Code Composer Studio - [Graphical Display]
File Edit View Project Debug Profiler GEL Option Tools DSP/BIOS Window Help
exp06.pjt
Debug
Files
GEL files
Projects
exp06.pjt (Debu
Dependent Proje
DSP/BIOS Config
Generated Files
Include
Libraries
Source
exp06.c
InitC5402.as
ad50.cmd
1.0e+4
9276
8433
7589
6746
5903
5060
4216
3373
2530
1687
843
0
-843
-1687
-2530
-3373
-4216
-5060
-5903
-6746
-7589
-8433
-9276
-1.0e+4
0
14.2
28.4
42.7
56.9
71.1
85.3
99.6
114
128
142
156
171
185
199
213
228
242
255
(127, 4213)
Time
Lin Auto Scale
ANIMATING
For Help, press F1
Ln 16, Col 1
NUM
开始
/sdgo5xx32/CPU_1...
文档 1 - Microso...
16:34

同样可以得到 D/A 实验的结果：



实验三：有限冲击响应滤波器（FIR）算法实验

一．实验目的

1．掌握用窗函数法设计 FIR 数字滤波器的原理和方法；
2．熟悉线性相位 FIR 数字滤波器特性；
3．了解各种窗函数对滤波特性的影响。

二．实验设备

计算机，CCS 2.0 版软件，实验箱，DSP 仿真器，短接块，导线。

三．实验原理

1．有限冲击响应数字滤波器的基础理论；
2．模拟滤波器原理（巴特沃斯滤波器、切比雪夫滤波器、贝塞尔滤波器）；
3．数字滤波器系数的确定方法。

四．实验步骤

1．复习如何设计 FIR 数字滤波。阅读本实验原理，掌握设计步骤；
2．阅读本实验所提供的样例子程序；
3．运行 CCS 软件，对样例程序进行跟踪，分析结果；
4．样例程序实验操作说明

1）实验前准备

在模拟信号产生单元中，一路信号源产生低频正弦波信号（S1 置“L”），

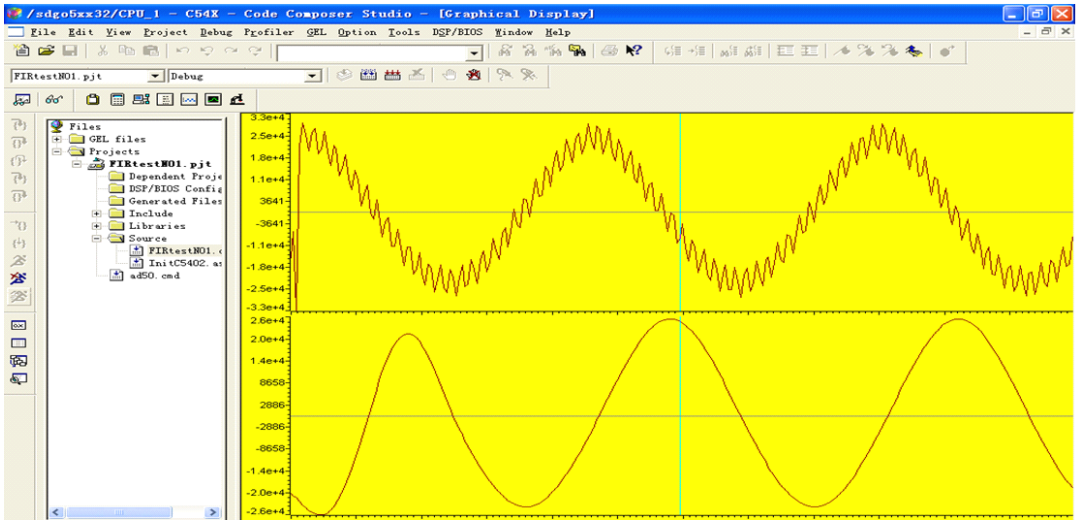
另一路信号源产生高频正弦波信号（S11 置“H”），检查模拟信号输出端口“A”与“B”应断开；实验箱上电，用示波器分别观测 out1 和 out2 输出的模拟信号，调节电位器 SPR1、SPR2（out1 输出信号的频率调节和幅值调节）和电位器 SPR11、SPR12（out2 输出信号的频率调节和幅值调节），直至满意为止；

本样例实验程序建议：低频正弦波信号为 100Hz/1V；
高频正弦波信号为 6KHz/1V；
实验箱掉电，做以下连接和检查；
短接输出端口“A”与“B”；
短接 JAD1 的 INM、INMF；
短接 JAD2 的 INP、INPF；
用导线连接 out2（模拟信号输出）和 JAD3 1 脚（MIC_IN）；
检查：JD 是否断开。

注：有关以上连接的说明，可参见第八章中语音接口跳线接输出接口配置使用，说明以及信号产生单元配置说明。正确完成计算机、DSP 仿真器和实验箱的连接后，系统上电。

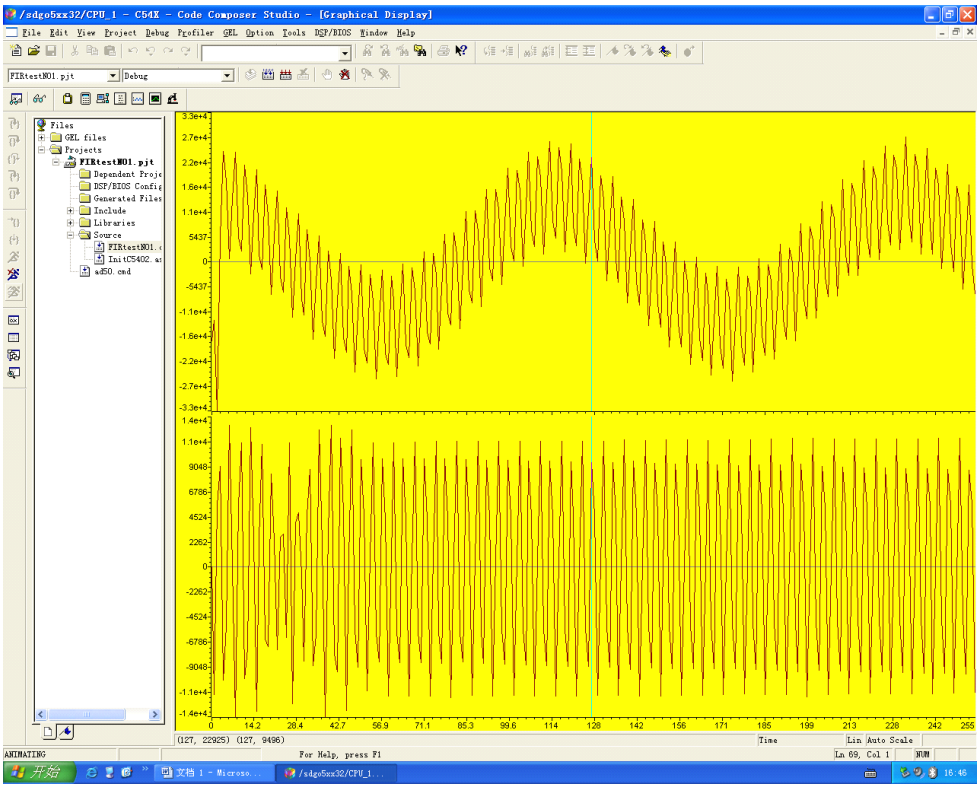
实验程序说明：该程序为 51 阶 FIR 低通滤波器算法程序，采用矩形窗函数实现，数组 h 和 xmid 长度均为 51，fs 为采样频率，fstop 为滤波器截止频率，可以修改以上参数来改变滤波器性能。重新“Rebuild All”后，并加载“Load”，单击“Animate”，可得到不同的结果；

实验结果：在 CCS2.0 环境，同步观察输入信号及其 FIR 低通滤波结果。实验滤波结果图入下图所示：



既然低通的有这么好的效果，那么我们就在低通的基础上修改为高通便可。

将语句中的 npass 修改为 1-npass 就可，这个关系和我们在 DSP 中设计滤波器时学的关系相对应，一个全通减去一个低通，那么不就是一个高通吗？我们设计的高通的结果图：



实验总结：

实验一是一些关于基本指令操作的例子，根据操作步骤一步一步进行就可以完成实验，其中可以通过单步运行指令查看每个语句的作用，同时也加深了对这门语言的理解。

实验二是A/D采样实验，通过此次实验，对A/D转换技术的基本原理和常用发放有了基本的了解，也加深了对DSP系统频谱混叠认识。

实验三是关于数字滤波器的实现，我了解了如何实现滤波器，且熟悉了滤波器的特性，掌握了滤波器的计算机仿真方法。数字信号处理的最主要应用领域是数字滤波。数字滤波器被认为是数字信号处理的另一块重要基石。这一部分的知识最让我感觉与课程的紧密相连，书本上的滤波器，到了现实中的效果和应用。

经过一段时间的DSP理论学习之后接触硬件实验，刚开始有点不知所措，感觉和书本上的东西关系不大，但是经过实验指导老师的耐心讲解，让我们明白了我们实验的意义何在，这样我们做起实验来，会有针对性地观察现象，重点的是什么？次要的是什么?这里面涉及到的知识很多，但是并不是我们都能够掌握的，如何才能够从这众多的信息中获取到最为有效的方案呢？这就需要我们拥有能够摘取有用信息的能力了。就像在看这些频域图的时候，我们也许并不清楚周围的那些功能是怎么用的，但我们看到波形头脑中应该能够和实际的低通高通联系在一起，知道该如何解释这个现象，这个现象代表了什么含义。虽然我们设计高通滤波器时不是最好的最快的，但是我们两个人一组在一起查阅资料，调试仪器，向别人虚心请教这个过程中也互相学习到了很多，我没有考虑到的地方，她会帮助我，而她想得不周全的地方我进行补充，这样才最终完成了实验。

相信对你有帮助的：
CCS 教程**(DSP** 开发软件**)**
DSP 软件与系统优化技术
数字和 **IP** 电话系统：**DSP** 软件代码

介绍 **dsp** 知识，为大家提供最新的 **dsp** 资讯，更多内容可以去南京研旭电气科技有限公司的官网 **www.njyxdq.com** **www.f28335.com** 或者官方论坛，嵌嵌 **dsp** 论坛 **www.armdsp.net** 进行交流学习

欢迎大家收听嵌嵌 **dsp** 论坛的官方微博

http://t.qq.com/qianqiandsp