

ICETEK-VC5509-C 评估板 使用说明书



北京瑞泰创新科技有限责任公司

地 址:北京市海淀区知春路 118 号知春大厦写字楼 A 座 1004 室

邮 编:100086

电 话:010-82671912/13/14/15

传 真:010-82671916

网 址:<http://www.realtimedsp.com.cn>

技术支持:welcome@realtimedsp.com.cn

目 录

第一部分 ICETEK – VC5509-C 评估板用户手册

一、TMS320VC5509 评估板实物图及扩展接口.....	3
二、TMS320VC5509 的存储空间和评估板的存储器映射.....	5
三、SDRAM 的特点和编程	9
四、Flash 的特点和编程	10
五、语音编解码芯片 TLV320AIC23 编程指南	13
六、标准串口 TL16C550 编程指南.....	20
七、板上 I/O 寄存器和使用.....	23

第二部分 ICETEK – VC5509-C 评估板实验手册

一、实验一：Code Composer Studio 入门实验	25
二、实验二：编制链接控制文件.....	31
三、实验三：数据存取实验.....	34
四、实验四：定点数除法实验.....	36
五、实验五：I/O 端口实验	41
六、实验六：语音编解码实验.....	43
七、实验七：自启动实验.....	45
八、实验八：异步串口通讯实验.....	49
九、实验九：卷积算法实验.....	54
十、实验十：有限冲击响应滤波器（FIR）算法实验	65
十一、实验十一：无限冲击响应滤波器（IIR）算法实验	68

第一章 ICETEK-VC5509-C 评估板用户手册

ICETEK-VC5509-C 评估板（下简称评估板或板）适用于针对 TMS320VC55x 系列数字信号处理器的软硬件开发工作。评估板为用户提供了—个完整且可以独立工作的 DSP 开发平台。

针对硬件部分的开发，评估板提供了详细的电路设计，可以用在—个全新的硬件项目的开发中。从使用评估板开始，用户能够迅速掌握整个 TMS320VC55x 系列数字信号处理器的电路设计，推荐的设计方案以及基本的外围设备的扩展等。此外，评估板还外扩了 TMS320C55x 的全部引脚，用户也可以在评估板上直接设计自己的用户电路。

从软件开发的角—度，评估板可以让用户完成从创建—个新的项目开始直到生成可独立运行的二进制用户程序的全部过程。同时，熟悉这个过程也可以加快用户自己的开发进度。

在这部分内容里，我们将介绍评估板的硬件接口，特别是对于 TMS320VC5509 芯片外的那些硬件设备，它们的编程和用法。

我们按照评估板的几大模块分别来说明板子上的接口，它们分别是 SDRAM、Flash、语音编解码芯片 TLV320AIC23、标准串口 TL16C550 和用于状态量输出的 4 位显示数码管和 4 位数字量输入的开关。针对每一个部分，我们都会有单独的一个小节来介绍他们。基于 TMS320VC5509 芯片的特殊结构，我们将在此之前首先介绍一下 TMS320VC5509 和评估板的存储器映射图，这是理解评估板扩展硬件资源的基础，也是为用户在设计自己硬件时需要了解的内容。

在这部分里，按照评估板使用的硬件资源，分为如下小节：

1. TMS320VC5509 评估板实物图及扩展接口
2. TMS320VC5509 的存储空间和评估板的存储器映射
3. SDRAM 的特点和编程
4. Flash 的特点和编程
5. 语音编解码芯片 TLV320AIC23 编程指南
6. 标准串口 TL16C550 编程指南
7. 板上 I/O 寄存器和使用

一、TMS320VC5509 评估板实物图及扩展接口

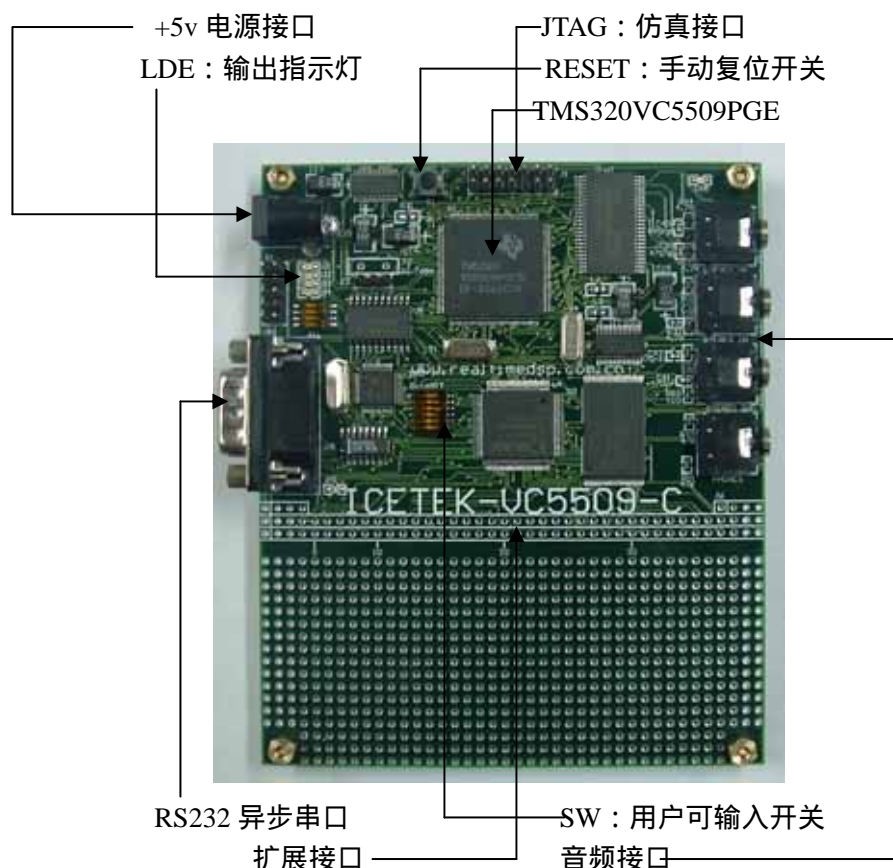


图 1-1-1 TMS320CV5509 评估板实物图

表 1-1-1 TMS320CV5509 评估板上接口说明

RESET	手动复位开关。
SW	用户可输入开关，软件可读。
LED	输出指示灯。
RS232 异步串口	标准异步串口，可以和计算机或其他设备通信。
JTAG	连接开发系统接口。
+5v 电源接口 (J1)	+5v 电源输入接口。
扩展接口	扩展总线输入输出接口。
TMS320CV5509 芯片	Dsp 主处理芯片。
音频接口	音频信号的输入输出

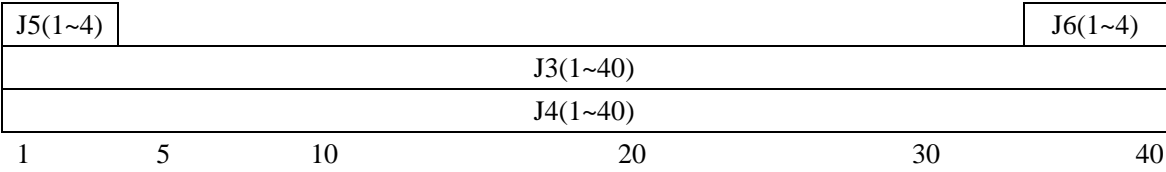


图 1-1-2 扩展接口示意图

表 1-1-2 扩展接口 J5 定义

Pin#	Signal
1	VCC
2	VCC
3	3.3V
4	3.3V

表 1-1-3 扩展接口 J6 定义

Pin#	Signal
1	GND
2	GND
3	AGND
4	AGND

表 1-1-4 扩展接口 J3 定义

Pin#	Signal	Pin#	Signal
1	GPIO3	2	GPIO4
3	GPIO6	4	TOUT
5	S10	6	S12
7	PU	8	DP
9	DN	10	USBVDD
11	GPIO7	12	GPIO2
13	GPIO1	14	GPIO0
15	X2	16	CLKOUT
17	ARE	18	AOE
19	AWE	20	ARDY
21	CE0	22	CE1
23	CE2	24	CE3
25	BE0	26	BE1
27	A13	28	A12
29	A11	30	A10

31	A9	32	A8
33	A7	34	A6
35	A5	36	A4
37	A3	38	A2
39	A1	40	A0

表 1-1-5 扩展接口 J4 定义

Pin#	Signal	Pin#	Signal
1	S11	2	S14
3	S15	4	S13
5	S20	6	S22
7	S21	8	S24
9	S25	10	S23
11	RTCLKIN	12	ADIN1
13	ADIN0	14	ADVDD
15	ADGND	16	INT4
17	INT3	18	INT2
19	INT1	20	INT0
21	DSPRST	22	SCL
23	SDA	24	XF
25	D0	26	D1
27	D2	28	D3
29	D4	30	D5
31	D6	32	D7
33	D8	34	D9
35	D10	36	D11
37	D12	38	D13
39	D14	40	D15

二． TMS320VC5509 的存储空间和评估板的存储器映射

TMS320VC5509 数字信号处理芯片(下简称为 5509)具有一个比较复杂的存储空间分配体系。因此,在使用之前,首先需要了解一下 5509 的存储空间体系。关于 5509 的存储空间的详细说明,请参考 TMS320VC5509 Datasheet 和 TMS320VC5509 DSP External Memory Interface (EMIF) (编号为 SPRU670) 用户手册。

TMS320VC5509 芯片具有两种不同的封装,它们的数据存储空间因而也有所不同。下面是两种封装在存储空间方面的简单比较。

表1-2-1 存储空间比较

封装形式	数据总线	地址总线
GHH	D15-D0	A20-A0
PGE	D15-D0	A13-A0

从上面的比较来说，GHH封装的芯片具有较多的地址线，因此，可以扩展更多的异步存储器或硬件设备，而PGE封装的芯片则相对来说地址线较少，因此可扩展的异步地址资源就小一些。在评估板上，我们采用了PGE封装的芯片。

5509的外部扩展空间信号线可以复用为通用输入输出脚（GPIO）或HPI信号线，所以，在使用存储空间时应该注意这些引脚的状态。下面的表格列出了TMS320VC5509PGE的引脚信号，及他们所有功能和配置。

表1-2-2 TMS320VC5509PGE的引脚定义

管脚名	硬件控制上电复位状态GPIO0； ESCR寄存器最低2位			
	复位时无此状态	GPIO0=1	复位时无此状态	GPIO0=0
	ESCR[1:0]=00	ESCR[1:0]=01	ESCR[1:0]=10	ESCR[1:0]=11
A[13:0]	GPIO.A[13:0]	EMIF.A[13:0]	HPI.HA[13:0]	HPI.HA[13:0] GPIO.A[13:0]
D[15:0]	EMIF.D[15:0]		HPI.HA[15:0]	
C0	EMIF.ARE		GPIO8	
C1	EMIF.AOE		HPI.HINT	
C2	EMIF.AWE		HPI.HR/W	
C3	EMIF.ARDY		HPI.HRDY	
C4	EMIF.CE0		GPIO9	
C5	EMIF.CE1		GPIO10	
C6	EMIF.CE2		HPI.HCNTL0	
C7	EMIF.CE3		GPIO11	HPI.HCNTL1
C8	EMIF.BE0		HPI.HBE0	
C9	EMIF.BE1		HPI.HBE1	
C10	EMIF.SDRAS		GPIO12	HPI.HAS
C11	EMIF.SDCAS		HPI.HCS	
C12	EMIF.SDWE		HPI.HDS1	
C13	EMIF.SDA10		GPIO13	
C14	EMIF.CLKMEM		HPI.HDS2	

图例：EMIF:存储空间，GPIO：通用输入输出引脚，HPI:主机并行接口。

圆点后的是信号的名称，详细信息，参考6609数据手册。5509的存储空间引脚信号的控制如下：

在上电复位时，GPIO0的电平决定这些信号的功能，此时

当 GPIO0电平= 高（逻辑“1”）时

上述引脚功能为EMIF,

ESCR[1:0] = 0

否则, 当 GPIO0电平= 低(逻辑“0”) 时

在上电复位结束之后, ESCR寄存器的最低2位ESCR[1:0]决定上述引脚的功能, 此时如上表所示。

5509的存储空间可以连接同步的SDRAM设备, 也可以连接异步的SRAM设备, 因此, 在使用这些设备之前, 应该首先注意对存储空间寄存器的配置。在接下来的说明中, 我们会介绍一些存储空间的配置寄存器。

5509的地址寻址也有比较特殊的地方, 从逻辑上说, 5509采用统一的编址方式, 即存储器的地址号没有重叠。但是, 存储器宽度分为两种不同的情况, 当存储器按照程序存储空间使用时, 地址编码采用字节寻址方式, 即每8位存储器占用一个地址编号, 此时, A0信号有效, 而按照数据存储空间使用时, 地址编码采用字寻址方式, 即每16位存储器占用一个地址编号, 此时, A0信号无效。因此按照不同的计算方法, 5509的存储器表示为:

表1-2-3 5509的存储器

程序存储空间	数据存储空间
16M字节, 16M*8bit	8M字 8M*16bit

下面是评估板的内存映射图, 红色部分是板子上有外扩硬件资源的地址空间, 其他部分来自于 TMS320VC5509 芯片的数据手册 (data sheet), 关于与芯片相关的地址空间的详细介绍, 请参考 TMS320VC5509 数据手册 (data sheet) 的相关说明。其中, 左边的地址按照程序存储空间编址, 而右边按照数据空间编址。此后所有的地址编码, 我们均按照数据存储空间的编码方式说明。**注意, 数据手册会不定期的修改, 除了红色部分, 其他地址如与数据手册有出入, 以数据手册的说明为准。我们推荐用户在使用评估板时, 下载最新的数据手册。**

表1-2-4 5509的地址译码

块大小字节	字节地址	存储器块	字地址	片外扩展
192	000000	存储器映射寄存器 (MMR) (保留)		
32K-192	0000C0	DARAM/HPI 访问		
32K	008000	DARAM	004000	
192K	010000	SARAM	008000	
16K异步存储器 4M-256K同步存储器	040000	外部扩展存储空间 (CE0)	002000	1M*16 SDRAM
16K异步存储器 4M同步存储器	400000	外部扩展存储空间 (CE1)	200000	512K*16位 Flash (分页访问)
16K异步存储器 4M同步存储器	800000	外部扩展存储空间 (CE2)	400000	评估板寄 存器组
			400004	保留

16K异步存储器 4M同步存储器	C00000			400200	串口寄存器组
				400208	保留
				400400	未用
		外部扩展存储空间 (CE3)		600000	
	32K	FF0000	ROM 当 MPNMC=0 时有效	外部扩展存储空间 (CE3) 当MPNMC=1 时有效	
16K	FF8000		ROM 当 MPNMC=0 时有效	外部扩展存储空间 (CE3) 当MPNMC=1 时有效	
16K	FFC000		SROM 当 MPNMC=0 SROM=0 时有效	外部扩展存储空间 (CE3) 当MPNMC=1 时有效	
FFFFFF					

每一个内存块给出首地址

DARAM：双存取RAM，分为8个8K的块，每个8K的块每周可以访问两次

SARAM：单存取RAM，分为24个8K的块，每个8K的块每周只能访问一次

外部扩展的存储空间由CE[3:0]分为4个部分，每个部分都可以支持同步或异步存储器类型

被减去的256K包含DARAM/HPI 访问32K,32K的DARAM存储器，192K的SARAM存储器

ROM: 每个块每次访问占用2个时钟周期，共有2个32K的块

此处扩展的存储器只有1M字节，但是片选信号CE0直接连在存储器上，因此CE0空间被完全占用，不能再外扩其他设备。此时低2M地址和高2M地址互为镜像

TMS320VC5509PGE只能最多外扩16K异步存储器，因此，要访问全部512K字节地址需要按照分页方式访问，具体配置参见本文件第3节

此部分保留给评估板，不能用来外扩其他设备

当使用CE2空间外扩设备时，必须保证设备放在A[13:10]=2以上的地址中，A[13:10]=2以下的地址被评估板使用

三．SDRAM 的特点和编程

5509 可以和标准的 SDRAM 接口，评估板上提供了 512K*2Blank*16 位的 SDRAM.在使用之前需要包含调试环境下的配置，请注意，如果没有正常配置，在访问 5509 片外时有可能出现仿真器死机现象，因此，使用前，最好使用 gel 下的文件初始化调试环境。使用方法如下：

执行 File->Load GEL,选中 gel 目录下的 C5509C.gel 文件即可。

评估板针对 SDRAM 的参数配置如下表：

表 1-3-1 SDRAM 的参数配置

评估板上与 SDRAM 有关的 5509 寄存器	SDRAM 配置
EBSR	0xa01
EGCR	0x220
CE01	0X3000
EMIRST	0
SDC1	0X5958
SDC2	0X38F
INIT	0

四．Flash 的特点和编程

评估板使用 AM29LV800 芯片作为外部 ROM 使用。外部 ROM 一般用来固化程序，上电后，可以利用 5509 芯片的 Bootloader 功能从 ROM 中加载程序到存储器中使用。此外，AM29LV800 芯片还可以在线编程，保存使用中需要保留的数据。

Flash 的最大特点是，在读操作中，类似普通的 ROM，在写操作中需要使用特殊的编程例程，且可以随时编程。下面是 Flash 的编程命令说明：

表 1-4-1 FLASH 编程命令说明

命令序列 (1)		周期		总线周期(2)(3)(4)(5)											
				第1步		第2步		第3步		第4步		第5步		第6步	
				地址	数据	地址	数据	地址	数据	地址	数据	地址	数据	地址	数据
读数据(6)		1		R A	R D	2AA									
复位(7)		1		X X X	F 0	555									
Autoselect (8)	厂商 I D	字	4	555	AA	2AA	55	555	90	X00	01				
		字节		AAA		555		AAA							
	设备 I D, 顶端启动块	字	4	555	AA	2AA	55	555	90	X01	22DA				
		字节		AAA		555		AAA		X02	DA				
	设备 I D, 项底启动块	字	4	555	AA	2AA	55	555	90	X01	225N				
		字节		AAA		555		AAA		X02	5B				
	扇区保护校验 (9)	字	4	555	AA	2AA	55	555	90	(SA) X04	XX00 XX01				
		字节		AAA		555		AAA		(SA) X04	00				
											01				
编程		字	4	555	AA	2AA	55	555	AD	PA	PD				
		字节		AAA		555		AAA							
Unlock 旁路		字	3	555	AA	2AA	55	555	20						
		字节		AAA		555		AAA							
Unl o c k 旁路编程(10)		2		XXX	A0	PA	PD								
Unl o c k 旁路复位(11)		2		XXX	90	XXX	00								
芯片擦除		字	6	555	AA	2AA	55	555	80	555	AA	2AA	55	555	10
		字节		AAA		555		AAA5		AAA		555		AAA	
扇区擦除		字	6	555	AA	2AA	55	555	80	555	AA	2AA	55	AAA	30
AAA		字节		AAA		555		AAA5		AAA		555		SA	
擦除 暂停(12)		1		XXX	B0										
擦除 继续(13)		1		XXX	30		10								

表例：

X：任意值，不用关心

RA：要读的地址单元号.

RD：RA中的内容

PA：要编程的地址单元号.地址值在最后一个WE信号或CE信号的下降沿被锁存。

PD：要编程到PA中的数据。数据值在最先一个WE信号或CE信号的上升沿被锁存。Data

SA：需要校验或擦除的扇区地址。高位地址线A18-A12决定了唯一的扇区号

注释

请查阅相应FLASH芯片数据手册的表1，了解芯片的总线操作。

所有的数据都是十六进制数。

除了reading array 或者autoselect操作数据之外，其他操作都是写操作。

在unlock操作和命令操作期间，数据位DQ15-DQ8为任意值，可以忽略。

除了SA操作和PA操作之外，在unlock和命令操作期间，地址线A18-A11为任意值，可以忽略。

读数据不需要unlock操作和命令操作。

当DQ5为“1”时，复位操作完成。

autoselect操作的第4步是读操作。

数据为“00”表示未保护扇区，“01”表示保护扇区。

unlock旁路操作之前需要unlock旁路编程操作。

在unlock旁路模式下，unlock旁路复位操作需要返回读取的数据。

在擦除暂停模式下，可以读取或编程没有执行擦除操作的扇区。擦除暂停操作只有在擦除模式下有效。

擦除继续操作只有在擦除暂停模式下有效。

关于 AM29LV800 芯片的其他信息，请看随板附带的数据手册。

TMS320VC5509PGE 只能最多外扩 16K 异步存储器，因此，要访问全部 512K 字节地址需要按照分页方式访问，这个访问可以通过控制 FAR 寄存器来实现。下面是 ROM 连接的示意图：

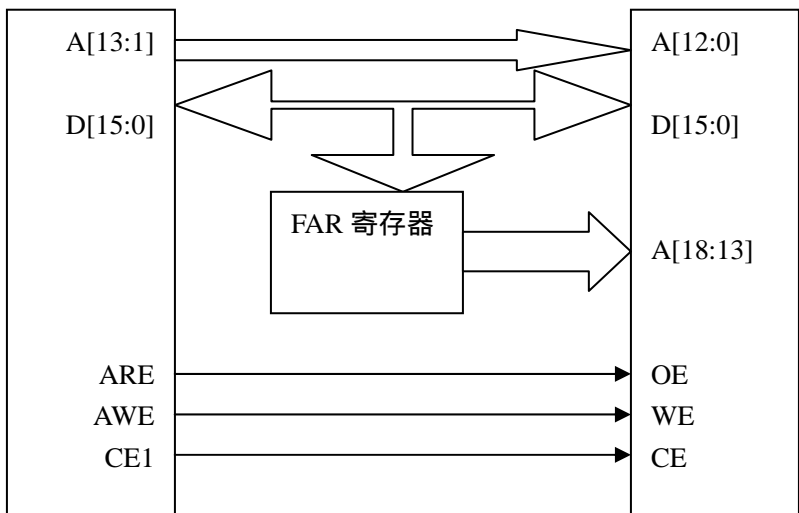


图 1-4-1 AM29LV800 连接图

其中 Flash 的高位地址线由 5509 的 FAR 控制寄存器控制，FAR 可以驱动与 Flash 的高位地址线处于一个固定的状态，从而实现分页的目的。上电复位时，FAR 寄存器的值被设置为 0，此时，所有的高位地址线处于低电平状态，5509 访问 Flash 的最低 16K 地址单元。此后随着复位的结束，用户程序开始工作，这样，就可以对 FAR 寄存器写值，改变 Flash 的高位地址，实现换页功能。下面是 FAR 寄存器的说明：

FAR 寄存器：地址是 0x400000

表 1-4-2 FAR 寄存器

7	6	5	4	3	2	1	0
保留	A18	A17	A16	A15	A14	A13	A13
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

表例：R 读允许，W：写允许，R/W:读写允许，-0：复位值，-x 没有固定值

表 1-4-3 FAR 寄存器说明

位	名称	说明
5-0	A[18:13]	每一位控制同名的 Flash 高位地址线，当其中某一位写“1”时，此位地址线为高，当其中某一位写“0”时，此位地址线为低。复位时所有位为“0”，选通 Flash 的最低 16K 地址单元。读这个寄存器可以知道最后一次写操作后各个高位地址线状态。此寄存器与 Flash 的地址映射关系如下： 访问的 Flash 的地址单元= (FAR<<12)左移 12 位)+DSP 地址线 A[13:1]

7-6 保留 读写状态位无效

在随板的软件中给出了对 Flash 进行操作的程序。

五．语音编解码芯片 TLV320AIC23 编程指南

TLV320AIC23 是一个高性能的多媒体数字语音编解码器，它的内部 ADC 和 DAC 转换模块带有完整的数字滤波器。(digital interpolation filters) 数据传输宽度可以是 16 位，20 位，24 位和 32 位，采样频率范围支持从 8khz 到 96khz。在 ADC 采集达到 96khz 时噪音为 90-dBA，能够高保真的保存音频信号。在 DAC 转换达到 96khz 时噪音为 100-DbA，能够高品质的数字回放音频，在回放时仅仅减少 23 mW。

TLV320AIC23 详细指标：

- 高品质的立体声多媒体数字语音编解码器
 - 在 ADC 采用 48 kHz 采样率时噪音 90-dB
 - 在 DAC 采用 48 kHz 采样率时噪音 100-dB
 - 1.42 V ~ 3.6 V 核心数字电压：兼容 TI C54x DSP 内核电压
 - 2.7 V ~3.6 V 缓冲器和模拟：兼容 TI C54x DSP 内核电压
 - 支持 8kHz ~96kHz 的采样频率
- 软件控制通过 TI McBSP接口
- 音频数据输入输出通过 TI McBSP接口

立体声接口图示：

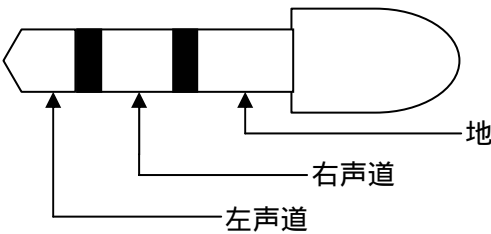


表 1-5-1 TLV320AIC23 的 映射寄存器定义：

ADDRESS	REGISTER
0000000	左声道输入控制
0000001	右声道输入控制
0000010	左耳机通道控制
0000011	右耳机通道控制
0000100	模拟音频通道控制

北京瑞泰创新科技有限责任公司---- ICETEK-VC5509-C 评估板使用说明书

0000101	数字音频通道控制
0000110	启动控制
0000111	数字音频格式
0001000	样本速度控制
0001001	数字界面激活
0001111	初始化寄存器

表1-5-2 左声道输入控制(Address: 0000000)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	LRS	LIM	X	X	LIV4	LIV3	LIV2	LIV1	LIV0
Default	0	1	0	0	1	0	1	1	1

LRS : 左右声道同时更新 0 = 禁止 1 = 激活

LIM : 左声道输入衰减 0 = Normal 1 = Muted

LIV[4:0] : 左声道输入控制衰减 (10111 = 0 dB 缺省)

最大 11111 = +12 dB 最小 00000 = -34.5 dB

表1-5-3 右声道输入控制(Address: 0000001)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	RLS	RIM	X	X	RIV4	RIV3	RIV2	RIV1	RIV0
Default	0	1	0	0	1	0	1	1	1

LRS : 左右声道同时更新 0 = 禁止 1 = 激活

LIM : 右声道输入衰减 0 = Normal 1 = Muted

LIV[4:0] : 右声道输入控制衰减 (10111 = 0 dB 缺省)

最大 11111 = +12 dB 最小 00000 = -34.5 dB

表 1-5-4 左耳机通道控制 (Address: 0000010)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	LRS	LZC	LHV6	LHV5	LHV4	LHV3	LHV2	LHV1	LHV0
Default	0	1	1	1	1	1	0	0	1

LRS : 左右耳机通道控制 0 = 禁止 1 = 激活

LZC : 0 点检查 0 = Off 1 = On

LHV[6:0] : 左耳机通道控制音量衰减(1111001 = 0 dB default)

最大 1111111 = +6 dB

最小 0110000 = -73 dB (mute)

1-5-5 右耳机通道控制(Address: 0000011)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	RLS	RZC	RHV6	RHV5	RHV4	RHV3	RHV2	RHV1	RHV0
Default	0	1	1	1	1	1	0	0	1

LRS : 左右耳机通道控制 0 = 禁止 1 = 激活

LZC : 0 点检查 0 = Off 1 = On

LHV[6:0] : 右耳机通道控制音量衰减(1111001 = 0 dB default)

最大 1111111 = +6 dB

最小 0110000 = -73 dB (mute)

表 1-5-6 模拟音频通道控制 (Address: 0000100)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	X	STA1	STA0	STE	DAC	BYP	INSEL	MICM	MICB
Default	0	0	0	0	1	1	0	1	0
STA[1:0]	侧音衰减		00 = -6 dB		01 = -9 dB		10 = -12 dB		11 = -15 dB
STE	侧音激活		0 = 禁止		1 = 激活				
DAC	DAC 选择		0 = DAC 关闭		1 = DAC 选择				
BYP	旁路		0 = 禁止		1 = 激活				
INSEL	模拟输入选择		0 = 线路		1 = 麦克风				
MICM	麦克风衰减		0 = 普通		1 = 衰减				
MICB	麦克风增益		0 = 0dB		1 = 20dB				

表 1-5-7 数字音频通道控制 (Address: 0000101)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	X	X	X	X	X	DACM	DEEMP1	DEEMP0	ADCHP
Default	0	0	0	0	0	0	1	0	0

DACM : DAC 软件衰减 0 = 禁止 1 = 激活

DEEMP[1:0] : De-emphasis 控制 00 = 禁止 01 = 32 kHz 10 = 44.1 kHz 11 = 48 Khz

ADCHP : ADC 滤波器 0 = 禁止 1 = 激活

X : 保留

表 1-5-8 启动控制 (Address: 0000110)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	X	OFF	CLK	OSC	OUT	DAC	ADC	MIC	LINE
Default	0	0	0	0	0	0	1	1	1
OFF	设备电源			0= On			1=OFF		
CLK	时钟			0= On			1=OFF		
OSC	振荡器			0= On			1=OFF		
OUT	输出			0= On			1=OFF		
DAC	DAC			0= On			1=OFF		
ADC	ADC			0= On			1=OFF		
MIC	麦克风输入			0= On			1=OFF		
LINE	Line 输入			0= On			1=OFF		

表1-5-9 数字音频格式(Address: 0000111)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	X	X	MS	LRSWAP	LRP	IWL1	IWL0	FOR1	FOR0
Default	0	0	0	0	0	0	0	0	1

MS：主从选择 0 = 从模式 1 = 主模式

LRSWAP：DAC 左/右通道交换 0 = 禁止 1 = 激活

LRP：DAC 左/右通道设定 0 = 右通道在 LRCIN 高电平

1 = 右通道在 LRCIN 低电平

IWL[1:0]：输入长度 00 = 16 bit 01 = 20 bit 10 = 24 bit 11 = 32 bit

FOR[1:0]：数据初始化 11 = DSP 初始化, 帧同步来自于两个字

10 = I²S 初始化,

01 = MSB 优先, 左声道排列 00 = MSB 优先, 右声道排列

X：保留

表 1-5-10 样本速度控制(Address: 0001000)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	X	CLKOUT	CLKIN	SR3	SR2	SR1	SR0	BOSR	USB/Normal
Default	0	0	0	1	0	0	0	0	0
CLKIN		时钟输入分割			0 = MCLK				1 = MCLK/2
CLKOUT		时钟输出分割			0 = MCLK				1 = MCLK/2
SR[3:0]		样本速度控制							
BOSR		基础速度比率							
		USB 模式:			0= 250 f _s			1= 272 f _s	

	普通模式:	0= 250 f	1= 384 f _s
USB/Normal	时钟模式选择	0= 普通	1= USB
X	保留		

数字界面激活 (Address: 0001001)

表1-5-11 数字界面激活(Address: 0001001)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	X	X	X	X	X	X	X	X	ACT
Default	0	0	0	0	0	0	0	0	0

ACT : 激活控制 0 = 停止 1 = 激活

X : 保留

表1-5-12 初始化寄存器(Address: 0001111)

BIT	D8	D7	D6	D5	D4	D3	D2	D1	D0
Function	RES	RES	RES	RES	RES	RES	RES	RES	RES
Default	0	0	0	0	0	0	0	0	0

RES : 写 000000000 到这个寄存器引发初始化

关于 TLV320AIC23 的详细信息, 请参考随板软件中的数据手册。下图是 5509 与 TLV320AIC23 的连接示意图。关于 5509 对 TLV320AIC23 的编程, 请参考随后的实验。

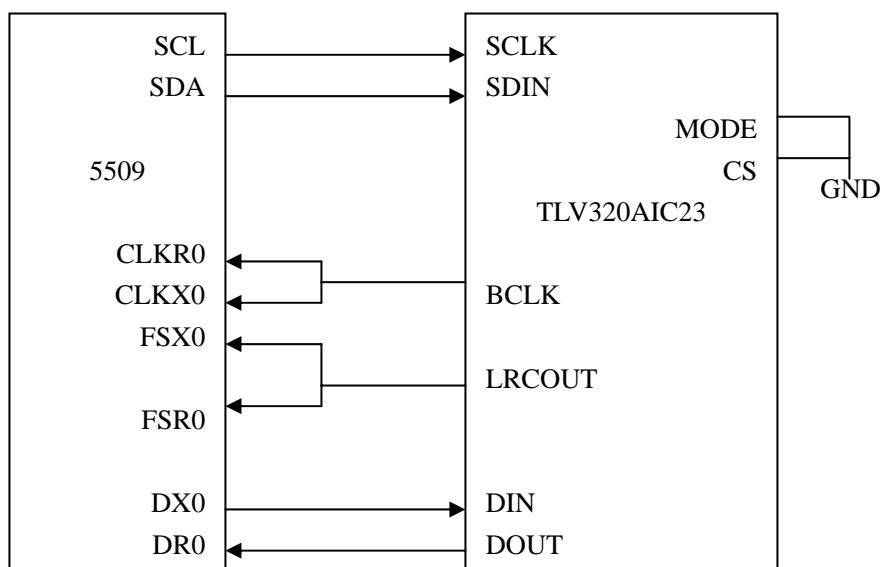


图 1-5-1 5509 与 TLV320AIC23 的连接示意图

AIC23 芯片定义：

// AIC23 Control Register addresses

```
#define AIC23_LT_LINE_CTL      0x00  // 0
#define AIC23_RT_LINE_CTL      0x02  // 1
#define AIC23_LT_HP_CTL        0x04  // 2
#define AIC23_RT_HP_CTL        0x06  // 3
#define AIC23_ANALOG_AUDIO_CTL  0x08  // 4
#define AIC23_DIGITAL_AUDIO_CTL 0x0A  // 5
#define AIC23_POWER_DOWN_CTL    0x0C  // 6
#define AIC23_DIGITAL_IF_FORMAT 0x0E  // 7
#define AIC23_SAMPLE_RATE_CTL   0x10  // 8
#define AIC23_DIG_IF_ACTIVATE    0x12  // 9
#define AIC23_RESET_REG         0x1E  // F - Writing 0 to this reg triggers reset
```

// AIC23 Control Register settings

```
#define lt_ch_vol_ctrl      0x0017 /* 0 */
#define rt_ch_vol_ctrl      0x0017 /* 1 */
#define lt_ch_headph_ctrl   0x0079 /* 2 */
#define rt_ch_headph_ctrl   0x0079 /* 3 */
#define alog_au_path_ctrl    0x0000 /* 4 */
#define digi_au_path_ctrl    0x0000 /* 5 */
#define pow_mgt_ctrl_ctrl    0x0002 /* 6 */
#define digi_au_intf_ctrl    0x000D /* 7 */
#define au_FS_TIM_ctrl       0x0000 /* 8 MCLK=12MHz, Sample Rate setting */
#define digi_intf1_ctrl      0x0001 /* 9 */
#define digi_intf2_ctrl      0x00FF /* 10 */
```

```
#define DIGIF_FMT_MS         0x40
#define DIGIF_FMT_LRSWAP     0x20
#define DIGIF_FMT_LRP        0x10
#define DIGIF_FMT_IWL        0x0c
#define DIGIF_FMT_FOR        0x03
```

```
#define DIGIF_FMT_IWL_16     0x00
#define DIGIF_FMT_IWL_20     0x04
#define DIGIF_FMT_IWL_24     0x08
#define DIGIF_FMT_IWL_32     0xc0
```

```
#define DIGIF_FMT_FOR_MSBRIGHT    0x00
#define DIGIF_FMT_FOR_MSLEFT      0x01
#define DIGIF_FMT_FOR_I2S         0x02
#define DIGIF_FMT_FOR_DSP         0x03

#define POWER_DEV                  0x80
#define POWER_CLK                  0x40
#define POWER_OSC                  0x20
#define POWER_OUT                  0x10
#define POWER_DAC                  0x08
#define POWER_ADC                  0x04
#define POWER_MIC                  0x02
#define POWER_LINE                 0x01

#define SRC_CLKOUT                 0x80
#define SRC_CLKIN                  0x40
#define SRC_SR                     0x3c
#define SRC_BOSR                   0x02
#define SRC_MO                     0x01

#define SRC_SR_44                  0x20
#define SRC_SR_32                  0x18

#define ANAPCTL_STA                0xc0
#define ANAPCTL_STE                0x20
#define ANAPCTL_DAC                0x10
#define ANAPCTL_BYP               0x08
#define ANAPCTL_INSEL              0x04
#define ANAPCTL_MICM               0x02
#define ANAPCTL_MICB              0x01

#define DIGPCTL_DACM               0x08
#define DIGPCTL_DEEMP              0x06
#define DIGPCTL_ADCHP              0x01
#define DIGPCTL_DEEMP_DIS          0x00
#define DIGPCTL_DEEMP_32           0x02
#define DIGPCTL_DEEMP_44           0x04
#define DIGPCRL_DEEMP_48           0x06
```

```
#define DIGIFACT_ACT      0x01
#define LT_HP_CTL_LZC     0x80
#define RT_HP_CTL_RZC     0x80
```

六．16C550 异步串口使用说明。

16C550 有 11 个寄存器，通过 A2~A0 和线路控制寄存器中的 DLAB 位对它们进行寻址。

16C550 的寄存器如下表所示：

表 1-6-1 16C550 的寄存器

寄存器	DLAB	A2	A1	A0	地址	操作
接收缓冲寄存器 RBR	0	0	0	0	00H	只读
发送缓冲寄存器 THR	0	0	0	0	00H	只写
中断始能寄存器 IER	0	0	0	1	01H	读/写
中断标志寄存器 IIR	X	0	1	0	01H	只读
FIFO 控制寄存器 FCR	X	0	1	0	01H	只读
线路控制寄存器 LCR	X	0	1	1	03H	读/写
MODEM 控制寄存器 MCR	X	1	0	0	04H	读/写
线路状态寄存器 LSR	X	1	0	1	05H	读/写
MODEM 状态寄存器 MSR	X	1	1	0	06H	读/写
暂存寄存器 SCR	X	1	1	1	07H	读/写
低位除数寄存器 DLL	1	0	0	0	00H	读/写
低位除数寄存器 DLM	1	0	0	1	01H	读/写

1．线路控制寄存器：寄存器地址（port2003）

表 1-6-2 线路控制寄存器

D7	D6	D5	D4	D3	D2	D1	D0
DLAB	BREAK	SPB	EPS	PEN	STB	WLS1	WLS0

WLS1 WLS0： 设置数据长度：

0： 5 位

1： 6 位

0： 7 位

1： 8 位

STB： 设置停止位个数

0： 一个停止位

1： 1.5 个停止位(5 位数据长度时)，2 个停止位(6，7，8 位数据长度时)

PEN： 奇偶校验使能

- 0 : 奇偶校验无效
 1 : 奇偶校验有效
 EPS : 奇偶校验选择
 0 : 奇校验
 1 : 偶校验
 DLAB : 寄存器访问选择
 0 : 访问其余寄存器
 1 : 访问除数和功能切换寄存器

2. 线路状态寄存器：寄存器地址（port2005）

表 1-6-3 线路状态寄存器

D7	D6	D5	D4	D3	D2	D1	D0
FERR	TEMT	THRE	BI	FE	PE	OE	DR

- DR : 接收数据准备好标志
 0 : 接收数据缓冲器空
 1 : 接收数据缓冲器中有数据
 OE : 溢出错误标志(上一个接收数据被当前接收数据覆盖)
 0 : 无溢出
 1 : 有溢出
 PE : 奇偶校验错误标志
 0 : 无奇偶校验错误
 1 : 有奇偶校验错误
 THRE : 发送保持寄存器空标志
 0 : 非空
 1 : 空
 TEMT : 发送器空标志
 0 : 发送保持寄存器和发送移位寄存器非空
 1 : 发送保持寄存器和发送移位寄存器都空

3. 中断使能寄存器：寄存器地址（port2001）

表 1-6-4 中断使能寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	EMSI	ELSI	ETHREI	ERDAI

ERDAI : 接收中断使能

0：接收中断禁止
 0：接收中断使能
 ETHREI：发送中断使能
 0：接收中断禁止
 1：接收中断使能
 ELSI：接收错误中断使能
 0：接收错误中断禁止
 1：接收错误中断使能
 EMSI：MODEM 中断使能
 0：MODEM 中断禁止
 1：MODEM 中断使能

4. 中断标志寄存器：寄存器地址（port2002）

表 1-6-5 中断标志寄存器

中断标志寄存器				中断设置与清除			
D3	D2	D1	D0	优先级	中断类型	中断源	中断清除
0	0	0	1	---	无中断	无中断	
0	1	1	0	最高	接收错误	溢出，奇偶，帧错误	读线路状态寄存器
0	1	0	0	第二	接收	接收缓冲器中数据	读接受缓冲器
1	1	0	0	第二	FIFO 超时	FIFO 超时	读接收缓冲器
0	0	1	0	第三	发送	发送保持寄存器空	写发送保持寄存器
0	0	0	0	第四	MODEM	MODEM 状态	读状态寄存器

5. 设置波特率

16C550 的波特率可通过除数寄存器 DLM，DLL 来设置，除数寄存器值和波特率之间的换算公式如下：除数值=输入频率÷(波特率×16)，16C550 的输入频率为：3.6864MHz，波特率和除数之间的关系如表所示：

表 1-6-6 波特率和除数之间的关系

波特率	高位除数寄存器 DLM	低位除数寄存器 DLL
1200	00H	C0H
2400	00H	60H
4800	00H	30H
9600	00H	18H
19200	00H	0CH

38400

00H

06H

6. 串口标准

RS232 标准

七、板上 I/O 寄存器和使用

ICETEK-VC5509-C 评估板提供了四位的状态显示数码管、四位可读入数字量的开关和控制 GPIO 通断的控制寄存器。这些接口不使用 TMS320VC5509 的芯片管脚实现。下面是这三个寄存器的列表：

表 1-7-1 I/O 寄存器地址

外扩寄存器地址	寄存器名称	有效位数	读写状态	上电复位状态
0x400001	LEDR	D3 到 D0 有效	读/写允许	1111b
0x400002	SWR	D3 到 D0 有效	读允许	-----
0x400003	GPIOER	D0 有效	读/写允许	1b

下面是显示数码管、四位可读入数字量的开关寄存器和控制 GPIO 通断的控制寄存器的说明：

1. LEDR 寄存器：地址是 0x400001

表 1-7-2 LEDR 寄存器

7	4	3			0
无效位		LEDR3	LEDR2	LEDR1	LEDR0
		R/W-0	R/W-0	R/W-0	R/W-0

图例：R 读允许，W：写允许，R/W:读写允许，-0：复位值,-x 没有固定值

位	名称	说明
3	LEDR3	四位显示数码管。向某位写“0”，点亮相应的发光管，写“1”，则使数码管熄灭。这个寄存器可读，在使用时可以通过读操作得到最后一次写操作的值。
2	LEDR2	
1	LEDR1	
0	LEDR0	

2. SWR 寄存器: 地址是 0x400002

表 1-7-3 SWR 寄存器

7	4	3			0
无效位		SWR3	SWR2	SWR1	SWR0
		R-x	R-x	R-x	R-x

图例：R 读允许，W：写允许，R/W:读写允许，-0：复位值,-x 没有固定值

位	名称	说明
---	----	----

3	SWR3	四位开关。当开关连通（处于 ON 一侧）时，寄存器读入“0”，当开关断开（不处于 ON 一侧）时，寄存器读入“1”。寄存器可读，在使用时可以通过读操作得到四个开关的当前状态。
2	SWR2	
1	SWR1	
0	SWR0	

3. 控制 GPIO 通断的控制寄存器

由于 5509 芯片管脚 GPIO0-3 在上电复位时有配置芯片的作用，因此，我们设计这个寄存器，当上电复位时，这个寄存器打开，此时，由配置开关 U10 上的高低电平就可以影响 DSP 的上电初始化，达到配置 DSP 工作的目的。而结束复位之后，我们还可以关闭这个寄存器，使得 GPIO0-3 可以自由的使用。下面是硬件的示意图：

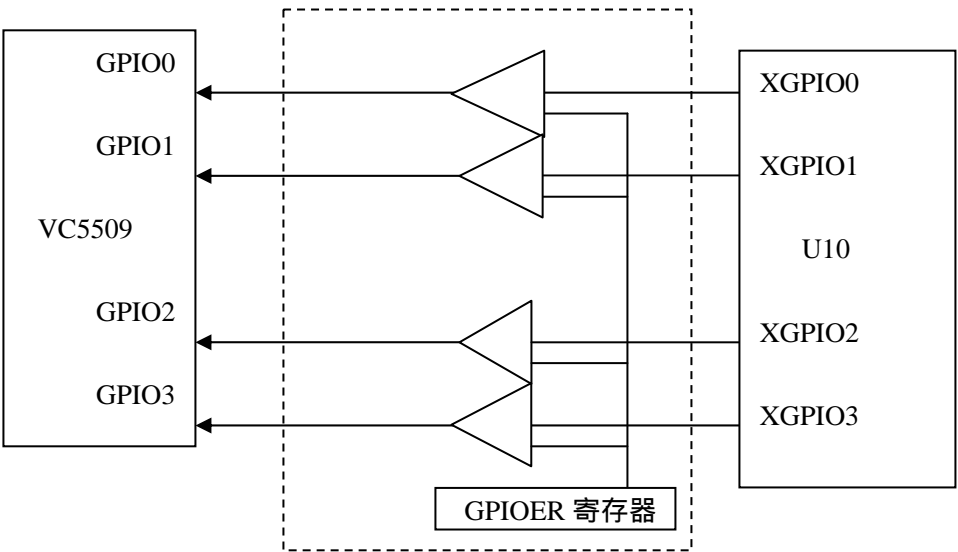


图 1-7-1 硬件连接图

GPIOER 寄存器:地址是 0x400003

表 1-7-4 GPIOER 寄存器

7	1	0
保留		SPIOE

图例：R 读允许，W：写允许，R/W:读写允许，-0：复位值,-x 没有固定值

R/W-1

位	名称	说明
0	GPIOE	这一位控制 DSP 的 GPIO0-3 引脚是否与 U10 的 4 位开关连通。当此位置“1”时各个引脚连通，当此位置“0”时各个引脚断开，此时，DSP 的 GPIO0-3 接口引脚上为高阻态，可以连接其他设备。上电复位时，此位为“1”。

第二章 ICETEK – VC5509-C 评估板实验指导

实验一：Code Composer Studio 入门实验

一. 实验目的

1. 掌握 Code Composer Studio 2.0 的安装和配置。
2. 了解 DSP 开发系统和计算机与目标系统的连接方法。
3. 了解 Code Composer Studio 2.0 软件的操作环境和基本功能，了解 TMS320C5xxx 软件开发过程。
 - . 学习创建工程和管理工程的方法。
 - . 了解基本的编译和调试功能。
 - . 学习使用观察窗口。
 - . 了解图形功能的使用。

二. 实验设备

1. PC 兼容机一台；操作系统为 Windows2000 (或 WindowsNT、Windows98、WindowsXP，以下假定操作系统为 Windows2000)。Windows 操作系统的内核如果是 NT 的应安装相应的补丁程序（如：Windows2000 为 Service Pack3，WindowsXP 为 Service Pack1）。
2. 配备 ICETEK-ICETEK-USB 仿真器或 ICETEK-ICETEK-PP 仿真器和 ICETEK-VC5509-C 评估板，5V 电源一只。
3. USB 连接电缆一条(如使用 PP 型仿真器换成并口电缆一条)。

三. 实验原理

开发 TMS320C5xxx 应用系统一般需要以下几个调试工具来完成：

.软件集成开发环境(Code Composer Studio 2.0)：完成系统的软件开发，进行软件和硬件仿真调试。它也是硬件调试的辅助手段。

.开发系统(ICETEK 5100 USB 或 ICETEK 5100 PP)：实现硬件仿真调试与硬件系统的通信，控制和读取硬件系统的状态和数据。

.评估模块(ICETEK -VC5509-C 等)：提供软件运行和调试的平台和用户系统开发的参照。

Code Composer Studio 2.0 主要完成系统的软件开发和调试。它提供一整套的程序编制、维护、编译、调试环境，能将汇编语言和 C 语言程序编译连接生成 COFF (公共目标文件)格式的可执行文件，并能将程序下载到目标 DSP 上运行调试。

#用户系统的软件部分可以由 Code Composer Studio 建立的工程文件进行管理，工程文件一般包含以下几种文件：

- .源程序文件：C 语言或汇编语言文件(*.ASM 或*.C)
- .头文件(*.H)
- .命令文件(*.CMD)
- .库文件(*.LIB,*.OBJ)

四. 实验步骤

1. 实验准备

. 连接实验设备

如使用 USB 型仿真器,将提供的 USB 电缆的扁平端连接到计算机的 USB 接口上,另一端不接;

如使用 PP 型仿真器,首先确认计算机电源处于关闭状态,然后将提供的并口电缆的一端连接到计算机的并行接口上;

连接仿真器的仿真电缆接头到 DSP 系统板上的 JTAG 接头。注意仿真器接头上有一个插孔中有一个封针,DSP 系统板上的 JTAG 接口的相应插针是被空开的,这样保证了仿真接头的方向不会接反。

如使用 PP 型仿真器,连接并口电缆的未连接端到仿真器上相应接头;连接 5V 电源;

使用 ICETEK-VC5509-C 评估板的 DSP 系统板,关闭 DSP 系统板上的电源开关;

将 5V 电源连接到 DSP 系统板上;

. 开启设备

接通计算机电源,进入 Windows 操作系统。

使用 ICETEK-VC5509-C 评估板的 DSP 系统板,DSP 系统板上电源指示灯亮。

. 安装 Code Composer Studio 2.0(可选做)

将实验箱附带的教学光盘插入计算机光盘驱动器;

利用桌面上“我的电脑”打开光盘的 CCS 开发软件目录,双击“ccs5000.exe”,进入安装程序;

选择“Code Composer Studio”,按照安装提示进行安装,并重新启动计算机;

安装完毕,桌面上出现两个新的图标“Setup CCS 2(‘C5000)’”、“CCS 2(‘C5000)’”;

利用桌面上“我的电脑”打开光盘的 CCS 开发软件目录,双击“C5000-2.20.00-FULL-to-C5000-2.21.00-FULL.exe”,进入 CCS 软件补丁安装程序。(此补丁用于 CCS 开发软件对 55XX 系列 DSP 软件的仿真和调试。最新的补丁安装名称以光盘中附带的为准。)

. 安装 DSP 开发系统驱动程序(可选做)

安装 USB 型仿真器的驱动程序:

请参考光盘中附带的 ICETEK-5100pp(usb)通用开发系统使用说明书。

安装 PP 型仿真器的驱动程序:

请参考光盘中附带的 ICETEK-5100pp(usb)通用开发系统使用说明书。

2. 设置 Code Composer Studio2.0 在软件仿真(Simulator)方式下运行(可选做)

. 双击桌面上“Setup CCS 2(‘C5000)”,启动“Code Composer Studio Setup”。

. 在“Import Configuration”对话框中单击“Clear”按钮,在接下来的对话框中选择“是”,清除原先的系统设置;窗口“Code Composer Studio Setup”中左侧“System Configuration”栏中“My System”项被清空。

- . 在 “ Available Configurations ” 列表中，单击选择 “ C5510 Device Simulator ” 驱动，并单击 “ Import ” 按钮；窗口 “ Code Composer Studio Setup ” 中左侧 “ System Configuration ” 栏中 “ My System ” 项中被加入 “ C5510 Device Simulator ” 项。
 - . 单击 “ Close ” 按钮，退出 “ Import Configuration ” 对话框。
 - . 选择 “ Code Composer Studio Setup ” 窗口 “ File ” 菜单中 “ Exit ” 项推出，并在接下来显示的对话框中选择 “ 是 ”，保存设置；选择 “ 否 ”，不启动 CCS。
3. 设置 Code Composer Studio 2.0 在硬件仿真(Emulator)方式下运行
- . 双击桌面上 “ Setup CCS 2(‘C5000) ”，启动 “ Code Composer Studio Setup ”。
 - . 在 “ Import Configuration ” 对话框中单击 “ Clear ” 按钮，在接下来的对话框中选择 “ 是 ”，清除原先的系统设置；窗口 “ Code Composer Studio Setup ” 中左侧 “ System Configuration ” 栏中 “ My System ” 项被清空。
 - . 对于 USB 型仿真器(如使用 PP 型仿真器则跳过此步)，在 “ Available Configurations ” 列表中，单击选择 “ ICETEK-5100 USB Emulator for 55x ” 驱动，并单击 “ Import ” 按钮；窗口 “ Code Composer Studio Setup ” 中左侧 “ System Configuration ” 栏中 “ My System ” 项中被加入 “ C55xx XDS510 Emulator ” 项。
 - . 单击 “ Close ” 按钮，退出 “ Import Configuration ” 对话框。
 - . 选择 “ Code Composer Studio Setup ” 窗口 “ File ” 菜单中 “ Exit ” 项推出，并在接下来显示的对话框中选择 “ 是 ”，保存设置；
4. 启动 Code Composer Studio 2.0
- 双击桌面上 “ CCS 2(‘C5000) ”，启动 Code Composer Studio 2.0；可以看到显示出的 C55XX Code Composer Studio 窗口；
5. 创建工程
- . 创建新的工程文件：

选择菜单 “ Project ” 的 “ New...” 项；在 “ Project Creation ” 对话框中，在 “ Project ” 项输入 volume；单击 “ Location ” 项末尾的浏览按钮，改变目录到 ICETEK – VC5509-C\Lab1-UseCC，单击 “ OK ”；单击 “ 完成 ”；这时建立的是一个空的工程文件；展开主窗口左侧工程管理窗口中 “ Projects ” 下新建的 “ volume.pjt ”，其中各项均为空。
 - . 在工程文件中添加程序文件：

选择菜单 “ Project ” 的 “ Add Files to Project...” 项；在 “ Add Files to Project ” 对话框中选择文件目录为 ICETEK–VC5509-C\Lab1-UseCC，改变文件类型为 “ C Source Files(*.c;*.ccc) ”，选择显示出来的文件 “ volum.c ”；重复上述各步骤，添加 volume.cmd 文件到 volum 工程中。
 - . 编译连接工程：

选择菜单 “ Project ” 的 “ Rebuild All ” 项；注意编译过程中 CCS 主窗口下部的 “ Build ” 提示窗中显示编译信息，最后将给出错误和警告的统计数。
6. 编辑修改工程中的文件
- . 查看工程文件

展开 CCS 主窗口左侧工程管理窗中的工程各分支，可以看到 “ volume.pjt ” 工程中包含 “ volume.h ”、“ volume.c ” 和 “ volume.cmd ” 文件，其中第一个为程序在编译时

根据程序中的“include”语句自动加入的。

· 查看源文件

双击工程管理窗中的“volume.c”文件，可以查看程序内容。

双击工程管理窗中的“volume.h”文件，打开此文件显示，可以看到其中有主程序中要用到的一些宏定义如“BUF_SIZE”等。

“volume.cmd”文件定义程序所放置的位置，此例中描述了5509的存储器资源，指定了程序和数据在内存中的位置。

· 编辑修改源文件

打开“volume.c”，找到“main()”主函数，将语句“input = &inp_buffer[0];”最后的分号去掉，这样程序中就出现了一个语法错误；重新编译连接工程，可以发现编译信息窗口出现发现错误的提示，双击红色错误提示，CC自动转到程序中出错的地方；将语句修改正确(这里是将语句末尾的分号加上)；重新编译；注意，重新编译时修改的文件被CC系统自动保存。

· 修改工程文件的设置

选择“Project”菜单中的“Build Options...”项，打开“Build Options for volume.pjt”对话框，选择“Linker”卡片，在“Stack Size”项后输入1024；单击“确定”完成设置；通过此设置，重新编译后，程序中的堆栈的尺寸被设置成1024个字。

7. 基本调试功能

· 执行File→Load Program，在随后打开的对话框中选择刚刚建立的C:\ICETEK-VC5509-C\Lab1-UseCC\Debug\volume.out文件。

· 在项目浏览窗口中，双击volume.c激活这个文件，移动光标到main()的下一行上，右击鼠标选择Toggle Breakpoint或按F9设置断点。

· 选择Debug→Run或按F5运行程序，程序会自动停在main()函数头上。

按F10执行到write_buffer()函数上。

再按F8，程序将转到write_buffer函数中运行。

此时，为了返回主函数，按shift-F7完成write_buffer函数的执行。

再次执行到write_buffer一行，按F10执行程序，对比与F8执行的不同。

注意：在执行C语言的程序时，为了快速的运行到主函数调试自己的代码，可以使用Debug→Go main命令，上述实验中的使用的是较为繁琐的一种方法。

8. 使用观察窗口

· 执行View→Watch Window打开观察窗口。

· 在volume.c中，选中任意一个变量，右击鼠标，选择“Quick Watch”，CCS将打开Quick Watch窗口并显示选中的变量。

· 在volume.c中，选中任意一个变量，右击鼠标，选择“Add to Watch Window”，CCS将把变量添加到观察窗口并显示选中的变量值。

· 在观察窗口中双击变量，则弹出修改变量窗口，此时，可以在这个窗口中改变程序变量的值。

· 把str变量加到观察窗口中，点击变量左边的“+”，观察窗口可以展开结构变量，并且显示结构变量的每个元素的值。

- . 把 str 变量加到观察窗口中;执行程序进入 write_buffer 函数,此时 num 函数超出了作用范围,可以利用 Call Stack 窗口察看在不同作用范围的变量:
 - 执行 View→Call Stack 打开堆栈窗口。
 - 双击堆栈窗口的 main()选项,此时可以察看 num 变量的值。

9. 文件输入/输出

这一节介绍如何从 PC 机上加载数据到目标机上。可用于使用已知的数据流测试算法的正确性。

在完成下面的操作以前,先介绍 Code Composer Studio 的 Probe (探针)断点,这种断点允许用户在指定位置提取/注入数据。Probe 断点可以设置在程序的任何位置,当程序运行到 Probe 断点时,与 Probe 断点相关的事件将会被触发,当事件结束后,程序会继续执行。在这一节里,Probe 断点触发的事件是:从 PC 机的数据文件加载数据到目标系统的缓冲区中。

- . 在真实的系统中,read_signals 函数用于读取 A/D 模块的数据并放到 DSP 缓冲区中。在这里,代替 A/D 模块完成这个工作的是 Probe 断点。当执行到函数 read_signals 时,Probe 断点完成这个工作。
 - 在程序行 read_signals(int *input)上单击鼠标右键,选择“Toggle breakpoint”,设置软件断点。
- . 执行 File→File I/O,打开对话框。
- . 点击 Add File 把 sine2.dat 文件加到对话框中。
- . 完成设置:
 - 在 Address 中,输入 inp_buffer
 - 在 Length 中,输入 100
 - 保证 wrap around 被选中;
- . 关联事件和 Probe 断点:
 - 点击 Add Probe Point 按钮,打开对话框;
 - 点击 Probe Point 列表中的内容,使之被选中;
 - 在 Connect 中选择 sine2.dat 文件;
 - 点击 Replace 按钮确认设置;
 - 点击“确定”关闭对话框。
- . 点击“确定”关闭对话框,此时,已经配置好了 Probe 断点和与之关联的事件.进一步的结果在下面实验中显示;

10. 图形功能简介

下面我们使用 CC 的图形功能检验上一节的结果

- . 执行 View→Graph→Time/Frequency 打开 Graph Property Dialog 窗口;
- . 修改属性为如下值并确定:

Graph Title :	Input
Start Address :	inp_buffer
Acquisition Buffer Size :	100
Display Data Size	100

DSP Type : 16-bit signed integer

-在弹出的图形窗口中单击鼠标右键，选择“Clear Display”。

. 按 F12 运行程序.观察 input 窗口的内容。

五. 实验结果

通过对工程文件“volume”的编译、执行后得到结果的图形显示如下：

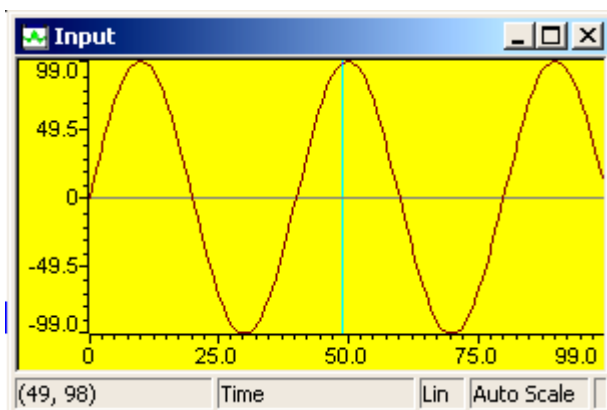


图 2-1-1 正弦数据波形图

六. 问题与思考

实验二：编制链接控制文件

一. 实验目的

1. 学习用汇编语言编制程序；了解汇编语言程序与 C 语言程序的区别和在设置上的不同；
2. 学习编制命令文件控制代码的连接；
3. 学会建立和改变 map 文件，以及使用它观察内存使用情况的方法。
4. 熟悉使用软件仿真方式调试程序。

二. 实验设备

PC 兼容机一台，操作系统为 Windows2000(或 Windows98，WindowsXP，以下默认为 Windows2000)，安装 Code Composer Studio 2.0 软件。

三. 实验原理

1. 汇编语言程序

汇编语言程序除了程序中必须使用汇编语句之外，其编译选项的设置与 C 语言编制的程序也稍有不同。其区别为：

汇编语言程序在执行时直接从用户指定入口开始，常见的入口标号为“start”，而 C 语言程序在执行时，先要调用 C 标准库中的初始化程序(入口标号为“_c_init00”)，完成设置之后，才转入用户的主程序 main()运行；为了支持 C 初始化代码的连接，C 程序在编译时要包含 C 语言库和与之相配的头文件，这需要用户将库添加到工程中。

由于 Code Composer Studio 的代码链接器默认支持 C 语言，在编制汇编语言程序时，需要设置链接参数，选择非自动初始化，注明汇编程序的入口地址。

2. 命令文件的作用

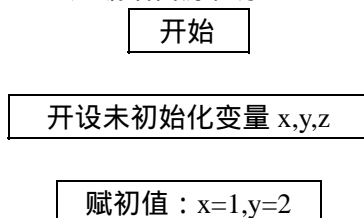
命令文件(文件名后缀为 cmd)为链接程序提供程序和数据在具体 DSP 硬件中的位置分配信息。通过编制命令文件，我们可以将某些特定的数据或程序按照我们的意图放置在 DSP 所管理的内存中。命令文件也为链接程序提供了 DSP 外扩存储器的描述。

3. 内存映射(map)文件的作用

一般地，我们设计、开发的 DSP 程序在调试好后，要固化到系统的 ROM 中。为了更精确地使用 ROM 空间，我们就需要知道程序的大小和位置，通过建立目标程序的 map 文件可以了解 DSP 代码的确切信息。当需要更改程序和数据的大小和位置时，就要适当修改 cmd 文件和源程序，再重新生成 map 文件来观察结果。

4. 源程序分析

图 2-2-1 汇编语言源程序 UseCMD.asm 框图：



在累加器中计算 $x+y$

输出结果到 z

循环

四. 实验步骤

1. 实验准备

设置软件仿真模式：

启动 CC 驱动设置窗口：双击桌面上“Setup CCS 2(‘C5000)”图标。

清除原先驱动设置：单击“Clear”按钮。

安装软件仿真驱动 (Simulator)：单击“C5510 Device Simulator”驱动名，单击“Import”按钮。

完成设置：单击“Close”，菜单“File”、“Exit”，“是”。

2. 打开工程文件

· 双击桌面上“CCS 2(‘C5000)”，启动 Code Composer Studio 2.0。

· 打开菜单“Project”的“New...”项；在“Project”项中输入 UseCMD，在“Location”中选择 ICETEK-VC5509-C\Lab2-UseCMD 目录，单击“完成”建立 UseCMD.pjt。

3. 设置工程文件

· 打开设置窗口：选择菜单“Project”的“Build Options...”项。

· 选择链接设置：单击“Linker”属性页。

· 观察汇编语言程序的特殊设置：

· “Autoinit Model”项设置成“ No Autoinitialization ”

· “Code Entry Point”项中输入“start”。

· 退出设置窗口：单击“确定”按钮。

4. 编译源文件，下载可执行程序

· 单击菜单“Project”、“Rebuild All”；

· 执行 File→Load Program，在随后打开的对话框中选择刚刚建立的 UseCMD.out 文件。完成后，系统自动打开源程序文件 UseCMD.asm。

5. 打开观察窗口

· 开启 CPU 寄存器观察窗口：单击菜单“View”、“CPU Registers”。

· 在内存观察窗口中观察变量的值：

选择“View”菜单中“Memory...”项，在“Memory Window Options”窗口中的“Address”项中输入 x ，单击“OK”完成设置；在随后显示的“Memory”窗口中单击鼠标右键，选择“Float In Main Window”项。

6. 观察程序运行结果

这时，代表程序运行位置的黄色光标条停在 start 标号下面语句上，程序将从此开始执行。

· 单步执行程序（按 F10 键）1 次，可观察到 CPU 寄存器窗口中 PC 的值有变化。

- . 单步运行 2 次，在变量窗口中观察到变量 x、y 被赋值。
- . 单步执行到 xh 标号后面的语句，观察 AC0 寄存器和变量 z 值的变化。
- 7. 生成内存映像文件
 - . 单击菜单“Project”、“Options...”，启动“Build Options”工程设置对话框。
 - . 单击“Linker”属性页，在“Map Filename”项中输入需要生成的 map 文件名，比如可以输入 UseCMD.map
 - . 单击“确定”，完成设置。
 - . 选择菜单“Project”、“Rebuild All”，重新编译工程，生成新设置的 map 文件。
- 8. 对照观察 map 文件和 cmd 文件的内容
 - . 选择菜单“File”、“Open...”，将找到 ICETEK-VC5509-C 评估板 \Lab2-UseCMD 目录，将文件类型改为“Memory Map Files”，选择刚刚生成的 UseCMD.map 文件、打开。
 - . 展开工程管理窗中的 UseCMD.pjt，双击其中的 UseCMD.cmd 文件。
 - . 程序的入口地址：cmd 文件的 SECTION 中指定.text 段放到程序区的 PROG 中，在 MEMORY 中指定 PROG 从内存地址 200h 开始，长度为 2000h；再看 map 文件中“ENTRY POINT SYMBOL”中说明了“start”标号的地址为十六进制 0000200，两者相符。
 - . 内存的占用情况：通过观察 map 文件中的“MEMORY CONFIGURATION”段可以了解内存的使用情况，可以看到，程序所占用的长度为十六进制 15，即 15 个字长，而数据区因开设了 3 个变量，所以占用了 6 个字的地址空间
- 9. 改变内存分配
 - 修改 cmd 文件中的
 - PROG : o=200h,l=1f00h
 - 改为
 - PROG : o=200h,l=1e00h
 - 重新编译工程，观察 map 文件中有何变化。

五. 实验结果

通过实验可以发现，修改 cmd 文件可以安排程序和数据在 DSP 内存资源中的分配和位置；map 文件中描述了程序和数据所占用的实际尺寸和地址。

实验程序中计算变量的取值之和，由于取值较小，所以结果仍为 16 位数，程序中仅考虑保存 acc 的低 16 位作为结果。但如果计算中有进位等问题就需要考虑保存 acc 的高 16 位结果了。

六. 问题与思考

请修改程序完成无符号数 0f000h+0e000h 的计算。

实验三：数据存取实验

一、实验目的

- 1、了解 TMS320VC5509 的内部存储器空间的分配及指令寻址方式；
- 2、了解 ICETEK-VC5509-C 评估板扩展存储器空间寻址方法，及其应用；
- 3、学习用 Code Composer Studio 修改、填充 DSP 内存单元的方法；
- 4、学习操作 TMS320C55xx 内存空间的指令。

二、实验设备

计算机， ICETEK 仿真器+ICETEK – VC5509-C 评估板+相关连线及电源。

三、实验原理

实验程序分析

源程序 Memory.asm

```
.global start    ; 定义全局标号
.mmregs
.text
start:
    nop
    ld      #68,dp    ;直接寻址，装载 DP 值，页指针指向数据区 0x2200
    st      #1,1      ; 绝对地址 2201H 开始的四个单元存 1，2，3，4
    st      #2,2
    st      #3,3
    st      #4,4

    stm      #2205h,ar1    ; 间接寻址，使用辅助寄存器 1
    rpt      #3            ; 循环重复执行下条语句 4 次
    st      #1234h,*ar1+    ; 将绝对地址 2205H 开始的 4 个单元存成 1234H

                                ; 下面将 2201H 开始的 8 个数读出
                                ; 存到 2300H 开始的 8 个单元
    stm      #7h,ar3        ; 循环计数器=7（8 次循环）
    stm      #2201h,ar1     ; 源起始地址
    stm      #2300h,ar2     ; 目的起始地址
loop:
    ld      *ar1+,T    ; 将 ar1 指向单元内容读入 T 寄存器，ar1 的值+1
    st      T,*ar2+    ; 将 T 寄存器的值转存到 ar2 指向的目的地址，ar2 的值+1
    banz    loop,*ar3-    ; ar3 的值-1，循环计数不等于 0 则循环
xh:
```

b xh ; 空循环
.end

四.实验步骤

1. 实验准备

. 连接设备：

关闭计算机电源。

使用 PP 型仿真器则用附带的并口连线连接计算机并口和仿真器相应接口；

. 开启设备：

打开计算机电源。

打开电源开关，注意 ICETEK – VC5509-C 评估板上指示灯 D10 亮。

如使用 USB 型仿真器用附带的 USB 电缆连接计算机和仿真器相应接口，注意仿真器上两个指示灯均亮。

. 设置 Code Composer Studio 为 Emulator 方式：

参见“Code Composer Studio 入门实验”之四.3。

. 启动 Code Composer Studio

双击桌面上“CCS 2(‘C5000)”图标，启动 Code Composer Studio 2.0。

2. 打开工程文件

打开菜单“Project”的“Open”项；选择 ICETEK – VC5509-C 评估板\Lab3-Memory 目录中的“Memory.pjt”。

3. 运行程序观察结果

编译和下载程序：单击菜单“Option”、“Customize...”，选择“Program Load Options”卡片，在“Load Program After Build”之前加上选择符号，单击“OK”按钮，此设置完成在每次编译完成后将程序自动下载到 DSP 上；选择菜单“Project”、“Rebuild All”，编译、连结和下载程序；观察“PROG”窗口中的变化。

五.实验结果

实验程序运行之后，位于数据区地址 2201H 开始的 8 个单元的数值被复制到了数据区 2300H 开始的 8 个单元中。

程序中使用了立即寻址、直接寻址和间接寻址方式。

*寻址方式的运用还有许多方法，可以完成复杂的寻址。

*如果在下载程序的过程中，Code Composer Studio 报告 xxxx 地址出现校验错误，可以使用内存显示和修改的方法，验证一下是否是该单元不能正确读写，以确定错误原因。

*通过改写内存单元的方式，我们可以手工设置 DSP 的一些状态位，从而改变 DSP 工作的状态。

六.问题与思考

实验四：定点数除法实验

一. 实验目的

1. 熟悉'C55x 指令系统，掌握常用汇编指令，学习设计程序和算法的技巧。
2. 学习用减法和移位指令实现除法运算。

二. 实验设备

PC 兼容机一台，操作系统为 Windows2000(或 Windows98，WindowsXP，以下默认为 Windows2000)，安装 Code Composer Studio 2.0 软件。

三. 实验原理

由内置的硬件模块支持，数字信号处理器可以高速的完成加法和乘法运算。但 TMS320 系列 DSP 不提供除法指令，为实现除法运算，需要编写除法子程序来实现。二进制除法是乘法的逆运算。乘法包括一系列的移位和加法，而除法可分解为一系列的减法和移位。本实验要求编写一个 16 位的定点除法子程序。

1. 除法运算的过程

设累加器为 8 位，且除法运算为 10 除以 3，除的过程包括与除数有关的除数逐步移位，然后进行减法运算，若所得商为正，则在商中置 1，否则该位商为 0

例如：4 位除法示例：

数的最低有效位对齐被除数的最高有效位

```
00001010
- 00011000
-----
```

11110010

由于减法结果为负，丢弃减法结果，将被除数左移一位再减

```
00010100
- 00011000
-----
```

11111000

结果仍为负，丢弃减法结果，将被除数左移一位再减

```
00101000
- 00011000
-----
```

00010000

结果为正，将减法结果左移一位后把商置 1，做最后一次减

```
00100001
- 00011000
-----
```

00001001

结果为正，将减法结果左移一位加 1 得最后结果，高 4 位是余数，低 4 位商：

00010011

2. 除法运算的实现

为了提高除法运算的效率，'C55x 系列提供了条件减指令 SUBC 来完成除法操作。

SUBC 指令的功能如下：

若 (ACC) = 0 且 (数据存储器地址) = 0

PC+1 然后

$(ACC) - [(数据存储器地址) \times 2^{15}] \rightarrow ALU \text{ 输出}$

如果 ALU 输出 = 0

则： $(ALU \text{ 输出}) \times 2 + 1 \rightarrow ACC$

否则： $(ACC \times 2) \rightarrow ACC$

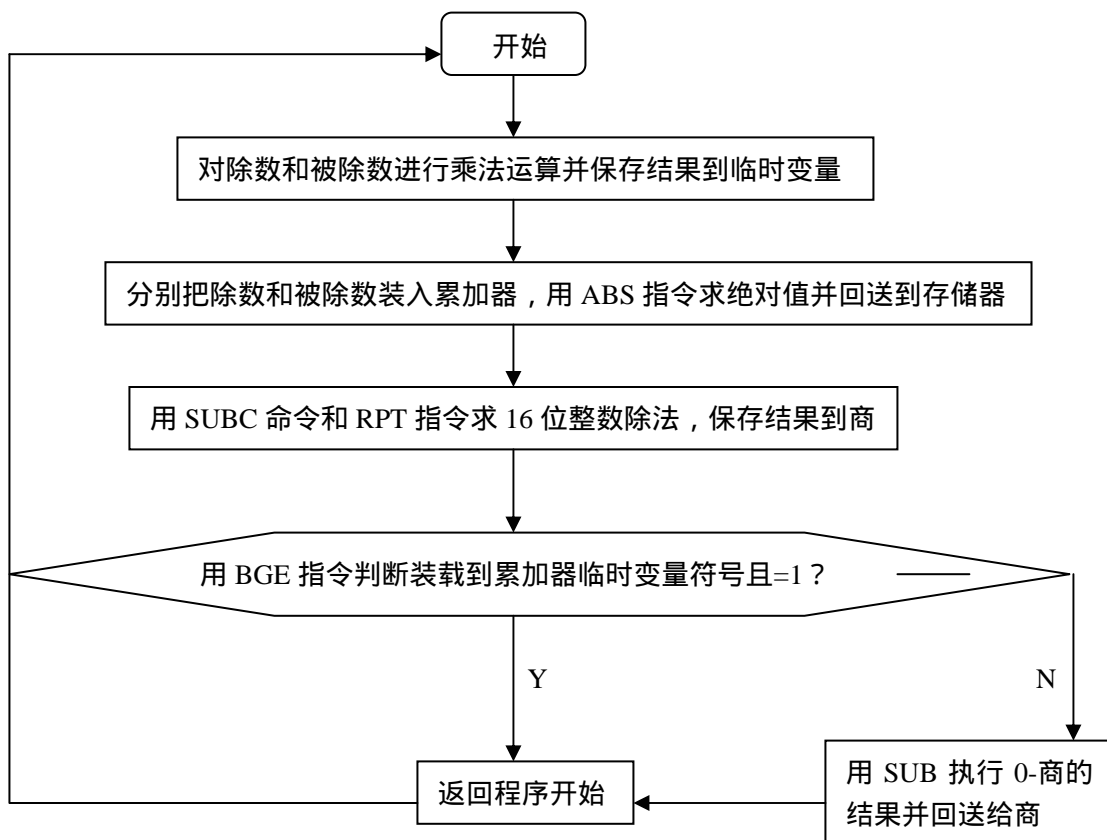
实际上，SUBC 指令完成的是除法中的减除数求商的过程，即余数末位补 0，减去除数，若结果为正，该位商为 1，否则商为 0。

SUBC 指令实现条件减，可以用于如下除法：把 16 位的正被除数放在累加器的低 16 位，累加器的高 16 位清 0，16 位的正除数放在数据存储器单元中。执行 SUBC 指令 16 次，最后一次 SUBC 指令完成后，累加器的低 16 位是除法的商，高 16 位是余数。若累加器和/或数据存储器单元的内容为负，则不能用 SUBC 指令实现除法。为了完成多次的 SUBC 指令，还需要用到循环指令 RPT，它可以使 RPT 后的一条指令重复 1—256 次。

SUBC 指令仅能对正数除法进行运算，因此，要扩展到所有数值的除法，还需要：在程序开头对被除数和除数做乘法，并保存到临时变量，除数和被除数分别取绝对值，在除法运算完成后，根据临时变量的值修改商的符号。

3. 除法运算程序流程：

图 2-4-1 除法运算程序流程



四. 实验步骤

1. 用 Simulator 方式启动 Code Composer。
2. 执行 Project→New 建立新的项目，输入 div 作为项目的名称，将程序定位在 ICETEK-VC5509-C\Lab4-Division 目录。
3. 执行 File→New→Source File 建立新的程序文件，为创建新的程序文件命名为 div.asm 并保存；执行 Project→Add Files to Project，把 div.asm 加入项目中。
4. 执行 File→New→Source File 建立新的文件并保存为 div.cmd；执行 Project→Add Files to Project，把 div.cmd 加入项目中。
5. 编辑 div.asm 加入如下内容：

```

.global start ; 定义全局标号
.mmregs

.data
.bss NUMERA,1
.bss DENOM,1
.bss QUOT,1
.bss ARIT,1
.bss TEMSGN,1
.text
start:
    nop
    nop
next:
    call DIV
    b     next
; 除法子程序
; 输入：NUMERA 被除数，DENOM 除数
; 输出：QUOT 商，ARIT 余数
DIV:ld    #NUMERA,dp
    ld    @NUMERA,T    ; 将被除数装入 T 寄存器
    mpy   @DENOM,A      ; 除数与被除数相乘，结果放入 A 寄存器
    ld    @DENOM,B      ; 将除数装入 B 寄存器的低 16 位
    abs   B              ; 求绝对值
    stl   B,@DENOM      ; 保存 B 寄存器的低 16 位
    ld    @NUMERA,B      ; 将被除数装入 B 寄存器低 16 位
    abs   B              ; 求绝对值
    rpt   #15            ; 重复下条 subc 指令 16 次

```

```

subc  @DENOM,B      ; 完成除法运算
bcd   done,AGT      ; 延时条转，先执行下面两条指令，然后判断 A,
                        ; 若 A>0 则跳转到标号 done,结束除法运算

stl   B,@QUOT      ; 保存商（B 寄存器的低 16 位）
sth   B,@ARIT      ; 保存余数（B 寄存器的高 16 位）
xor   B             ; 若两数相乘的结果为负，则商也应为负
                        ; 先将 B 寄存器清 0

sub   @QUOT,B      ; 将商反号
stl   B,@QUOT      ; 存回反号后的商值

done:
ret

```

6. 编辑 div.cmd 加入如下内容

MEMORY {

```

    DATA(RWI):  origin = 0x2200,      len = 0x1000
    PROG:        origin = 0x200,      len = 0x2000
    VECT:        origin = 0xd000,      len = 0x100

```

}

SECTIONS

{

```

.text:      {} > PROG
.vectors: {} > VECT
.trcinit: {} > PROG
.gblinit: {} > PROG
  frt:      {} > PROG

.cinit:     {} > PROG
.pinit:     {} > PROG
.sysinit: {} > PROG
.bss:       {} > DATA
.far:       {} > DATA
.const:     {} > DATA
.switch:    {} > DATA
.sysmem:    {} > DATA
.cio:       {} > DATA
.MEM$obj: {} > DATA
.sysheap: {} > DATA

```



```
.sysstack {} > DATA  
.stack:   {} > DATA
```

```
}
```

7. 执行 Project→Rebuild All 编译链接；编译错误如下：

warning : entry point symbol _c_int0 undefined

出错原因：缺省时 Code Composer Studio 设置项目程序为 C 语言编译，因此当我们编译汇编程序时,要对项目作适当配置。

8. 执行 Project→Build Options...打开编译选项；在 linker 属性页上单击，把 Autoinit Model 栏选择为 No Autoinitialization；按“确定”保存对配置的修改。

(1)执行 File→Load Program 装载程序，装载完程序后，Code Composer Studio 把指针指向程序区 0200 处。为了执行我们的程序代码，需要修改 DSP 的 PC 值；执行 View→CPU Registers→CPU Registers 打开寄存器窗口；双击窗口中的 PC 标号，CC 弹出修改对话框供修改寄存器；在对话框中输入“start”，程序将处于我们的程序入口点上。

(2)在观察窗口中添加变量：

(3)在 div.asm 窗口中 NUMERA 变量名上双击鼠标左键，再单击鼠标右键，选择“Add to Watch Window”，在观察窗口中出现 NUMERA 变量，请将窗口中变量名 (Name) 由“NUMERA”改成“(int*)NUMERA”；“Radix”由“hex”改成“dec”；

(4)重复 的操作，加入 DENOM、QUOT 和 ARIT 四个变量。

这样，我们就添加了为完成除法操作而需要的输入和输出变量。其中 NUMERA 是被除数，DENOM 是除数，QUOT 是商,而 ARIT 是余数。

(5)点击观察窗口上的 NUMERA 变量上的+号，在“Value”区(即第二行)输入数据“10”；同样输入 DENOM 变量的值“3”。

9. 按 F10 执行程序到“bnext”，观察程序的 QUOT 和 ARIT 两个变量是否是正确的结果。此外，还可以多运行几次程序，分别用不同的数据测试除法程序。

五. 实验结果

试验中用定点数计算代替除法运算，而在浮点 DSP 上可选用被除数乘以除数的倒数的方法进行除法替代运算。

六. 问题与思考

实验五：I/O 端口实验

一 实验目的

1. 了解 ICETEK-VC5509-C 评估板在 I/O 端口上的扩展；
2. 掌握 I/O 端口的控制方法；
3. 学习在 C 语言中控制 I/O 端口读写的方法。

二 实验设备

计算机， ICETEK 仿真器+ICETEK – VC5509-C 评估板+相关连线及电源。

三 实验原理

LEDR 寄存器：地址是 0x400001

表 2-5-1 LEDR 寄存器

7	4	3			0
无效位		LEDR3	LEDR2	LEDR1	LEDR0
		R/W-0	R/W-0	R/W-0	R/W-0

图例：R 读允许，W：写允许，R/W:读写允许，-0：复位值,-x 没有固定值

位	名称	说明
3	LEDR3	四位显示数码管。向某位写“0”，点亮相应的发光管，写“1”，则使数码管熄灭。这个寄存器可读，在使用时可以通过读操作得到最后一次写操作的值。
2	LEDR2	
1	LEDR1	
0	LEDR0	

四 实验步骤

1. 实验准备

(1) 连接设备：

关闭计算机电源。

如使用 PP 型仿真器则用附带的并口连线连接计算机并口和仿真器相应接口；

(2) 开启设备：

开启计算机电源

打开电源开关，注意 ICETEK – VC5509-C 评估板上指示灯 D10 亮。

如使用 USB 型仿真器，用附带的 USB 电缆连接计算机和仿真器相应接口，注意仿真器上两个指示灯均亮。

(3) 设置 Code Composer Studio 2.0 为 Emulator 方式：

参见“Code Composer Studio 入门实验”之四.3。

(4) 启动 Code Composer Studio

双击桌面上“CCS 2(‘C5000)”图标，启动 Code Composer Studio 2.0。

2. 打开工程文件

打开菜单“Project”的“Open”项；选择 ICETEK – VC5509-C \Lab5-IOPort 目录中的

“ IOPort1.pjt ”。

3. 浏览程序

在项目浏览器中，双击 main.c 文件，浏览该文件的内容，理解各语句作用。

4. 编译工程

打开“ Project ”选单，选择“ Rebuild All ”选项，Code Composer Studio 重新编译和链接这个工程项目，整个的处理过程在屏幕下方的“ Message ”窗口中返回信息。

5. 下载程序

打开“ File ”选单，选择“ Load Program ”选项，在“ Load Program ”对话框中，选中 ICETEK – VC5509-C \Lab5-IOPort\Debug 目录下的 led.out 文件；此时，Code Composer Studio 将把这个目标文件装载到 ICETEK – VC5509-C 评估板上，同时，Code Composer Studio 打开反汇编窗口显示被加载程序的汇编指令码。

6. 运行程序观察结果

打开“ Debug ”菜单，选择“ Run ”选项运行程序（或按 F5 键），可看到发光二极管在依次闪烁。

7. 结束运行，关闭工程

(1)打开“ Debug ”选单，选择“ Halt ”选项或按 Shift-F5 终止实验。

(2)选择菜单“ Project ”、“ Close ”关闭工程。

五 实验结果

*实验的最后现象：可以看到指示灯依次闪烁。

*I/O 端口操作的地址和数据均通过总线完成，与读写单独的数据存储单元的方式一样，只是使用的地址空间不同。

*请参照程序，总结使用 I/O 端口控制简单外围设备的方式和方法。

六 问题与思考

*I/O 端口操作与内存读写操作的效果一样吗？各自的优势是什么？

*如果我需要扩展一片 Flash，应在地址上如何进行映射才方便程序使用？如果扩展的是 FIFO 呢？

实验六：语音编解码实验

一. 实验目的

通过实验学习使用 5509 DSP 的扩展立体声音频设备的方法,了解音频芯片的控制编程方法。

二. 实验设备

PC 兼容机一台,操作系统为 Windows2000(或 Windows98, WindowsXP, 以下默认为 Windows2000),安装 Code Composer Studio 2.0 软件。

ICETEK-VC5509-C 目标板,双头音频线一根,CD 一张,音箱或耳机一个。

三. 实验原理

通过扩展音频芯片,我们可以使用 DSP 芯片进行音频的处理,加入一些滤波算法,对音频信号进行分析。

```
#include "5509.h"
void CLK_init( void );
void main()
{
    CLK_init();
    AIC23_Init();
    for(;;)
        AIC23_Mixer();
}
void CLK_init( void )
{
    ioport unsigned int *clkmd;
    clkmd=(unsigned int *)0x1c00;
    *clkmd = 0x2413;// 144MHz //2613
}
```

四. 实验步骤

实验准备

正确连接设备（请按照实验一中的操作步骤来操作）后,进行如下操作：

把 CD 盘放到 PC 机的光驱中,然后按光驱上的播放键播放 CD 音乐。

注意：如果 PC 机中同时运行其他播放软件,请关闭。

把双头音频线一头接在光驱的耳机输出孔,一头接在 icetek-vc5509-C 板的音频接口上。（连接的具体音频接口是第一个音频接口,硬件上标着 J7。）

把音箱的音频线接到 J9 上。

1. 打开工程文件

双击桌面上“CCS 2('C5000)”,启动 Code Composer Studio 2.0。

打开工程并浏览程序,工程目录为 audio.pjt

2. 编译源文件,下载可执行程序

单击菜单 “ Project ”、“ Rebuild All ”;

执行 File→Load Program ,在随后打开的对话框中选择刚刚建立的 AUDIO.out 文件。

3. 观察程序运行结果

五 . 实验结果

从音箱中放出光驱中播放的 CD 音乐。

六 . 问题与思考

试着把 FFT , FIR 等算法加入到例子程序中 , 看看输出的音乐有无变化 ?

实验七：自启动实验

一. 实验目的

1. 了解 TMS320VC5509 DSP 芯片的微处理器工作方式；
2. 了解 TMS320VC5509 DSP 芯片的自启动方式；
3. 学习了解 ICETEK-VC5509-C 评估板上 Flash 的扩展方式和特点；
4. 掌握 ICETEK-VC5509-C 评估板上 Flash 的烧写过程；
5. 学习自启动程序的设计。

二. 实验设备

计算机， ICETEK 仿真器+ICETEK – VC5509-C 评估板+相关连线及电源。

三. 实验原理

1. MP/MC 方式

TMS320VC5509 DSP 芯片有两种工作方式，一种为微控制器（MC）方式，一种为微处理器方式（MP）。

2. ICETEK – VC5509-C 评估板上扩展 Flash 的方式和特点

- Flash 扩展在数据空间，地址从 200000h 开始，容量为 1M 字节，采用 16 位方式。
- 如果需要访问 Flash 存储器，MP/MC 必须为低电平。

3. 5509 自启动加载器（BootLoader）

自启动加载器是一段固化在 5509 片内 ROM 中的程序，它主要完成在上电时从外部加载并执行用户的程序代码。

4. 自启动程序编制要点

程序一般先在外扩展的 RAM 中调试好，应注意调整程序的尺寸，而且一般要连续存放，以利于进行 BootLoad。程序中的常量部分应分配到程序区存储。从生成的内存映射文件（map 文件）中读取程序的入口地址，以便在自启动模式中转移到此处运行。

四. 实验步骤

1. 实验准备

. 连接设备

关闭计算机电源。

如使用 PP 型仿真器则用附带的并口连线连接计算机并口和仿真器相应接口。

检查 ICETEK – VC5509-C 评估板上拨动开关 SW-6BIT 中 MP/MC(第四个) 开关的状态，应设置在 ON 位置，即设置 DSP 工作在 MC 方式。

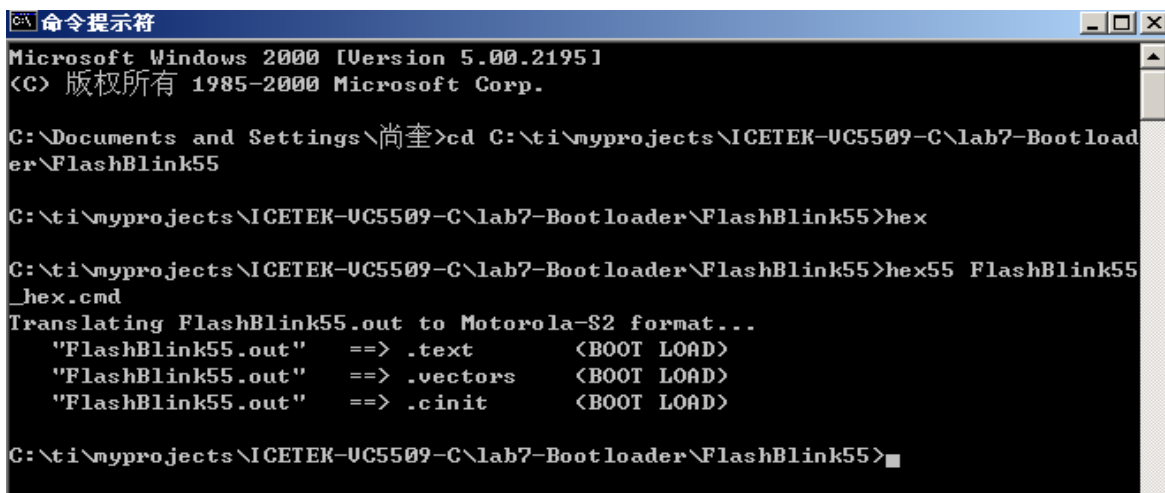
. 开启设备

打开计算机电源。

打开电源开关，注意 ICETEK – VC5509-C 评估板上指示灯 D10 亮。

如使用 USB 型仿真器用附带的 USB 电缆连接计算机和仿真器相应接口，注意仿真器上两个指示灯均亮。

- . 设置 Code Composer Studio 为 Emulator 方式：
参见“Code Composer Studio 入门实验”之四.3。
- . 启动 Code Composer Studio 2.0
- 2. 安装烧写 flash 的插件
-双击 ICETEK-5509-C\lab7-Bootload 中的 C5000C6000-2.00-SA-to-TI-FLASHBURN.exe 文件。按照提示安装插件，直到结束。
- 3. 打开工程并运行程序
-打开 ICETEK-5509-C\lab7-Bootloader\FlashBlink55 中的工程 FlashBlink55.pjt
-重新编译工程，生成.out 文件。
-单击“File”菜单的“Open...”；打开 FlashBlink55_hex.cmd 文件，查看：
FlashBlink55.out //要转换的.out 文件名称
-o FlashBlink.hex //要生成的.hex 文件名称
-map FlashBlink.mxp
-m2
-v5510:2
-boot
-parallel16
ROMS
{
 PAGE 0 : ROM : o=0x400000, l=0x100000
}
注意：以后要转换自己的.out 文件时只需修改头两行的参数即可，其他参数不变。
-单击“Project”、“Close”关闭工程。
- 4. 制作 HEX 文件
-启动“命令提示符”：单击“开始”菜单、“程序”、“附件”、“命令提示符”。
- 输入 hex 命令进行如下操作：



```
命令提示符
Microsoft Windows 2000 [Version 5.00.2195]
(C) 版权所有 1985-2000 Microsoft Corp.

C:\Documents and Settings\尚奎>cd C:\ti\myprojects\ICETEK-VC5509-C\lab7-Bootload
er\FlashBlink55

C:\ti\myprojects\ICETEK-VC5509-C\lab7-Bootloader\FlashBlink55>hex

C:\ti\myprojects\ICETEK-VC5509-C\lab7-Bootloader\FlashBlink55>hex55 FlashBlink55
_hex.cmd
Translating FlashBlink55.out to Motorola-S2 format...
"FlashBlink55.out" ==> .text <BOOT LOAD>
"FlashBlink55.out" ==> .vectors <BOOT LOAD>
"FlashBlink55.out" ==> .cinit <BOOT LOAD>

C:\ti\myprojects\ICETEK-VC5509-C\lab7-Bootloader\FlashBlink55>
```

图 2-7-1 Hex55 命令使用

这时在 ICETEK-VC5509-C\Lab7-BootLoader 目录中生成了 Flashblink.hex 文件，下面将利用这个文件烧写到 ICETEK – VC5509-C 评估板上的 Flash，以实现自启动功能。

5. 打开烧写插件

重新进入 ccs 开发环境，点击 tools 菜单下的 flashburn，弹出下图：

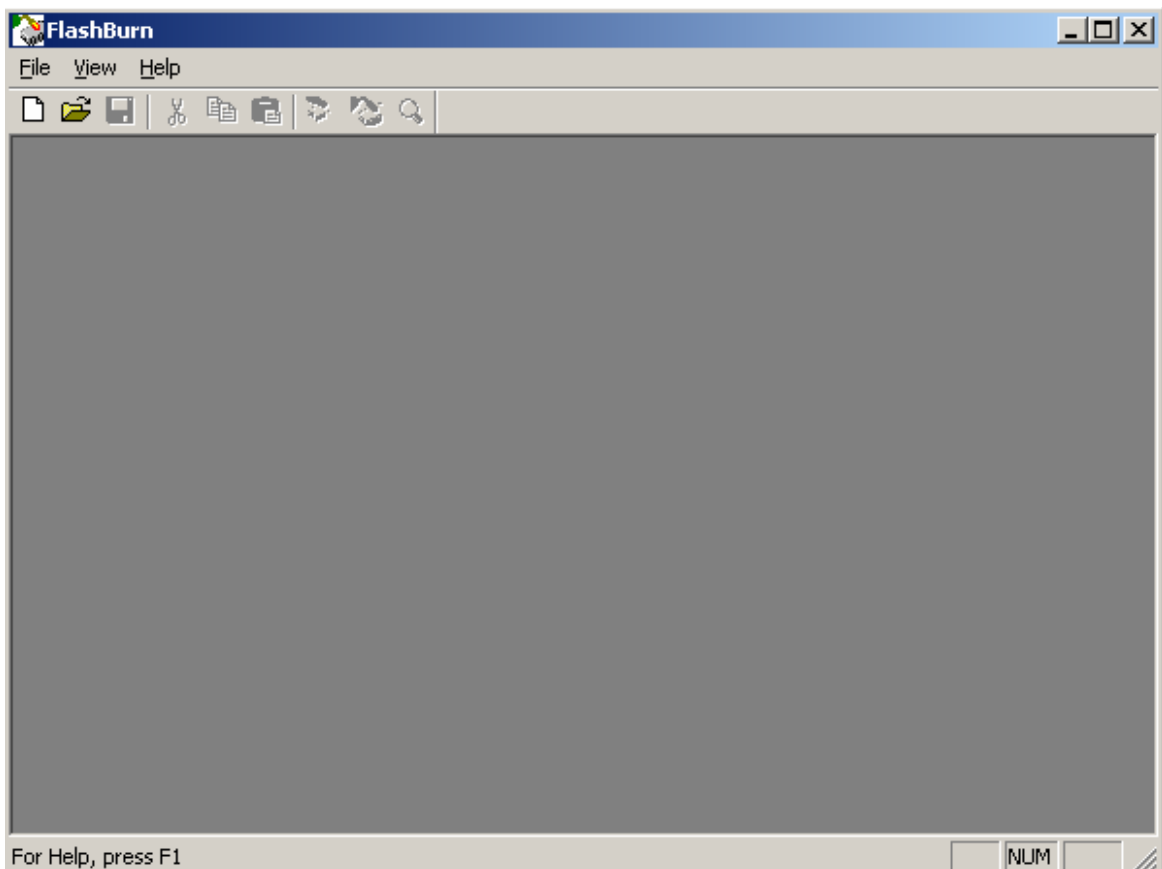


图 2-7-2 flashburn 图

点击 file 菜单下 open，选择 icetek-vc5509-c.cdd 文件。

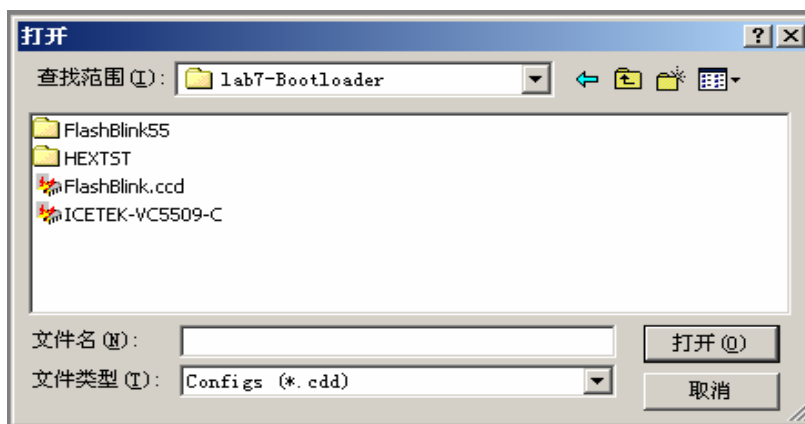


图 2-7-3 选择文件

6. 弹出图 2-7-4。

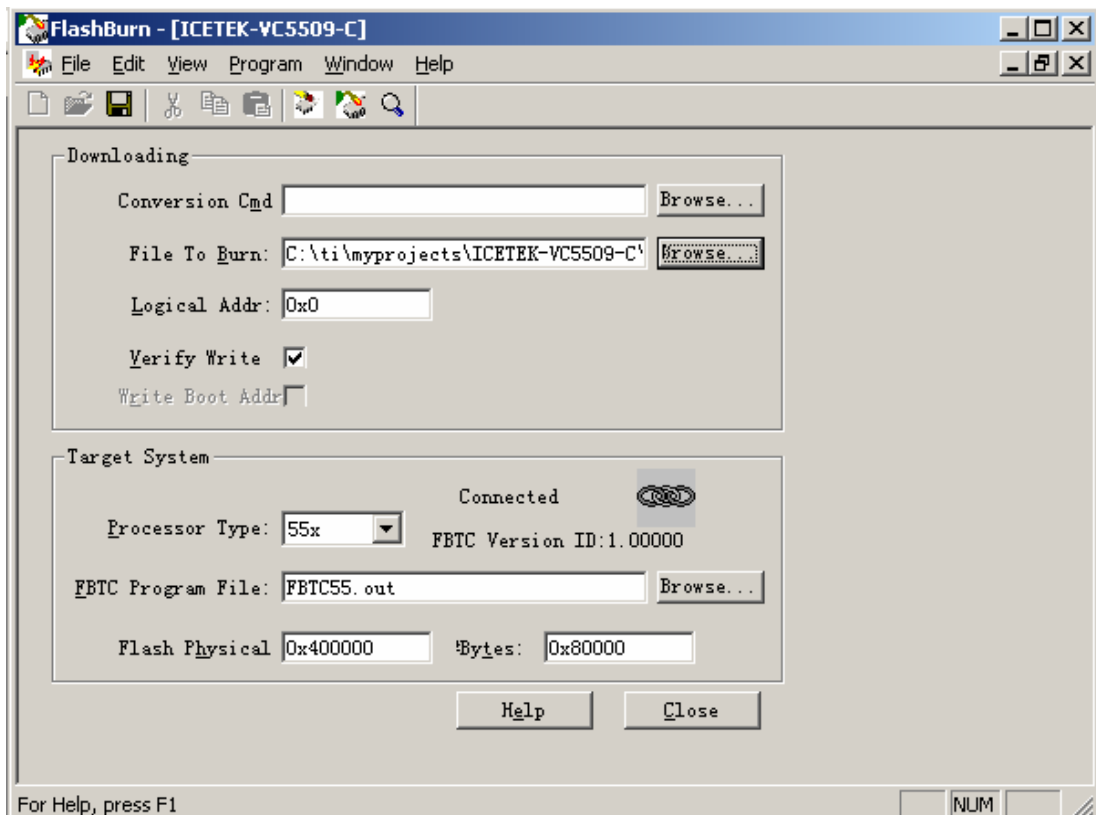


图 2-7-4 烧写 flash 界面

修改 file to burn：中的文件为 FlashBlink.hex。（通过 browse 按钮）

注意：target system 中 显示 connected 才可以完成对 flash 的操作。

7. 点击 program 菜单下的 erase flash，等待擦除结束。
8. 点击 program 菜单下的 program flash，等待擦除结束。
9. 测试自启动

拔掉仿真电缆，；重新打开电源；按一下 ICETEK – VC5509-C 评估板上复位按钮，观察板上指示灯亮；表明烧入 Flash 的程序正在运行；按一下板上复位按钮，程序将从新运行。

10. 关闭电源，将 DSP 系统板上 MP/MC 设置到“OFF”，即 MP 方式。

五. 实验结果

在脱离计算机和仿真器的情况下，自启动程序正常工作在 MC 方式。

六．问题与思考

实验八：异步串口通信实验

一. 实验目的

1. 了解 TL16C550 异步串行通信收发器。
2. 学会设置异步串行通信接口进行通信。
3. 了解 ICETEK-VC5509-C 评估板上 DSP 对 TL16C550 的连接设计。
4. 学习设计异步通信程序。

二. 实验设备

计算机， ICETEK 仿真器+ICETEK-VC5509-C 评估板+相关连线及电源。

三. 实验原理

1. TL16C550 异步串行通信收发器

TL16C550 有 11 个寄存器,通过 A2~A0 和线路控制寄存器中的 DLAB 位对它们进行寻址.

表 2-8-1 TL16C550 的寄存器

寄存器	DLAB	A2	A1	A0	地址	操作
接收缓冲寄存器 RBR	0	0	0	0	00H	只读
发送缓冲寄存器 THR	0	0	0	0	00H	只写
中断始能寄存器 IER	0	0	0	1	01H	读/写
中断标志寄存器 IIR	X	0	1	0	01H	只读
FIFO 控制寄存器 FCR	X	0	1	0	01H	只读
线路控制寄存器 LCR	X	0	1	1	03H	读/写
MODEM 控制寄存器 MCR	X	1	0	0	04H	读/写
线路状态寄存器 LSR	X	1	0	1	05H	读/写
MODEM 状态寄存器 MSR	X	1	1	0	06H	读/写
暂存寄存器 SCR	X	1	1	1	07H	读/写
低位除数寄存器 DLL	1	0	0	0	00H	读/写
低位除数寄存器 DLM	1	0	0	1	01H	读/写

1) 线路控制寄存器:

表 2-8-2 线路控制寄存器

D7	D6	D5	D4	D3	D2	D1	D0
DLAB	BREAK	SPB	EPS	PEN	STB	WLS1	WLS0

WLS1 WLS0:设置数据长度:

0 0 :5 位

0 1 :6 位

1 0 :7 位

1 1 :8 位

STB :设置停止位个数

0: 一个停止位

1: 1.5 个停止位(5 位数据长度时),2 个停止位(6,7,8 位数据长度时)

PEN:奇偶校验使能

0: 奇偶校验无效

1: 奇偶校验有效

EPS: 奇偶校验选择

0: 奇校验

1: 偶校验

DLAB:寄存器访问选择

0:访问其余寄存器

1:访问除数和功能切换寄存器

2) 线路状态寄存器

表 2-8-3 线路状态寄存器

D7	D6	D5	D4	D3	D2	D1	D0
FERR	TEMT	THRE	BI	FE	PE	OE	DR

DR:接收数据准备好标志

0: 接收数据缓冲器空

1: 接收数据缓冲器中有数据

OE: 溢出错误标志(上一个接收数据被当前接收数据覆盖)

0: 无溢出

1: 有溢出

PE: 奇偶校验错误标志

0: 无奇偶校验错误

1: 有奇偶校验错误

THRE:发送保持寄存器空标志

0: 非空

1: 空

TEMT:发送器空标志

0: 发送保持寄存器和发送移位寄存器非空

1: 发送保持寄存器和发送移位寄存器都空

3) 中断使能寄存器

表 2-8-4 中断使能寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	EMSI	ELSI	ETHREI	ERDAI

ERDAI: 接收中断使能

0 : 接收中断禁止

1 : 接收中断使能

ETHREI: 发送中断使能

0 : 接收中断禁止

1 : 接收中断使能

ELSI : 接收错误中断使能

0 : 接收错误中断禁止

1 : 接收错误中断使能

EMSI : MODEM 中断使能

0 : MODEM 中断禁止

1 : MODEM 中断使能

4) 中断标志寄存器

表 2-8-5 中断标志寄存器

中断标志寄存器				中断设置与清除			
D3	D2	D1	D0	优先级	中断类型	中断源	中断清除
0	0	0	1	---	无中断	无中断	
0	1	1	0	最高	接收错误	溢出,奇偶,帧错误	读线路状态寄存器
0	1	0	0	第二	接收	接收缓冲器中数据	读接受缓冲器
1	1	0	0	第二	FIFO 超时	FIFO 超时	读接收缓冲器
0	0	1	0	第三	发送	发送保持寄存器空	写发送保持寄存器
0	0	0	0	第四	MODEM	MODEM 状态	读状态寄存器

5) 设置波特率

TL16C550 的波特率可通过除数寄存器 DLM,DLL 来设置,除数寄存器值和波特率之间的换算公式如下:除数值=输入频率÷(波特率×16),TL16C550 的输入频率为:3.6864MHz,

表 2-8-6 波特率和除数之间的关系

波特率	高位除数寄存器 DLM	低位除数寄存器 DLL
1200	00H	C0H
2400	00H	60H
4800	00H	30H
9600	00H	18H
19200	00H	0CH
38400	00H	06H

6) 串口标准

RS232 标准

2.ICETEK-VC5509-C 评估板异步串口设计

16C550 芯片的控制寄存器映射到 5416 的 0x400200 空间,通过 CPLD 进行译码选通。

另外需要加上驱动电路。驱动电路主要完成将 DSP 输出的 0-3.3V 电平转换成异步串口电平的工作。转换电平的工作由 MAX232 芯片完成。

3. 串行通信接口设置

*CPU 进行串行通信时可以采用两种方式，一种是轮询方式，即 CPU 不断查询串口状态进行接收和发送，缺点是占用 CPU 时间太多；另一种是中断方式，接收和发送都可以产生中断信号，这样 CPU 可以在完成其他一些工作的同时进行串行通信。

*串行通信接口波特率计算

内部生成的串行时钟由系统时钟 SYSCLK 频率和波特率选择寄存器决定。串行通信接口使用 16 位波特率选择寄存器，数据传输的速度可以被编程为 65000 多种不同的方式。

不同通信模式下的串行通信接口异步波特率由下列方法决定：

-BRR=1—65535 时的串行通信接口异步波特率：

$$\text{SCI 异步波特率} = \text{SYSCLK} / [(BRR+1)*8]$$

$$\text{其中, } BRR = \text{SYSCLK} / (\text{SCI 异步波特率} * 8) - 1 ;$$

-BRR=0 时的串行通信接口异步波特率：

$$\text{SCI 异步波特率} = \text{SYSCLK} / 16$$

这里 BRR 等于波特率选择寄存器的 16 位值。

4. TL16C550 编程步骤如下:

- 1) 初始化(包括:设置数据长度,有无奇偶校验,奇/偶校验选择,停止位个数及波特率),
- 2) 开相应的中断,等中断
- 3) 在中断服务程序中完成数据的发送和接收.

四. 实验步骤

1. 实验准备

. 连接设备

关闭计算机电源。

如使用 PP 型仿真器则用附带的并口连线连接计算机并口和仿真器相应接口。

用附带的串行通信电缆连接计算机 COM 端口和 ICETEK-VC5509-C 评估板上九针接头 DB1。

. 开启设备

打开计算机电源。

打开实验箱电源开关，注意 ICETEK-VC5509-C 评估板上指示灯 D10 亮。

如使用 USB 型仿真器用附带的 USB 电缆连接计算机和仿真器相应接口，注意仿真器上两个指示灯均亮。

. 设置 Code Composer Studio 为 Emulator 方式：

参见“Code Composer Studio 入门实验”之四.3。

. 启动 Code Composer Studio 2.0

2. 打开工程，浏览程序，工程目录为 ICETEK-VC5509-C \Lab8Serial

3. 编译并下载程序

4. 运行“串口调试助手”

利用桌面上“我的电脑”，找到 ICETEK-VC5509-C \Lab8Serial 目录中的程序“串口调试助手 V2.0B.exe”，双击它启动；设置“串口调试助手”的串行端口为实际连接的计算机 COM 端口，设置波特率为 9600，设置传输方式为 8 位、无校验、1 个停止位。

5. 运行程序观察结果

运行程序后，切换窗口到“串口调试助手”；在“串口调试助手”的接收窗口中可看到 DSP 通过 SCI 发送来的“Hello PC!,Over|”字样；在“发送的字符/数据”栏中输入一些要发送到 DSP 的字符串，以“.”字符结尾；然后单击“手动发送”按钮；DSP 在接收到 PC 机的信息后会自动进行回答。

6. 结束程序运行退出。

五. 实验结果

通过 DSP 传送到 PC 机上的信息，可以看出：串行通信接口正确工作。

六. 问题与思考

请考虑用中断方式设计程序完成异步串行通信。

实验九：卷积算法实验

一．实验目的

- 1、掌握用窗函数法设计卷积算法的原理和方法；
- 2、熟悉卷积算法特性；
- 3、了解各种窗函数对卷积算法的影响。

二．实验设备

计算机，Code Composer Studio 2.0 for 'C5000 系统。

三．实验原理

1.卷积的基本原理和公式

卷积和：对离散系统“卷积和”也是求线性时不变系统输出响应(零状态响应)的主要方法。

$$Y(n) = \sum_{m=-\infty}^{\infty} X(m)h(n-m) = X(n) * h(n)$$

卷积和的运算在图形表示上可分为四步：

- 1) 翻褶 先在哑变量坐标 M 上作出 $x(m)$ 和 $h(m)$ ，将 $m=0$ 的垂直轴为轴翻褶成 $h(-m)$ 。
- 2) 移位 将 $h(-m)$ 移位 n ，即得 $h(n-m)$ 。当 n 为正整数时，右移 n 位。当 n 为负整数时，左移 n 位。
- 3) 相乘 再将 $h(n-m)$ 和 $x(m)$ 的相同 m 值的对应点值相乘。
- 4) 相加 把以上所有对应点的乘积叠加起来，即得 $y(n)$ 值。

依上法，取 $n=\dots,-2,-1,0,1,2,3,\dots$ 各值，即可得全部 $y(n)$ 值。

2.程序流程图

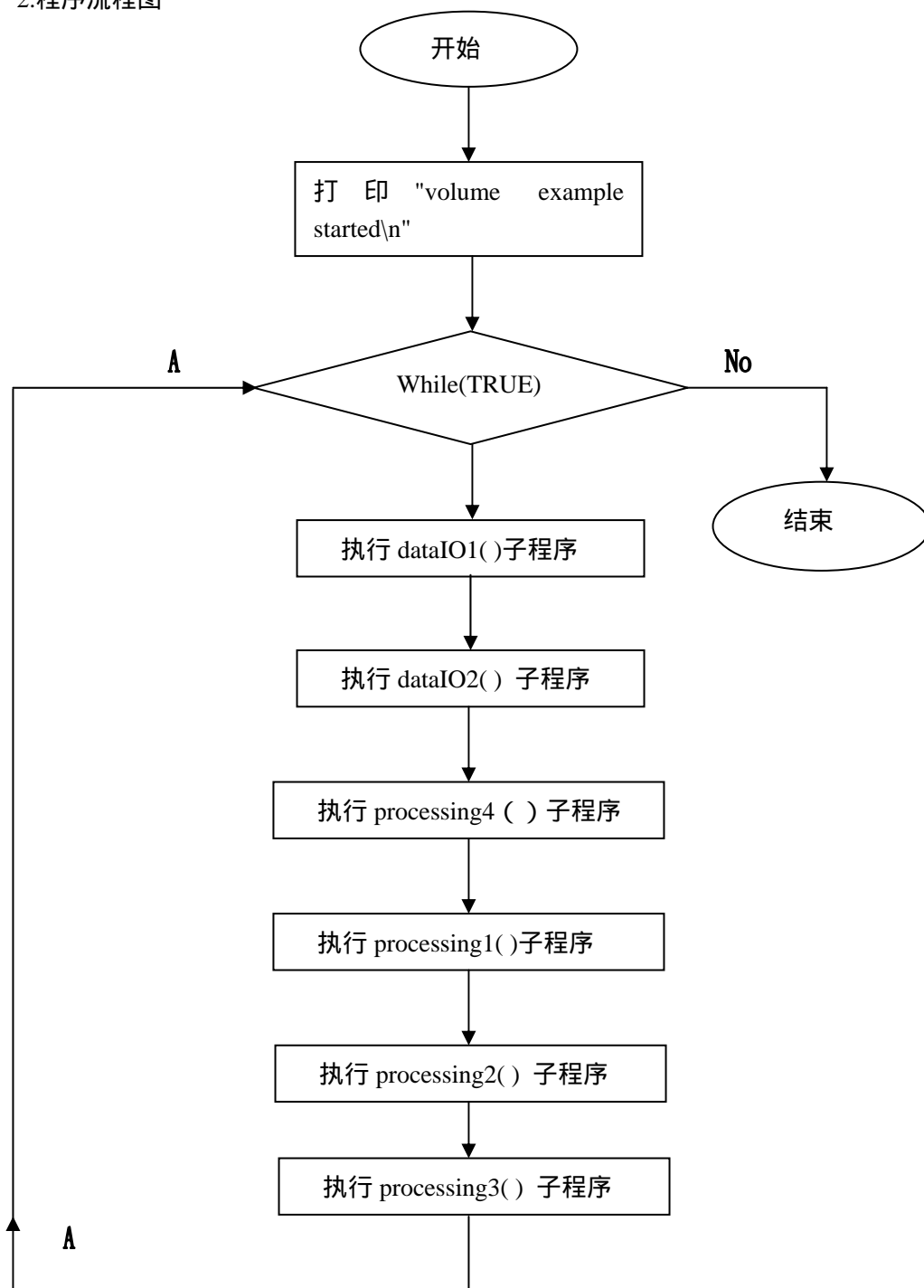


图 2-9-1 程序流程图

3.程序的自编函数及其功能

1) processing1(int *input2, int *output2)

调用形式：processing1(int *input2, int *output2)

参数解释：input2、output2 为两个整型指针数组。

返回值解释：返回了一个“TREN”，让主函数的 while 循环保持连续。

功能说明：对输入的 input2 buffer 波形进行截取 m 点，再以零点的 Y 轴为对称轴进行翻褶，把生成的波形上的各点的值存入以 OUTPUT2 指针开始的一段地址空间中。

2) processing2(int *output2, int *output3)

调用形式：processing2(int *output2, int *output3)

参数解释：output2、output3 为两个整型指针数组。

返回值解释：返回了一个“TREN”，让主函数的 while 循环保持连续。

功能说明：对输出的 output2 buffer 波形进行作 n 点移位，然后把生成的波形上的各点的值存入以 OUTPUT3 指针开始的一段地址空间中。

3) processing3(int *input1, int *output2, int *output4)

调用形式：processing3(int *input1, int *output2, int *output4)

参数解释：output2、output4、input1 为三个整型指针数组。

返回值解释：返回了一个“TREN”，让主函数的 while 循环保持连续。

功能说明：对输入的 input2 buffer 波形和输入的 input1 buffer 作卷积和运算，然后把生成的波形上的各点的值存入以 OUTPUT4 指针开始的一段地址空间中。

4) processing4(int *input2, int *output1)

调用形式：processing4(int *input2, int *output1)

参数解释：output1、input2 为两个整型指针数组。

返回值解释：返回了一个“TREN”，让主函数的 while 循环保持连续。

功能说明：对输入的 input2 buffer 波形截取 m 点，然后把生成的波形上的各点的值存入以 OUTPUT1 指针开始的一段地址空间中。

四、实验步骤

1、实验准备

(1) 连接设备

关闭计算机电源。

如使用 PP 型仿真器则用附带的并口连线连接计算机并口和仿真器相应接口。

(2) 开启设备

打开计算机电源。

如使用 USB 型仿真器用附带的 USB 电缆连接计算机和仿真器相应接口，注意仿真器上两个指示灯均亮。

(3) 设置 Code Composer Studio 为 Simulator 方式：

参见“Code Composer Studio 入门实验”之四.2。

(4) 启动 Code Composer Studio 2.0

2. 打开工程，浏览程序，工程目录为 ICETEK-VC5509-C \Lab9-Convolve

3. 编译并下载程序

4. 设置输入数据文件

请在 c 程序中的如下两行上设置 probe point :

```
dataIO1();
```

```
dataIO2();
```

设置方法是把光标指示到这一行上,按鼠标右键,从显示的菜单上分别选择 probe point。

再在 c 程序的“dataIO1();”和“dataIO2();”行上设置 break point。

5. 打开观察窗口

-选择菜单“View”、“Graph”、“Time/Frequency...”进行如下设置：

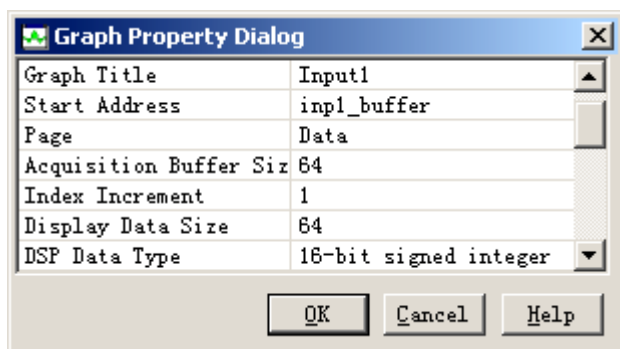


图 2-9-2 Graph Property Dialog 窗口 1

-选择菜单“View”、“Graph”、“Time/Frequency...”进行如下设置：

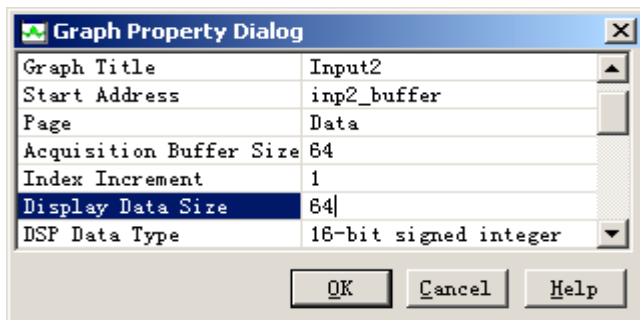


图 2-9-3 Graph Property Dialog 窗口 2

-选择菜单“View”、“Graph”、“Time/Frequency...”进行如下设置：

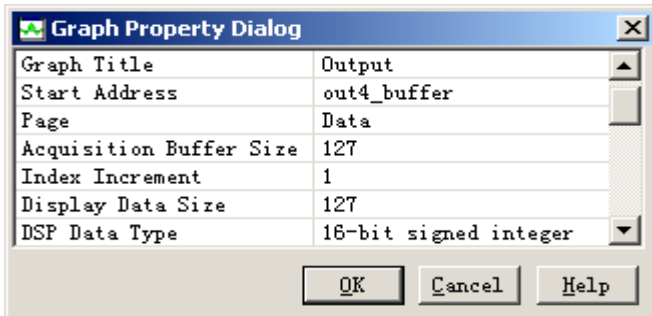


图 2-9-4 Graph Property Dialog 窗口 3

-在弹出的三个图形窗口中单击鼠标右键，选择“Clear Display”。

6. 设置波形输入文件

-选择“File”菜单中的“File I/O...”，打开“File I/O”窗口；单击“Add File”按钮，在“File Input”窗口中选择 ICETEK-VC5509-C \Lab9-Convolve 目录下的 sin.dat 文件，单击“打开”按钮；在“Address”项中输入 inp1_buffer，在“Length”项中输入 32，在“Warp Around”项前加上选择标记，单击“Add Probe Point”按钮；

-在“Break/Probe/Profile Points”窗口中单击“Probe Point”列表中的“Convolve.c line52 → No Connection”，再单击“Connect”项尾部的展开按钮，在显示的展开式列表中选择列表末尾的“FILE IN:C:\.\SIN.DAT”，单击“Replace”按钮，单击“确定”按钮。

-在“File I/O”窗口中单击“确定”，完成设置。

-选择“File”菜单中的“File I/O...”，打开“File I/O”窗口；单击“Add File”按钮，在“File Input”窗口中选择 ICETEK-VC5509-C \Lab9-Convolve 目录下的 sin.dat 文件，单击“打开”按钮；在“Address”项中输入 inp2_buffer，在“Length”项中输入 32，在“Warp Around”项前加上选择标记，单击“Add Probe Point”按钮；

-在“Break/Probe/Profile Points”窗口中单击“Probe Point”列表中的“Convolve.c line53 → No Connection”，再单击“Connect”项尾部的展开按钮，在显示的展开式列表中选择列表末尾的“FILE IN:C:\.\SIN.DAT”，单击“Replace”按钮，单击“确定”按钮。

-在“File I/O”窗口中单击“确定”，完成设置。

2. 运行程序，观察结果

-按 F5 键运行程序，待程序停留在软件断点；观察刚才打开的三个图形窗口，其中显示的是输入和输出的时域波形；

-观察频域波形：在各图形窗口中单击鼠标右键，选择“Properties...”，在“Graph Property Dialog”窗口中的第 1 项“Display Type”项中选择“FFT Magnitude”，单击“OK”完成；这时图形窗口中显示波形的频域图。（也可再打开显示频域图的窗口）

-验算结果：在各频域窗口中的波形上单击鼠标左键，将光标停到统一的位置（通过观察窗口状态栏中的第 1 个浮点数表示其坐标值），读取状态栏中的第 2 个浮点数，为卷积计算的输入和输出结果，请验算：Output Input1*Input2。

3. 将输入波形文件改成其他波形：选择“File”、“File I/O...”，将 2 个文件删除，将第 1 个文件换成 SIN11.DAT，将第 2 个文件换成 SIN22.DAT；将两个文件“Length”项中输入 64；把“VOLUME.C”文件中的输入“int sk=32;”改为“int sk= 64”，其他不变。再按 F5 运行，停止后观察波形。

4. 将第 2 个输入波形改成 SIN33.DAT，观察卷积运算后的波形。

5. 将第 2 个输入波形改成 SIN44.DAT，观察卷积运算后的波形。

6. 将第 2 个输入波形改成方波.DAT，观察卷积运算后的波形。

五、试验结果

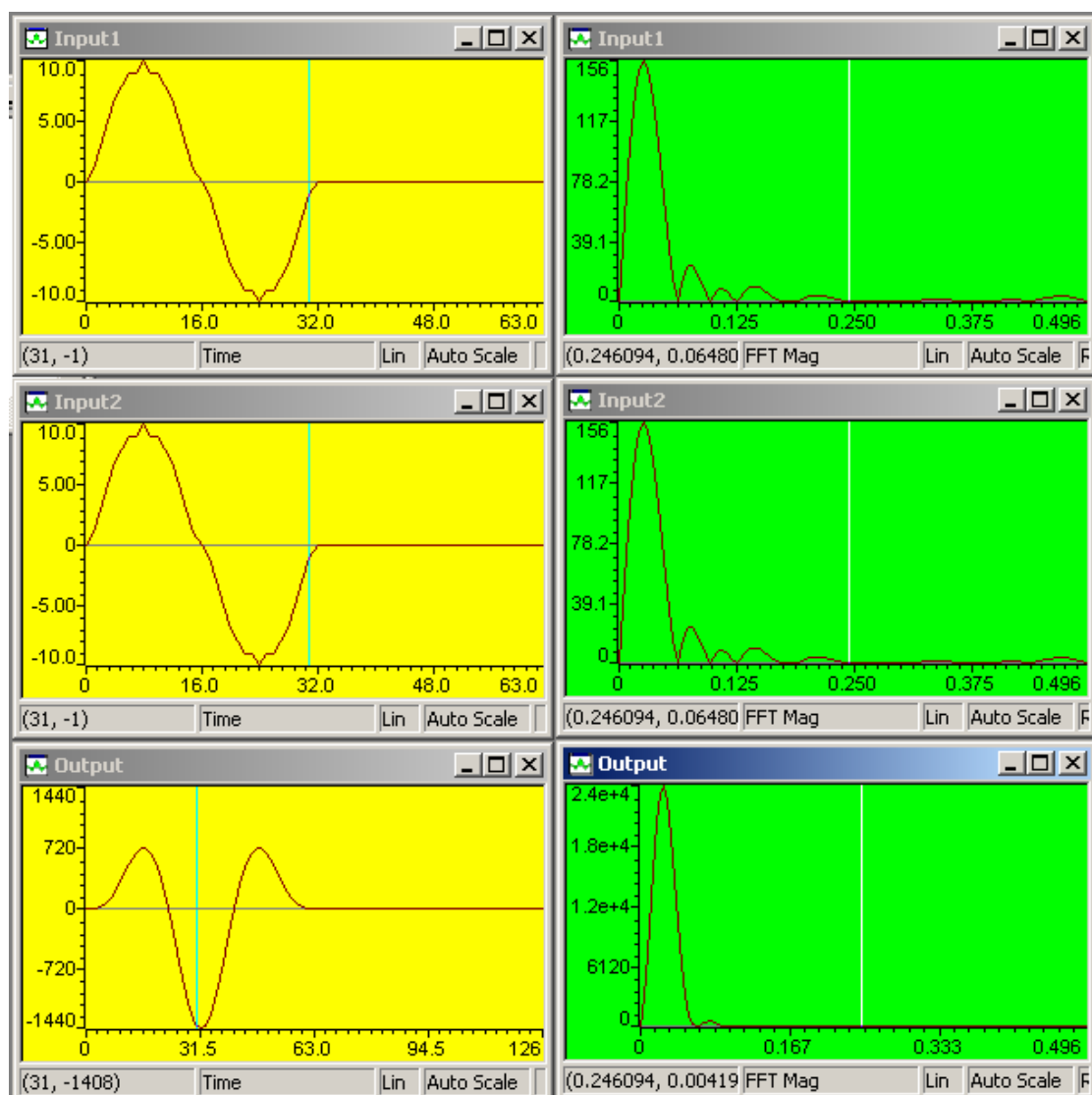


图 2-9-5 实验第 7 步的结果图

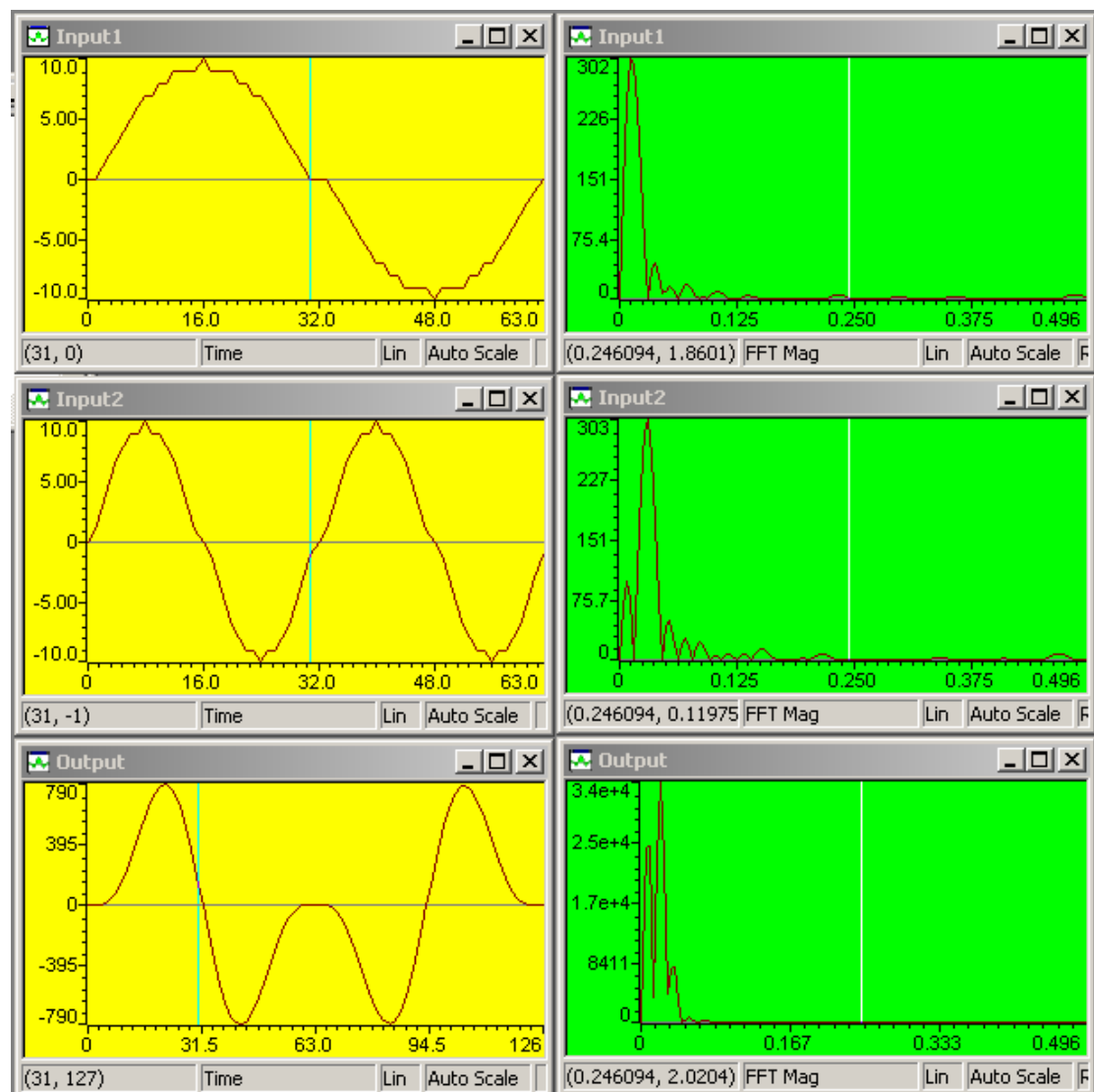


图 2-9-6 实验第 8 步的结果图

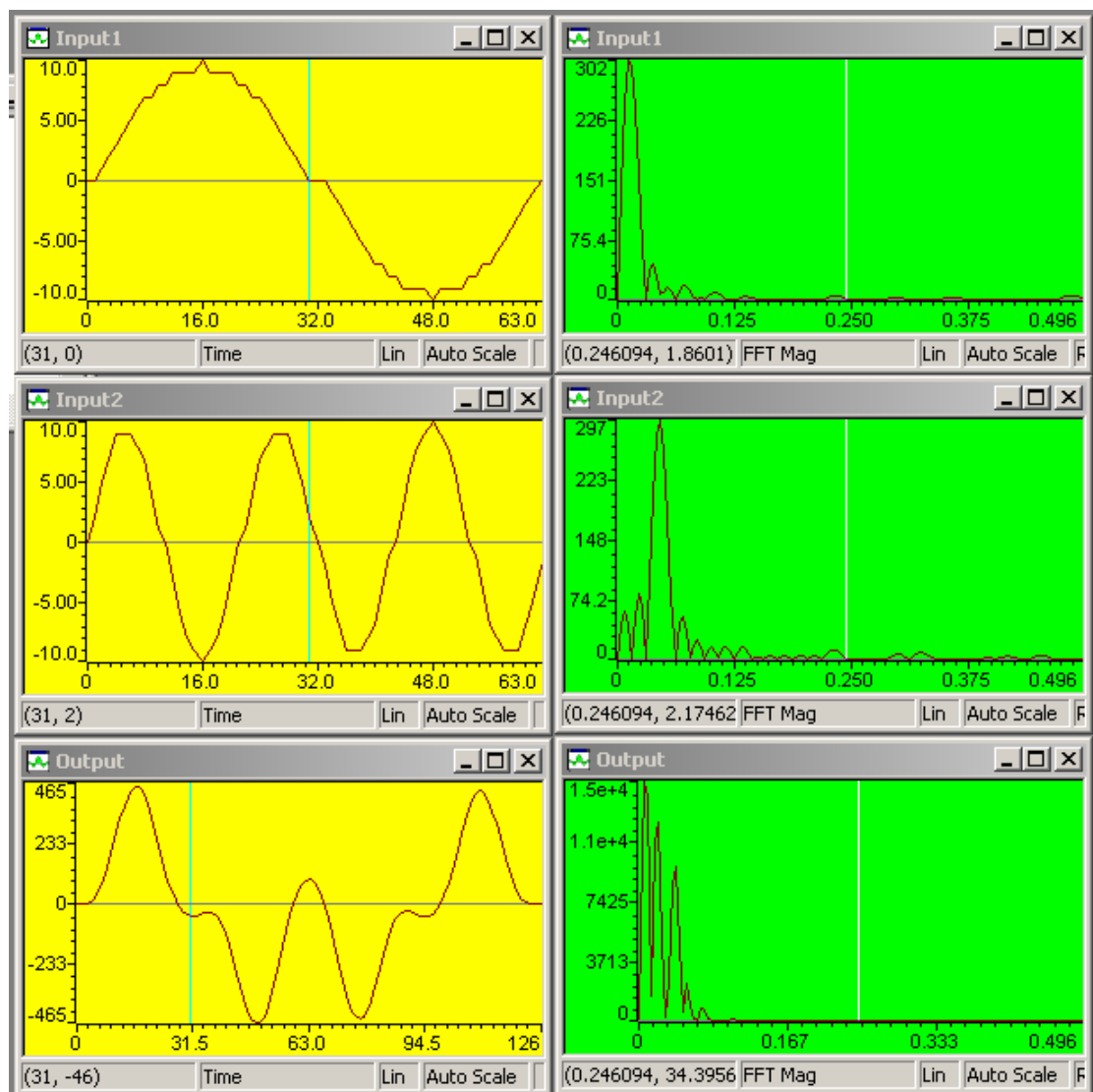


图 2-9-7 实验第 9 步的结果图

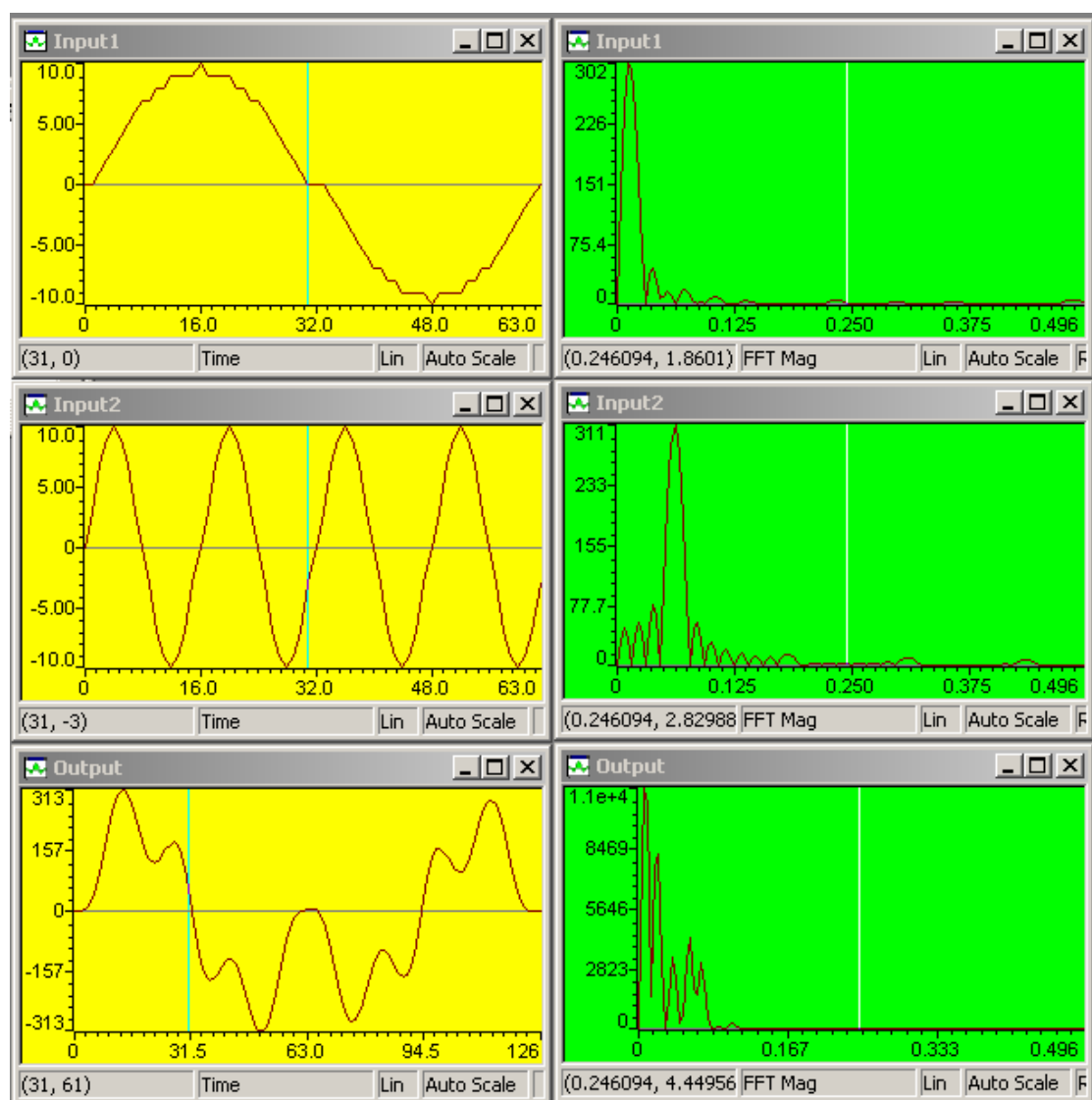


图 2-9-8 实验第 10 步的结果图

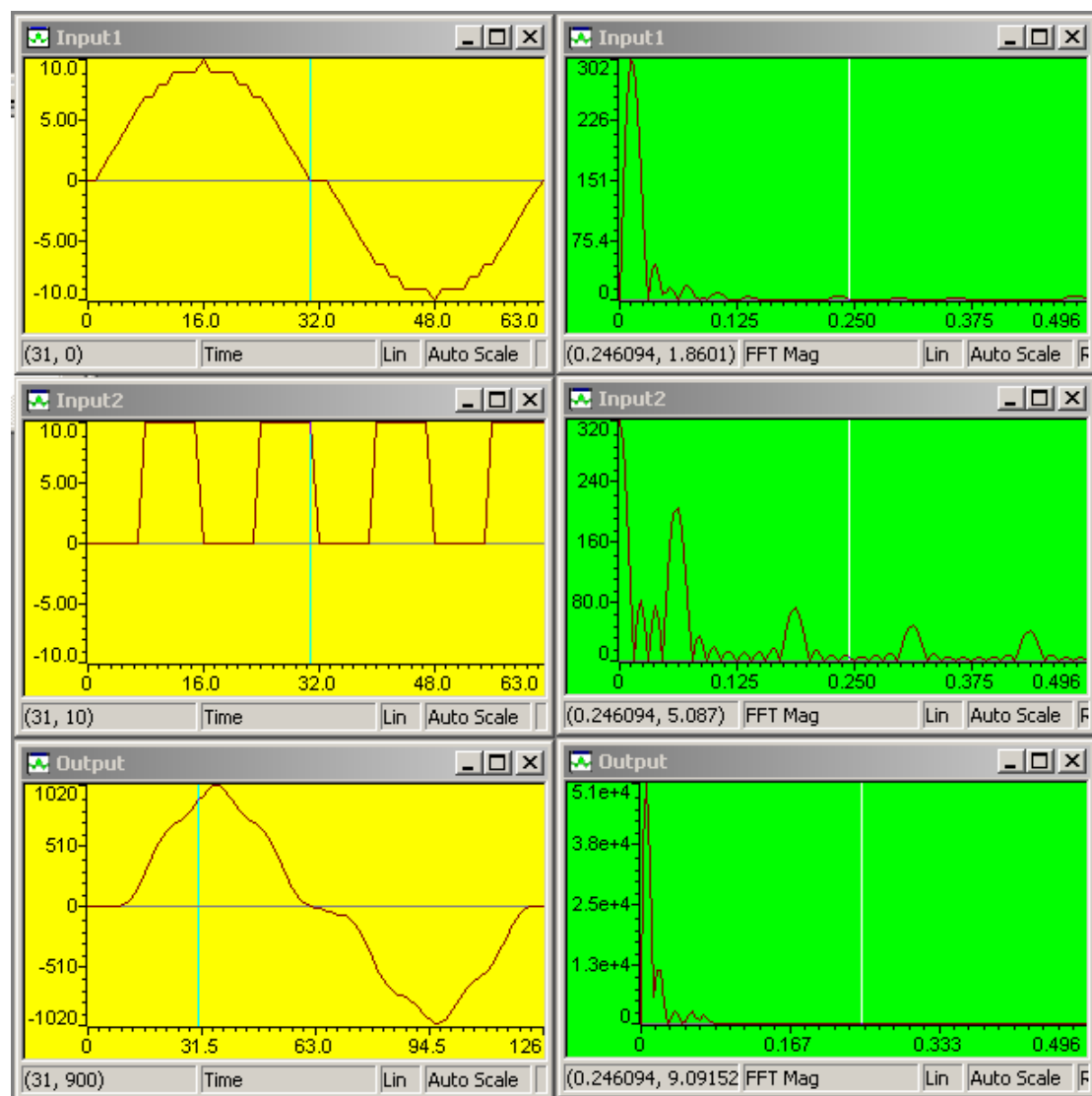


图 2-9-9 实验第 11 步的结果图：图 5

表 2-9-1

名称	Inp1 上取样点值 (样点位置)	Inp2 上取样点值 (样点位置)	Out4 上取样点值(样 点位置)	标准值
图 2-9-5	101.401(0.0117188)	101.401(0.011718 8)	10279.34(0.0117188)	10282.16280
图 2-9-6	156.796(0.234375)	217.879(0.234375)	33642.7(0.234375)	34162.555684
图 2-9-7	156.796(0.234375)	83.1131(0.234375)	12414(0.234375)	13031.8026176
图 2-9-7	301.704(0.0117188)	1.90995(0.011718 8)	14069.5(0.0117188)	13359.03
图 2-9-8	156.796(0.234375)	56.0769(0.234375)	8248.46(0.234375)	8792.634
图 2-9-8	301.704 (0.0117188)	35.2126(0.011718 8)	10663(0.0117188))	10623.78227
图 2-9-9	296.69156.796(0.234375)	81.743 (0.234375)	12593.0(0.234375)	12817
图 2-9-9	301.704 (0.0117188)	100.378(0.011718 8)	31667.2(0.0117188)	30284.4

-图表分析

输入图形频域图形采样通过频域采样取值比较，卷积后的结果与标准值只有很小的误差。所以说卷积实验结果正确，卷积程序正确无误。

六、问题与思考

实验十：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 FIR 数字滤波器特性。
- 3.了解各种窗函数对滤波器特性的影响。

二.实验设备

计算机，Code Composer Studio 2.0 for 'C5000 系统。

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、椭圆滤波器、贝塞尔滤波器）。
- 3.数字滤波器系数的确定方法。
- 4.根据要求设计低通 FIR 滤波器

要求：通带边缘频率 10kHz，阻带边缘频率 22kHz，阻带衰减 75dB，采样频率 50kHz。

设计：

-过渡带宽度=阻带边缘频率-通带边缘频率=22-10=12kHz

-采样频率：

$$f_1 = \text{通带边缘频率} + (\text{过渡带宽度})/2 = 10000 + 12000/2 = 16\text{kHz}$$

$$1 = 2 \quad f_1/f_s = 0.64$$

-理想低通滤波器脉冲响应：

$$h_1[n] = \sin(n-1)/n/ = \sin(0.64-n)/n/$$

-根据要求，选择布莱克曼窗，窗函数长度为：

$$N = 5.98f_s/\text{过渡带宽度} = 5.98*50/12 = 24.9$$

-选择 N=25，窗函数为：

$$w[n] = 0.42 + 0.5\cos(2\pi n/24) + 0.8\cos(4\pi n/24)$$

-滤波器脉冲响应为：

$$h[n] = h_1[n]w[n] \quad |n| \leq 12$$

$$h[n] = 0 \quad |n| > 12$$

-根据上面计算，各式计算出 $h[n]$ ，然后将脉冲响应值移位为因果序列。

-完成的滤波器的差分方程为：

$$\begin{aligned}
 y[n] = & -0.001x[n-2] - 0.002x[n-3] - 0.002x[n-4] + 0.01x[n-5] \\
 & - 0.009x[n-6] - 0.018x[n-7] - 0.049x[n-8] - 0.02x[n-9] \\
 & + 0.11x[n-10] + 0.28x[n-11] + 0.64x[n-12] \\
 & + 0.28x[n-13] - 0.11x[n-14] - 0.02x[n-15] \\
 & + 0.049x[n-16] - 0.018x[n-17] - 0.009x[n-18] + 0.01x[n-19] \\
 & - 0.002x[n-20] - 0.002x[n-21] + 0.001x[n-22]
 \end{aligned}$$

四.实验步骤

- 1.实验准备：设置 Code Composer studio 为 Simulator 方式并启动。（参考实验一来设置）
- 2.打开工程，浏览程序，工程目录为 ICETEK-VC5509-C\lab10-fir
- 3.编译并下载程序
- 4.打开观察窗口

*选择菜单“View”、“Graph”、“Time/Frequency...”进行如下设置：

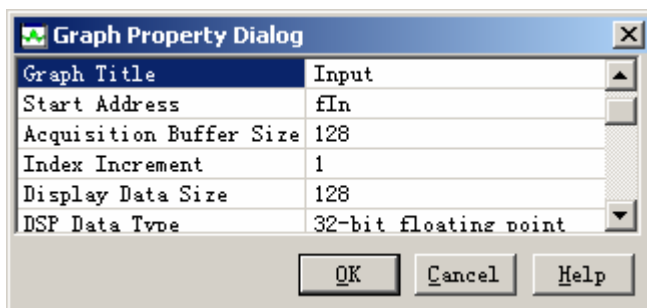


图 2-10-1 设置 1

*选择菜单“View”、“Graph”、“Time/Frequency...”进行如下设置：

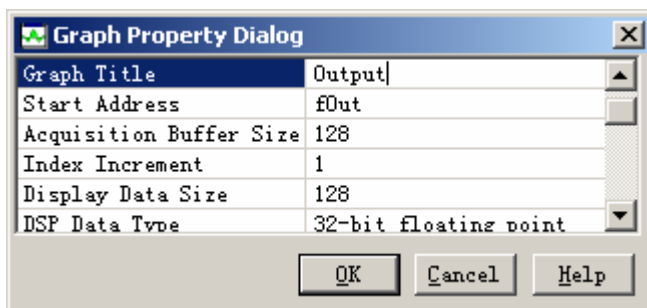


图 2-10-2 设置 2

-在弹出的图形窗口中单击鼠标右键，选择“Clear Display”。

5.设置断点

在标号“ nOut=0; /* 请在此句上设置软件断点 */ ”语句设置软件断点（Toggle breakpoint）。

6.运行并观察结果

选择“Debug”菜单的“Run”项，或按 F5 键运行程序。程序运行后停在断点上。

观察 “Input”、“Output” 窗口中时域图形 2-10-4；观察滤波效果。

鼠标右键单击 “Input” 和 “Output” 窗口，选择 “Properties...” 项，设置 “Display Type” 为 “FFT Magnitude”，再单击 “OK” 按钮结束设置。看图 2-10-4

观察 “Input”、“Output” 窗口中频域图形；理解滤波效果。

7. 停止程序运行并退出。

五. 实验结果

* 观察滤波器系数频谱，为低通滤波器

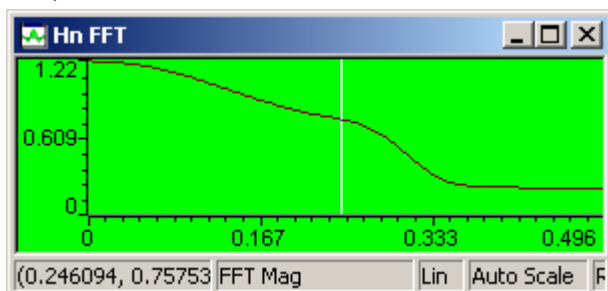


图 2-10-3 滤波器系数频谱

输入波形为一个低频率的正弦波与一个高频的余弦波叠加而成。如图：

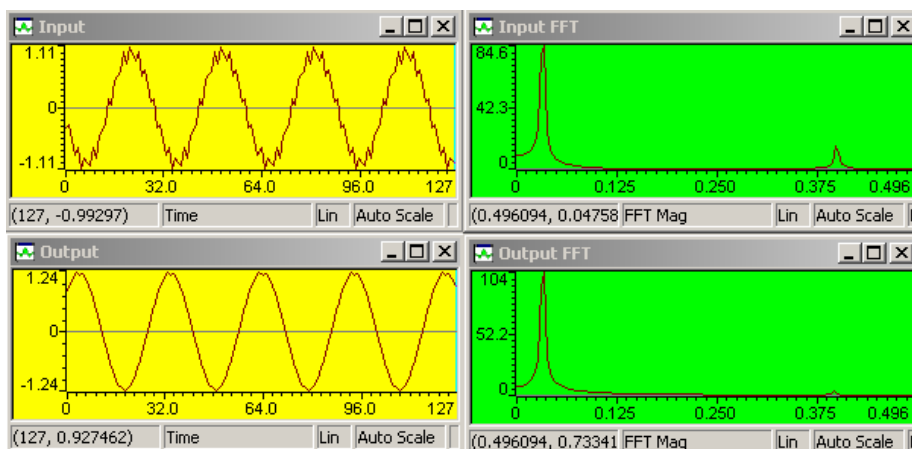


图 2-10-4 滤波前后的图形

通过观察频域和时域图，得知：输入波形中的低频波形通过了滤波器，而高频部分则被衰减。

六. 问题与思考

试设计合适的高通滤波参数滤掉实验的输入波形中的低频信号。

实验十一：无限冲击响应滤波器（IIR）算法实验

一.实验目的

- 1.掌握设计 IIR 数字滤波器的原理和方法。
- 2.熟悉 IIR 数字滤波器特性。
- 3.了解 IIR 数字滤波器的设计方法。

二.实验设备

计算机，Code Composer Studio 2.0 for 'C5000 系统。

三.实验原理

- 1.无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、椭圆滤波器、贝塞尔滤波器）。
- 3.数字滤波器系数的确定方法。
- 4.根据要求设计低通 IIR 滤波器

要求：低通巴特沃斯滤波器在其通带边缘 1kHz 处的增益为-3dB，12kHz 处的阻带衰减为 30dB，采样频率 25kHz。

设计：

-确定待求通带边缘频率 f_{p1} Hz、待求阻带边缘频率 f_{s1} Hz 和待求阻带衰减 $-20\log s$ dB。

模拟边缘频率为： $f_{p1}=1000\text{Hz}$ ， $f_{s1}=12000\text{Hz}$

阻带边缘衰减为： $-20\log s=30\text{dB}$

-用式 $\omega = 2\pi f/fs$ 把由 Hz 表示的待求边缘频率转换成由弧度表示的数字频率，得到 ω_{p1} 和 ω_{s1} 。

$\omega_{p1}=2\pi f_{p1}/f_s=2\pi \cdot 1000/25000=0.08$ 弧度

$\omega_{s1}=2\pi f_{s1}/f_s=2\pi \cdot 12000/25000=0.96$ 弧度

-计算预扭曲模拟频率以避免双线性变换带来的失真。

由 $\omega = 2\pi f \tan(\omega_a/2)$ 求得 ω_{p1} 和 ω_{s1} ，单位为弧度/秒。

$\omega_{p1}=2\pi f_s \tan(\omega_{p1}/2)=6316.5$ 弧度/秒

$\omega_{s1}=2\pi f_s \tan(\omega_{s1}/2)=794727.2$ 弧度/秒

-由已给定的阻带衰减 $-20\log s$ 确定阻带边缘增益 s 。

因为 $-20\log s=30$ ，所以 $\log s=-30/20$ ， $s=0.03162$

-计算所需滤波器的阶数：

$$n \geq \frac{\log(\frac{1}{\delta_s^2} - 1)}{2\log(\frac{\omega_{s1}}{\omega_{p1}})} = \frac{\log(\frac{1}{(0.03162)^2} - 1)}{2\log(\frac{794727.2}{6316.5})} = 0.714$$

因此，一阶巴特沃斯滤波器就足以满足要求。

-一阶模拟巴特沃斯滤波器的传输函数为：

$$H(s) = \omega_{p1} / (s + \omega_{p1}) = 6316.5 / (s + 6316.5)$$

由双线性变换定义 $s = 2fs(z-1)/(z+1)$ 得到数字滤波器的传输函数为：

$$H(z) = \frac{6316.5}{50000 \frac{z-1}{z+1} + 6316.5} = \frac{0.1122(1+z^{-1})}{1-0.7757z^{-1}}$$

因此，差分方程为： $y[n] = 0.7757y[n-1] + 0.1122x[n] + 0.1122x[n-1]$

四.实验步骤

- 1.实验准备：设置 Code Composer studio 为 Simulator 方式并启动。（参考实验一来设置）
- 2.打开工程，浏览程序，工程目录为 ICETEK-VC5509-C\lab11-iir
- 3.编译并下载程序
- 4.打开观察窗口

*选择菜单“View”、“Graph”、“Time/Frequency...”进行如下设置：

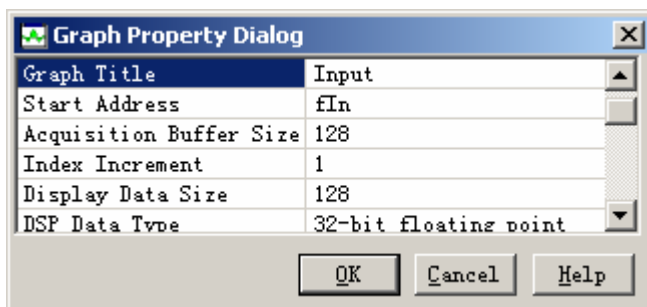


图 2-11-1 设置 1

*选择菜单“View”、“Graph”、“Time/Frequency...”进行如下设置：

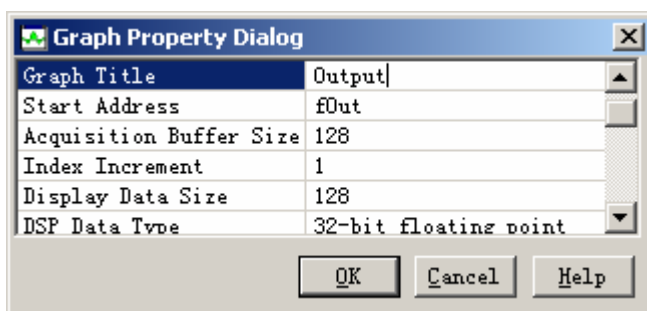


图 2-11-2 设置 1

5.设置断点：在标号“done”语句设置软件断点（Toggle breakpoint）。

6.运行并观察结果

选择“Debug”菜单的“Run”项，或按 F5 键运行程序。程序运行后停在断点上。

观察“Input”、“Output”窗口中时域图形 2-11-3；观察滤波效果。

鼠标右键单击“Input”和“Output”窗口，选择“Properties...”项，设置“Display Type”为“FFT Magnitude”，再单击“OK”按钮结束设置。看图 2-11-3

观察“Input”、“Output”窗口中频域图形；理解滤波效果。

五.实验结果

输入波形为一个低频率的正弦波与一个高频的余弦波叠加而成。如图：

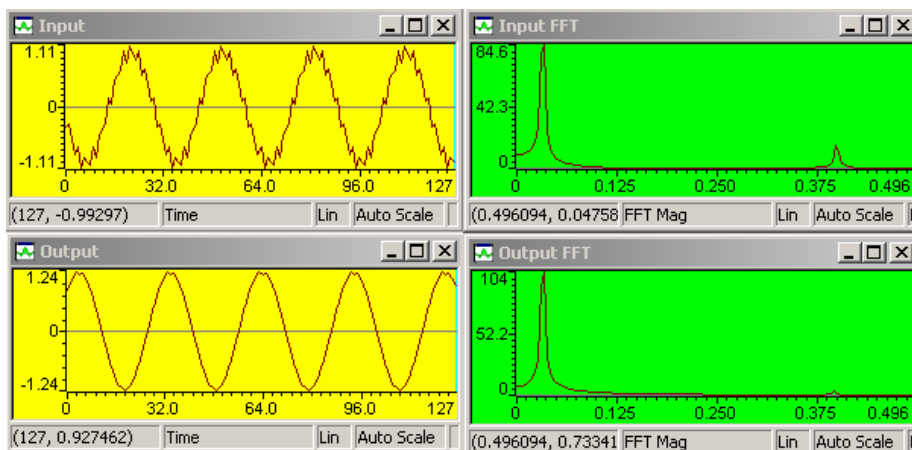


图 2-11-3 滤波前后图形

通过观察频域和时域图，得知：输入波形中的低频波形通过了滤波器，而高频部分则被衰减。

六.问题与思考

试设计合适的高通滤波参数滤掉实验的输入波形中的低频信号。