哈尔滨工业大学

数字信号处理器原理与应用
课程实验指导书

哈尔滨工业大学自动化测试与控制系

2005 年 6 月

目　录

第一部分　ICETEK–LF2407-A 评估板硬件使用指导

第一部分　ICETEK–LF2407-A评估板硬件使用指导.....1
第一节ICETEK-LF2407-A板级产品介绍.....1
1.1 ICETEK-LF2407-A概述.....1
1.2 ICETEK-LF2407-A 板功能.....1
1.3 结构框图.....1
第二节 板上器件功能与使用方法.....1
2.1 ICETEK-LF2407-A板构造.....1
2.2 ICETEK-LF2407-A 板的使用.....2
2.2.1 电源管理.....2
2.2.2 ICETEK-LF2407-A板的存储器空间.....2
2.2.3 I/O空间.....3
2.2.4 用户开关和指示灯.....4
2.2.5 选择振荡器.....4
2.2.6 数模转换.....4
2.2.7 JTAG接口.....5
2.2.8 SPI口.....5
2.2.9 异步串口.....5
2.2.10 CAN总线.....5
2.2.11　ICETEK-LF2407-A 跳线.....5
2.2.12 指示灯状态.....7
2.2.13 用户可控指示灯.....7
2.2.14 复位.....8
2.2.15 用户使用开关.....8
2.2.16 ON/OFF开关.....8
2.2.17 测试端.....8
第二部分 ICETEK－LF2407-A教学实验系统使用指导.....9
第一节　ICETEK DSP教学实验箱简介.....9
1.1 ICETEK DSP教学实验箱的特点.....9
1.2 ICETEK DSP教学实验箱的功能.....9
1.3 ICETEK DSP教学实验箱的组成.....10
1.4 ICETEK DSP教学实验箱性能指标.....11
1.5 ICETEK DSP教学实验箱结构图.....12
第二节 教学实验箱硬件接口和编程说明.....12
2.1 ICETEK DSP教学实验箱的外围接口.....12
2.2 ICETEK DSP教学实验箱硬件编程.....14
2.2.1 液晶显示模块编程控制.....14
2.2.2 发光二极管编程控制.....16
2.2.3 发光二极管显示阵列编程控制.....16
2.2.4 步进电机编程控制.....16

2.2.5 蜂鸣器编程控制.....17
2.2.6 键盘输入编程控制.....17
2.2.7 直流电机编程控制.....17
第三节 ICETEK DSP教学实验箱操作手册.....17
3.1 ICETEK DSP教学实验箱的使用.....17
3.1.2 连接各模块电源.....18
3.1.3 连接DSP评估板信号线.....18
3.2 ICETEK DSP教学实验箱使用注意事项.....18
3.3 ICETEK DSP教学实验箱故障判断及排除.....18
3.3.1 无法接通电源.....18
3.3.2 信号源没有输出.....18
3.3.3 显示/控制模块上步进电机不转.....19
3.3.4 显示/控制模块上液晶没有显示.....19
3.3.5 直流电机不停转动.....19
3.3.6 无法进入CCS软件仿真.....19
4.1 教学实验箱：ICETEK-EDU.....19
4.2 通用DSP开发系统：ICETEK5100-PP或ICETEK5100-USB.....19
4.3 DSP控制板：ICETEK-LF2407-A.....19
4.4 通用控制板：ICETEK-CTR.....20
第三部分　ICETEK－LF2407-A评估板软件实验指导.....1
实验一 数据存取实验.....1
实验二 I/O控制模块实验.....6
实验三 定时器实验.....9
实验四 模数转换实验.....15
实验五 数模转换实验.....23
实验六 PWM实验.....28
实验七 外设控制实验—发光二极管阵列显示实验.....33
实验八 外设控制实验—液晶显示器控制显示实验.....38
实验九 外设控制实验—键盘输入实验.....45
实验十 外设控制实验—步进电机控制实验.....52
实验十一 直流电机控制实验.....57
实验十二 异步串口通信实验.....65
实验十三 快速傅立叶变换（FFT）算法实验.....72

第一部分　ICETEK－LF2407-A 评估板硬件使用指导

第一节 ICETEK-LF2407-A 板级产品介绍

1.1 ICETEK-LF2407-A 概述

ICETEK-LF2407-A 板是一块独立的目标板，它非常适合检验 LF2407 DSP 的性能，此外，本目标板提供了 LF240x 系列芯片进行扩展和运行软件的标准平台。

ICETEK-LF2407-A 板使用了 TMS320LF2407 DSP 芯片，兼容所有 LF2407 的使用代码，它具有 2.5K 字节的片上数据存储器，128K 板上存储器，片上 UART，DAC7625 数模转换器。此 EVM 板还提供了 DSP 的扩展引脚，方便了用户外搭所需电路。

许多用户接口可利用简单的代码进行扩展，从而缩短了调试时间。

1.2 ICETEK-LF2407-A 板功能

它主要的接口包括目标存储器接口、模拟接口、CAN 总线接口、并口、用户指示灯和开关、外部扩展接口。

LF2407 提供了 128K 的静态存储器，外部 I/O 口支持相应的 64k I/O 端口，片上的 CAN 总线和 RS232 端口可用做扩展连接。

1.3 结构框图

ICETIK-LF2407-A 板的结构框图如图 1-1 所示。

第二节 板上器件功能与使用方法

本节描述了 EVM 板的构造和使用方法并提供了 EVM 板上接口的资料。

2.1 ICETEK-LF2407-A 板构造

ICETEK-LF2407-A 板由 6 个主要部分组成：

- LF2407 外部存储器
- A/D，D/A 转换
- 串口
- 指示灯和开关
- 片上 CAN 总线
- SPI 口
- DSP 扩展引脚
- JTAG 接口

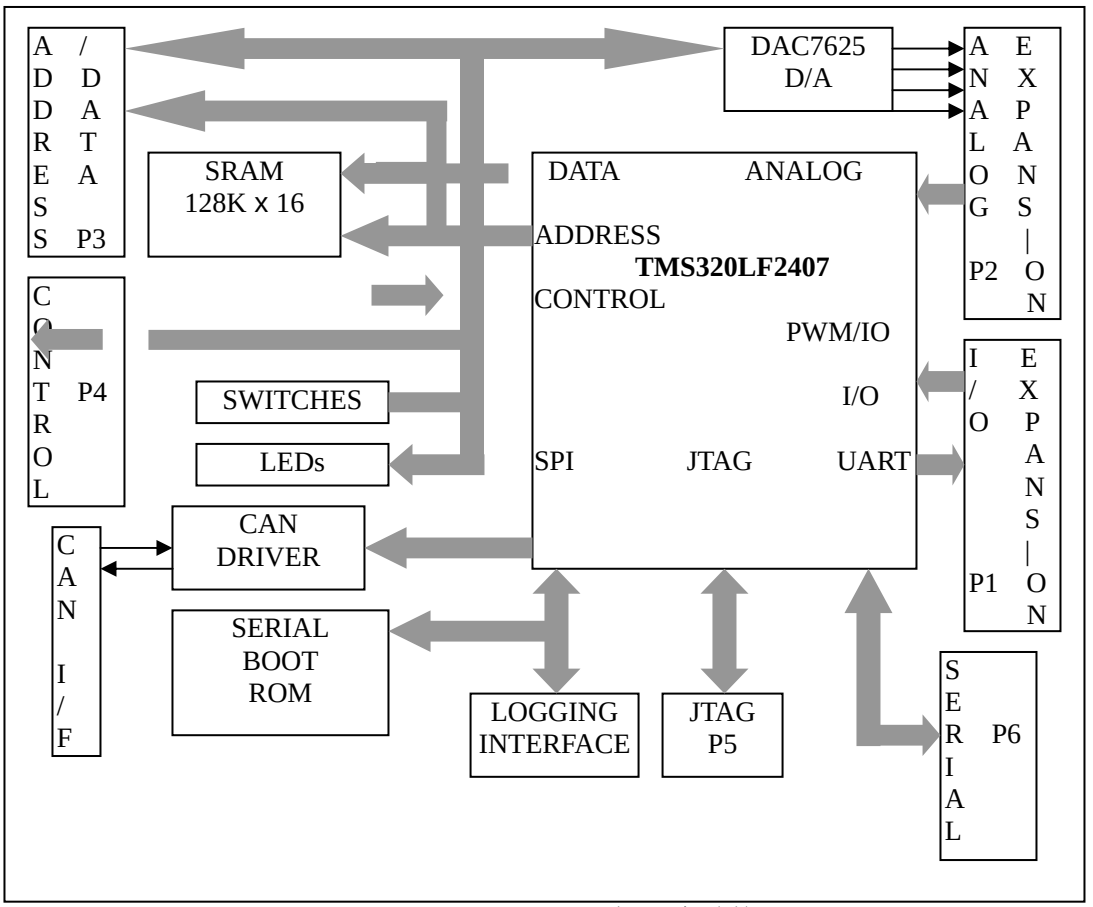


图1-1 TMS320LF2407 EVM板基本结构

2.2 ICETEK-LF2407-A 板的使用

2.2.1 电源管理

LF2407 使用的是 5V 电源输入，输入后在板上要对其进行降压，使其降为 3.3V 的内部电源，但板子需要 750 毫安的电流，要满足这个电流就必须从电路板上 2 毫米的插孔 J1 输入一个 5V 的电源。

2.2.2 ICETEK-LF2407-A 板的存储器空间

ICETEK-LF2407-A 板包括 64K 程序存储器，64K 数据存储器，合计为 128K 片上的静态存储器。

内部存储器在存储时比外部存储器具有更高的优先级，如果您需要更多关于 ICETEK-LF2407-A 板的存储器资料，请您查阅 TI TMS320LF2407 Users Guide。

- 程序存储器

程序存储器存在两种配置，我们可以通过 JP6 上的跳线选择使用哪一种结构，如果 JP6 选择为 2-3 ，那么 DSP 处于 MC 方式，并且其内部闪存从 0x0000 到 0x7fff 被激活，如果选择的为 1-2，那么内部 Flash 和 Rom 是不可用的，并且全部的程序存储空间都映射到外部存储器。

- 数据存储器

下图是数据存储器的结构，外部存储器从 0x8000-0xffff 有效：

0000 005F	存储器映射寄存器 保留区域
0060 007F	片上双存取访问 B2
0080 01FF	保留区域
0200 02FF	片上双存取访问 B0（CNF = 0） 保留区域 CNF = 1
0300 03FF	片上双存取访问 B0’（CNF = 0） 保留区域 CNF = 1
0400 07FF	保留区域
0800 0FFF	单存取访问 DON = 1 外部 DON = 0
1000 6FFF	无效区
7000 73FF	外部存储器映射寄存器 （系统、ADC、SCI、SPI、I/O、中断）
7400 743F	外部存储器映射寄存器 管理器 A
7440 74FF	保留区域
7500 753F	管理器 B
7540 77FF	保留区域
7800 7FFF	无效区
8000 FFFF	外部寄存器 JP7 = 1-2

图2-1 数据存储器结构

2.2.3 I/O 空间

下图显示的是 ICETEK-LF2407-A 板的 I/O 空间分配：

0000 0004	D/A 转换
0005 0007	保留区域
0008	4 种 DIP 开关
0009 000B	保留区域
000C	指示灯
000D 7FFF	保留区域
8000 FFFF	外部空间

图2-2 I/0空间分配

2.2.4 用户开关和指示灯

ICETEK-LF2407-A 板有 4 处开关和 4 个指示灯，方便使用，它们分别被设置在 I/O 映射区内的 0x0008 和 0x000C 地址上，使用 IN 或 OUT 指令，可以控制指示灯 D0 ~ D3。

2.2.5 选择振荡器

ICETEK-LF2407-A 板具有一个 15M 的振荡器，内核 CPU 接收到输入时钟 CLKIN/2（CPUCLK），在重新复位后，PLL 时钟模块预定义为 CPUCLK/4，大约为 2M 的输出。PLL 能够编程为 CPUCLK*4，这样的结果就可以得到 30M 的输出时钟。关于 PLL Clock Module 的详细资料，请查阅 TMS320LF2407 User’s Guide。

2.2.6 数模转换

ICETEK-LF2407-A 板，提供了 4 路 12-bit 的数模转换，转换后输出的将是从 0 ~ 3.3V 的直流电，转换器占用了 I/O 地址空间中 0x000 ~ 0x0004 的部分，0x000 ~ 0x0003 分别是数据寄存器的 4 条通路，地址 0x0004 用于寄存器控制，也就是说，您能在同一时间内分 4 路向寄存器写入数据并进行转换。

表 2-1 DAC I/O 地址

I/O 地址	通路
0x0000	1
0x0001	2
0x0002	3
0x0003	4
0x0004	转换

2.2.7 JTAG 接口

ICETEK-LF2407-A 板具有一个 14 根引脚的接口 P5，我们可以通过这个 JTAG 口对 TI 生产的 DSP 芯片进行仿真。

2.2.8 SPI 口

ICETEK-LF2407-A 板上具有与并口 SPI515 相兼容的 SPI 口，通过 SPI 口可以进行高速的数据传输。

如果需要使用此 SPI 口，那么跳线 JP4 就应该选择 2-3 脚。

2.2.9 异步串口

ICETEK-LF2407-A 板上具有一个符合 RS232 标准的 DB9 孔的异步串口 P6，我们可以通过这个串口进行数据的传输和计算机之间的通讯。如果用户要使用串口进行通讯，就要对跳线 JP10、JP11、JP12、JP14 进行设置。

2.2.10 CAN 总线

ICETEK-LF2407-A 板具有一个 4 孔的小型 CAN 接口 P7，这个串口可以进行高速的信号传输，如果用户使用此 CAN 接口，则要修改跳线 JP12 的设置。

CAN 总线的插头是一个 4 孔的连接器 P7。

2.2.11 ICETEK-LF2407-A 跳线

ICETEK-LF2407-A 板具有 16 根跳线。

下图显示了 ICETEK-LF2407-A 板上所有跳线所在位置（可以参考前面的电路版图： 图 2－1 上的标注）：

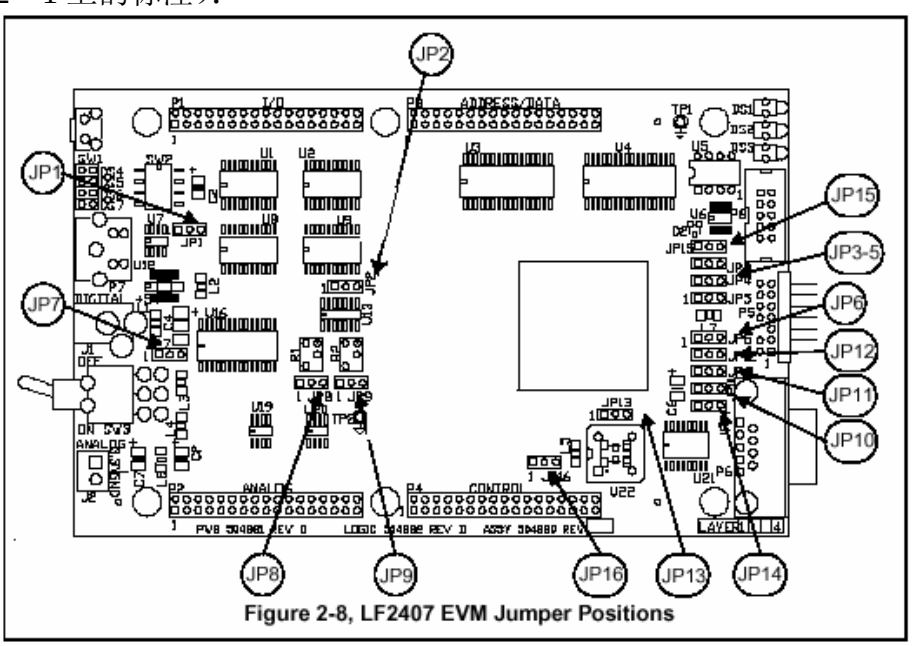


Figure 2-8, LF2407 EVM Jumper Positions

图2-3 LF2407 EVM板上跳线位置

- 跳线 1，能/否使用 CAN 总线

跳线 1 可以控制能/否使用 CAN 总线。选择 2-3 可以使用，选择 1-2 则不能使用。

- 跳线 2，选择 CAN 总线输入

跳线 2 可以选择 CANRX 输入信号的来源，如果选择了 1-2，CAN 总线直接可以接收 CAN 总线上的输入，如果选择了 2-3，则 CANRX/IOPC7 连接到 P4 的 24 脚接收传来的信号。

- 跳线 3，选择串行 EEPROM 的写保护

串行 EEPROM 可以被写保护，以防止 ROM 中的内容被破坏。

跳线 3 就可以设置单存取存储器写保护的状态，选择 1-2 时可以向 ROM 中写入内容，如果选择 2-3 就设置了写保护状态。

- 跳线 4，选择 SPI 口通讯

跳线 4 可以选择 SPI 口通讯方式，SPI 口可以选择外部扩展接口/串行 EEPROM 或 P8 的数据输入。如果选择 1-2 就可以通过 SPI 口进行数据的装载，如果选择 2-3，则就要通过外部扩展口/单存取存储器进行数据的装载。

- 跳线 5，能/否进行 FLASH 编程

跳线 5 连接了 LF2407 的 VCCP 管脚，在 LF2407 上 VCCP 管脚可以设置对内部闪存编程的设置，它还允许进行看门狗的设置，LF2407 User’s Guide 详细介绍了如何进行看门狗的设置。

- 跳线 6，MP/MC 能否进行内部闪存的设置

跳线 6 与 LF2407 上的 MP/MC 管脚相连，当选择 1-2 时，内部闪存是不可使用的，如果选择 2-3，则内部闪存可用。

- 跳线 7，模拟信号输入管理

跳线 7 控制了 ICETEK-LF2407-A 板的模拟信号输入，当选择 1-2 时系统滤掉数字信号保留模拟信号装载到 EVM 板，当选择 2-3 时，则模拟信号要经由 P2 装载到 EVM 板。

- 跳线 8，选择高电位

跳线 8 使用的是 LF2407 上的 VREFHI 管脚，当选择 1-2 时电压为 3.3V，如果选择 2-3，则控制电压通过 R1 把管理电压降为 0-3.3V。

- 跳线 9，选择低电位

跳线 9 使用的是 LF2407 上的 VREFLO 管脚，当选择 1-2 时电压为“0”接地，如果选择 2-3，则控制电压通过 R2 转化为 0-3.3V。

- 跳线 10，能否通过 DTR-复位

跳线 10 允许在 P7 端口进行复位，当选择 2-3 时就是把 P7 的第 4 脚拉低进行复位，如果选择 1-2 则不能复位。

- 跳线 11，能否向 BIO/IOPC1 发送请求

跳线 11 连接到 P6 口的 DSP’S BIO-/IOPC1 管脚。当选择 1-2 时不能发送请求，选择 2-3 时可以发送请求，这是使用串口进行通讯时硬件所必须的握手协议。

警　　告

如果选择 2-3 脚，那么一定不要从 P4 上设置 BIO-/IOPC1 脚。

- 跳线 12，能否从 SCIRXD/IOPA1 接收到数据，

跳线 12 能对 P6 上的 DSP'S SCIRXD/IOPA1 管脚进行设置，当选择 1-2 时可以接收到数据，当选择 2-3 时不能进行数据的接收并且 SCIRXD/IOPA1 管脚可进行外部扩展连接。

警　　告

如果选择 2-3 脚，那么一定不要从 P4 上设置 SCIORXD/IOOA1 脚。

- 跳线 13，选择震荡源

跳线 13 使用了 ICETEK-LF2407-A 上的输入时钟，当选择 1-2 时就选择了板上震荡器，当选择 2-3 时则从 P4 的 31 脚进行时钟的设置。

- 跳线 14，选择 DTS/RTS

跳线 14 选择 DSP 所接收的是否是时实信号。当选择 1-2 时，DSP 所接收的是固定信号，当选择 2-3 时 DSP 所能接收的是时实信号。

- 跳线 15，选择由 SPI/SCI 装载

跳线 15 允许用户选择片上程序装载的方式。用户可选择 SPI 或 SCI 中的任何一种方法进行程序的装载，当选择 1-2 时就是选择了 SPI 口进行装载，当选择了 2-3 时就是选择了 SCI 口装载。

- 跳线 16，选择输入

ICETEK-LF2407-A 板可以通过外部扩展接口或 RS232 口把数据装载到 EEPROM，使用 BOOT-LOAD 之前一定要进行跳线 6 的设置，如果选择从 SPI 口进行 BOOT-LOAD 那么还要对跳线 4 进行设置，并且跳线 16 要选择 2-3，跳线 15 要选择 SPI 口装载方式。如果选用 RS232 进行 BOOT-LOAD 就不需要进行设置跳线 4，跳线 6 还继续以前的设置，跳线 15 选择为 SCI 口装载方式，跳线 16 选择 1-2，这样就可以进行 BOOT-LOAD。

2.2.12 指示灯状态

ICETEK-LF2407-A 板上具有 3 个指示灯，两个受软件控制，DS3 是板子上电时的指示灯，下图显示了他们的关系：

LED #	Color	Controlling Signal	On Signal State
DS1	Red	W/R-/IOPC0 on DSP	1
DS2	Yellow	BIO-/IOPC1 on DSP	1
DS3	Green	Power On	N/A

2.2.13 用户可控指示灯

ICETEK-LF2407-A 板具有 4 个可以由用户进行编程设计的指示灯，在 I/O 空间中 0x000c 地址内写入二进制数就可以对这些指示灯进行控制，下图显示了他们的关系：

LED #	Color	Controlling Value	On Signal State
DS4	Red	0x01	1
DS5	Red	0x02	1
DS6	Red	0x04	1
DS7	Red	0x08	1

2.2.14 复位

ICETEK-LF2407-A 板可以进行多重复位，第一次复位产生与对 U12 的调整，这个设置将等待 LF2407 的 RESET 管脚上的电压到内部电压到达指定范围为止。

系统也可以进行输入输出的复位，内部复位是利用看门狗在 RS-管脚拉底的情况下产生的，外部复位是只要选择 SW1，主机复位是要对 P4 的 4 和 13 管脚进行设置，使其产生复位信号。

SW1 是用户复位开关，按下这个开关，ICETEK-LF2407-A 即刻就会复位。

2.2.15 用户使用开关

ICETEK-LF2407-A 具有一个 4 点的双列直插开关 SW2，每个开关都可手动进行设置。这个开关可以通过修改 I/O 中的地址 0x0008 进行设置，设置为“ON”时所读的数就为“1”。下图显示了他们的关系：

Position	Value Read	Switch State
1	0x01	On
2	0x02	On
3	0x04	On
4	0x08	On

2.2.16 ON/OFF 开关

开关 SW3 可以控制模拟和数字信号的输入，当它设置为“ON”时，EVM 板可以使用。

2.2.17 测试端

ICETEK-LF2407-A 板提供两个测试端，它们是一个接地一个模拟接地的两个探针。下图显示了他们的关系：

Test Point #	Signal
TP1	GND
TP2	Analog Ground

第二部分 ICETEK－LF2407-A 教学实验系统使用指导

第一节 ICETEK DSP 教学实验箱简介

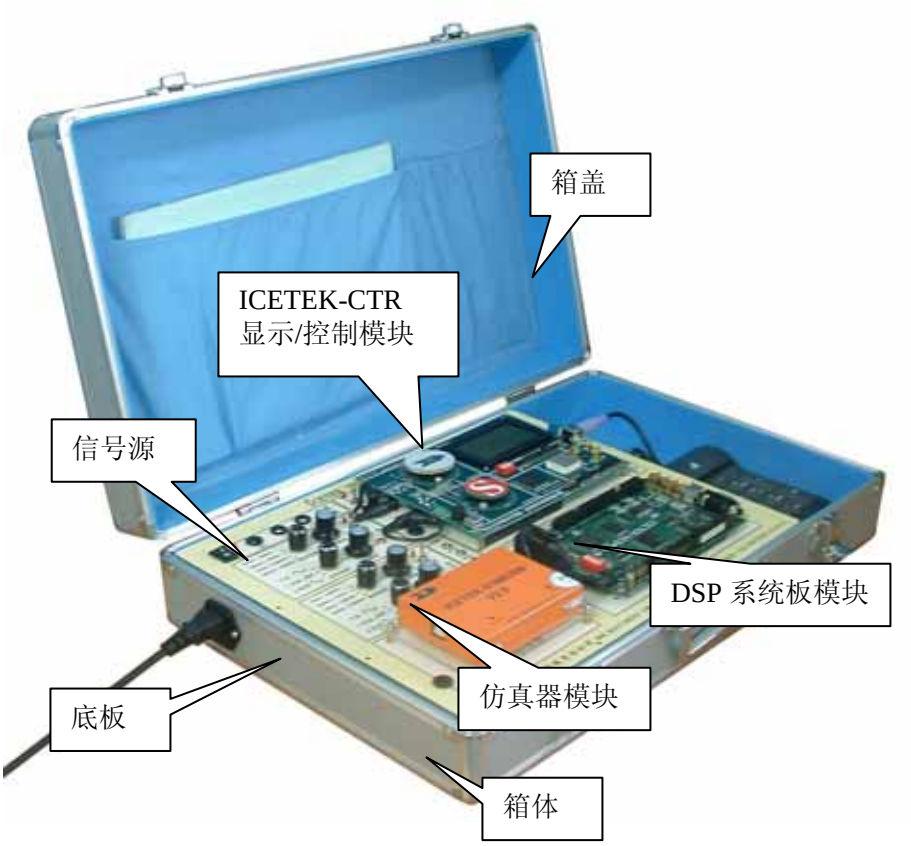
1.1 ICETEK DSP 教学实验箱的特点

- 完备性：提供完整的 DSP 实验环境。硬件上包括 DSP 仿真器、评估板、信号源、控制模块；软件上提供仿真软件、完全使用手册和实验例程。可以进行与 DSP 应用相关的大部分实验和测试。
- 易用性：完备的使用说明和实验手册使使用者可以轻松上手、尽快熟悉 DSP 使用的相关操作，多功能的控制模块提供从图象到声音、从输入到输出多种形象直观的显示、控制手段，使用户的知识得到感性的结果，从而加深对 DSP 的理解。
- 直观性：提供液晶图象显示、发光二极管阵列显示、电机指示等视觉实验效果，信号源也提供了容易控制、简单明了的测试手段，使实验现象能更加直观、具体、明确地展示出来。
- 灵活性：支持使用 ICETEK5100PP 和 ICETEK5100USB 仿真器；在接口相同的前提下，支持多种系列的评估板，如：ICETEK-LF2407-A 板、ICETEK-VC33-A 板、ICETEK-VC5416-A 板、ICETEK-VC33-AE 板、ICETEK-C6713-A 板和已经或即将推出的多种评估板。各模块更换操作简单、安装容易，可以适用各种教学需求。
- 适用性：针对 DSP 能同时进行多路信号处理的特点；提供两个独立的信号源，可单独设置，四路波形输出，充分测试 DSP 的并行数字信号处理能力。

1.2 ICETEK DSP 教学实验箱的功能

- 两个独立的信号发生器，可同时提供两种波形、四路输出；信号的波形、频率、幅度可调。
- 多种直流电源输出。支持对仿真器和评估板的直流电源连接插座。
- 显示输出：液晶图象显示器（LCD），可显示从 DSP 发送来的数据；发光二级管阵列(LEDArray)；发光二极管；马达指针 0-360 度指示。
- 音频输出：可由 DSP I/O 脚控制的蜂鸣器；D/A 输出提供音频插座，可直接接插耳机。
- 键盘输入：可由 DSP 回读扫描码；同时键盘产生中断信号作为 DSP 的外中断输入。
- 步进电机：四相步进电机，可由 DSP I/O 端口控制旋转和方向、速度。
- 直流电机：可以接收 DSP 输出的 PWM 控制信号，实现电机的转速和方向控制。
- 底板提供插座，可使用插座完成 DSP 评估板上的 A/D 信号输入和 D/A 输出。
- 测试模块：提供 14 个测试点，可以测量 PWM 输出、AD 输入和 DA 输出波形。
- 软件资料：相关 DSP 设计编程使用教材、实验教程、使用说明、实验程序等。

1.3 ICETEK DSP 教学实验箱的组成



如图所示，ICETEK DSP 教学实验箱主要由以下几个部分组成：

- 箱盖：保护实验箱设备；保存教材、使用手册、实验指导书、各种实验用的连线；可拆卸，在实验中可从箱体上拆下；带锁，可在关闭时用钥匙锁住。
- 箱体：装载实验箱设备；左侧外壁上有一个标准外接电源线插孔；通过固定螺丝与实验箱底板连为一体。
- 底板：固定各模块；提供电源开关、实验用直流电源插座(3)、A/D D/A 输入输出插座(8)、各模块直流供电插座(5)、信号插座(6)、信号源输出插座(4)、测试点(14)；实现显示控制模块和 DSP 评估板模块的信号互连。
- 信号源：两组、四路输出，可使用专门开关启动；提供切换选择输出方波、三角波和正弦波，另可选择输出频率范围(10Hz-100Hz，100Hz-1KHz，1KHz-10KHz，10KHz-100KHz)，还可进行频率和幅度(0-3.3V)的微调。
- 仿真器模块：固定 ICETEK 仿真器，支持 PP 型和 USB 型；提供 PP 型仿真器供电 5V 电源插座；仿真器可从底板上拆下更换。
- 显示控制模块：通过信号线连接到底板；从底板提供的 5V 和 12V 直流电源插座输入电源；提供液晶图形显示(128x64 象素)，发光二极管阵列显示(8x8 点)，指示灯(12 只，分为红、黄、绿三种颜色)，四相步进电机，直流电机，键盘(外接 PSII 接口)，蜂鸣器。显示控制模块可从底板上拆下更换。
- 测试模块：提供对常用信号的测试点，其中有 PWM 信号(4 路，仅针对 DSP

系统为 ICETEK-LF2407-A 的实验箱)、模数转换信号(4 路)、和数模转换信号(4 路)，另外还包括两个地线(DGND、AGND)。

- DSP 评估板模块：固定各种 DSP 评估板；提供 5V 直流电源插座(两个位置)；34Pin 信号线插座(4 个)，用于连接 DSP 评估板和实验箱底板。DSP 评估板模块可从底板上拆下更换。

1.4 ICETEK DSP 教学实验箱性能指标

- 直流电源：+5V(5A)，+12V(1A)，-12V(0.5A)，地
- 信号源(A、B)：
  - -双路输出
  - -频率范围：分为 4 段(10Hz—100Hz，100Hz—1KHz，1KHz—10KHz，10KHz—100KHz)，可通过拨动开关进行选择
  - -频率微调：在每个频率段范围内进行频率调整
  - -波形切换：提供 3 种波形(方波，三角波，正弦波)，可通过拨动开关进行选择
  - -幅值微调：0—3.3V 平滑调整
- 信号接插孔：4 路 A/D 输入(ADCIN0-ADCIN3)，4 路 D/A 输出(DACOUT1—DACOUT4)，每路均提供信号和地
- 显示/控制模块(可选)：
  - -液晶显示(LCD)：128×64 点阵图形显示屏，可调整显示对比度
  - -发光二极管显示阵列：8×8 点阵
  - -发光二极管
  - -蜂鸣器
  - -步进电机:四相八拍,步距角 5.625,起动频率≥300PPS,运行频率≥900PPS
  - -直流电机：空载转速 3050 转/分，输出功率 1.35W，启动力矩 21.3N
  - -键盘：PSII 接口，标准键盘
  - -拨动开关(DIP)：4 路，可实现复位和设置 DSP 应用板参数
- 电源输入：220V 交流

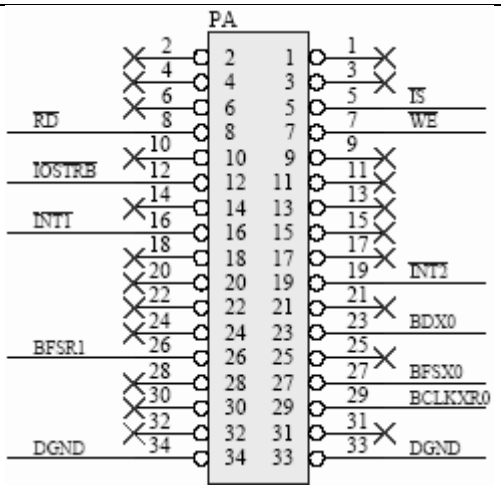
1.5 ICETEK DSP 教学实验箱结构图

第二节 教学实验箱硬件接口和编程说明

2.1 ICETEK DSP 教学实验箱的外围接口

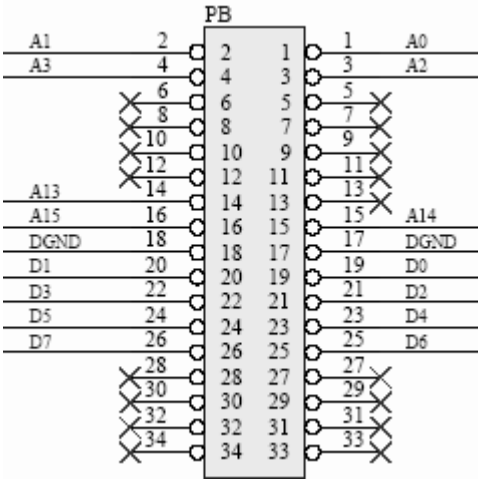
- 外围接口 PA：

  2407-A 扩展接口 P4(其中连接到实验箱底板的引脚加下划线，其他未连接)



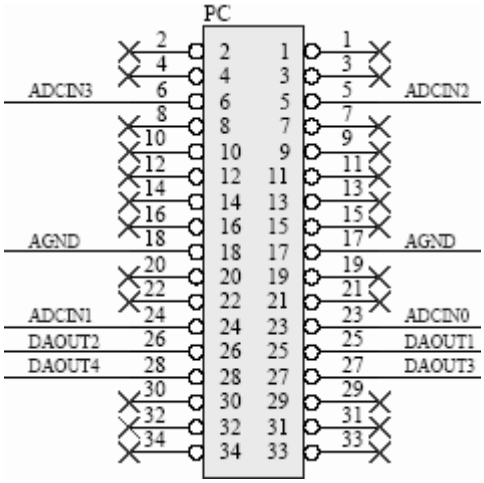
- 外围接口 PB：

  2407-A 扩展接口 P3(其中连接到实验箱底板的引脚加下划线，其他未连接)



- 外围接口 PC：

2407-A 扩展接口 P2(其中连接到实验箱底板的引脚加下划线，其他未连接)



2.2 ICETEK DSP 教学实验箱硬件编程

显示控制模块有一个全局控制寄存器，地址映射在 2407 的 I/O 扩展空间上，地址为 8000H。其各位上的定义如下：

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
GS	LEDA2	LEDA1	LEDA0	BUZZE	PWME	IOPE	DCME

GS：全局控制标志位

LEDA2-0：发光二极管阵列列显示控制位

BUZZE：蜂鸣器使能

PWME：PWM 控制使能

IOPE：通用 I/O 端口(FSX0)直接控制交通灯北方向红灯使能

DCME：直流电机使能

例如需要使能直流电机，可以用以下 C 语言语句：

```
port8000=1;
```

2.2.1 液晶显示模块编程控制

液晶显示模块的访问、控制是由 2407DSP 对扩展 I/O 接口的操作完成。

控制 I/O 口的寻址：命令控制 I/O 接口的地址为 0x8001，数据控制 I/O 接口的地址为 0x8003 和 0x8004，辅助控制 I/O 接口的地址为 0x8002。

显示控制方法：

-液晶显示模块中有两片显示缓冲存储器，分别对应屏幕显示的象素，向其中写入数值将改变显示，写入“1”则显示一点，写入“0”则不显示。其地址与象素的对应方式如下：

左侧显示内存						右侧显示内存					
Y=	0	1	···	62	63	0	1	···	62	63	行号
	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	0
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
X=0	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	7
	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	8
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	55
X=7	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	DB0	56
	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	DB7	63

-发送控制命令：向液晶显示模块发送控制命令的方法是通过向命令控制 I/O 接口写入命令控制字，然后再向辅助控制接口写入 0。由于液晶模块相对于 DSP 来讲是慢速设备，在命令之间可能需要增加延时语句。下面给出的是基本命令字、解释和 C 语言控制语句举例。

- 显示开关：0x3f 打开显示；0x3e 关闭显示；

```
port8001=0x3f; port8002=0; //将液晶显示打开
port8001=0x3e; port8002=0; //将液晶显示关闭
```

- 设置显示起始行：0x0c0+起始行取值，其中起始行取值为 0 至 63；

```
port8001=0x0c0; port8002=0; // 设置从存储器第 0 行开始显示
port8001=0x0c8; port8002=0; // 设置从存储器第 8 行开始显示
```

- 设置操作页：0x0b8+页号，其中页号取值为 0-7；

```
port8001=0x0b0; port8002=0; //设置即将操作的存储器第 0 页
port8001=0x0b2; port8002=0; //设置即将操作的存储器第 2 页
```

- 设置操作列：0x40+列号，其中列号为取值为 0-63；

```
port8001=0x40; port8002=0; //设置即将操作的存储器第 0 列
port8001=0x44; port8002=0; //设置即将操作的存储器第 4 列
```

-写显示数据：在使用命令控制字选择操作位置(页数、列数)之后，可以将待显示的数据写入液晶显示模块的缓存。将数据发送到相应数据控制 I/O 接口即可。由于液晶模块相对于 DSP 来讲是慢速设备，在命令之间可能需要增加延时语句。C 语言语句举例说明：

```
port8003=0x80; port8002=0;//向左侧屏幕缓存存入数 0x80，
                          //如果显示行、页号和列号均为 0
                          //时，屏幕上第 8 行第 1 列将显
                          //示黑色象素
port8004=0x01; port8002=0;//向右侧屏幕缓存存入数据 1，如
                          //果显示行、页号和列号均为 0
                          //时，屏幕上/第 1 行第 65 列将显
                          //示黑色象素
```

2.2.2 发光二极管编程控制

显示/控制模块上的发光二极管是由连接在 2407DSP 扩展 I/O 接口上的寄存器 EWR 和 SNR 控制的。这两个寄存器均为 6 位寄存器，其位定义见下表：

EWR：

bit5	bit4	bit3	bit2	bit1	bit0
东-红	东-黄	东-绿	西-红	西-黄	西-绿

SNR：

bit5	bit4	bit3	bit2	bit1	bit0
南-红	南-黄	南-绿	北-红	北-黄	北-绿

两个寄存器的地址均映射到 2407DSP 的 I/O 空间，地址为 8007H，DSP 通过对 I/O 区域该地址的写操作来修改两个寄存器上各位的状态，当寄存器某位取‘1’值时，相应指示灯被点亮，取‘0’值则熄灭。当写入 8007H 的数据(8 位有效值)的高两位为‘00’时，数据的低 6 位将写入 EWR 寄存器；当高两位的值为‘01’时，写入 SNR 寄存器。

例如：需要点亮东、西方向的红灯和南、北方向的绿灯，其它灯均熄灭时，可以用下面 C 语句完成。对于高速 DSP，可能需要在两个语句之间加入延时语句。

port8007=0x024; port8007=0x49;

2.2.3 发光二极管显示阵列编程控制

发光二极管显示阵列的显示是由 I/O 扩展端口控制，DSP 须将显示的图形按列的顺 序存储起来(8×8 点阵，8 个字节，高位在下方，低位在上方)，然后定时刷新控制显示。具体方法是，将以下控制字按先后顺序，每两个为一组发送到全局控制寄存器的第 6-4 位和端口 0x8005，发送完毕后，隔不太长的时间(以人眼观察不闪烁的时间间隔)再发送一遍。由于位值为“0”时点亮，所以需要将显示的数据取反。

000B, 第 8 列数据取反；001B, 第 7 列数据取反；

010B, 第 6 列数据取反；011B, 第 5 列数据取反；

100B, 第 4 列数据取反；101B, 第 3 列数据取反；

110B, 第 2 列数据取反；111B, 第 1 列数据取反。

注意：在使用前须在 I/O 端口 8007 写入控制字 0x0C1，以打开此设备。关闭时写 0x0C0。

2.2.4 步进电机编程控制

步进电机是由寄存器 PWMR 控制。这个寄存器映射在 2407DSP 的 I/O 空间 8007H 上，当 DSP 向该地址写数据(8 位有效值)时，高两位为‘10’时数据的低 4 位将写入 PWMR 寄存器(PWM4、PWM3、PWM2、PWM1)。

WM1-4 按照下面拍的顺序给电机的四相输入端送入控制信号，且频率大于 500PPS，电机将开始正向转动。如果按照拍的逆序送控制信号，且频率大于 500PPS，电机将开始反向转动。

拍	PWM4	PWM3	PWM2	PWM1
1	1	1	1	0
2	1	1	0	0
3	1	1	0	1
4	1	0	0	1
5	1	0	1	1
6	0	0	1	1
7	0	1	1	1
8	0	1	1	0

控制的方法是：首先设置全局控制寄存器中的 PWME 位为‘1’，再使 DSP 以一定的频率改变 PWM4-1 各位状态，输出正向或反向的 PWM 波。

2.2.5 蜂鸣器编程控制

蜂鸣器由 DSP 通用 I/O 管脚输出控制，可将此管脚上的频率输出转换成声音输出。2407 的通用 I/O 口 BDX0 控制蜂鸣器的输出频率。

控制的方法是首先设置全局控制寄存器中的 BUZZE 位为‘1’，再使用 DSP 通用定时器设置 BDX0 以一定的频率改变高低状态，输出方波。

2.2.6 键盘输入编程控制

键盘的扫描码由 DSP 的 I/O 扩展地址 0x8001 给出，当有键盘输入时，读此端口得到扫描码，当无键被按下时读此端口的结果为 0。读取的方法如下：

```
nScanCode=port8001; nnn=port8002;                    // nScanCode 中为扫描码
```

2.2.7 直流电机编程控制

直流电机需要加上适合的电压(1.5-3.0V)、通过一定的电流(0.12A)才能转动，当电流改变时会使电机转速改变，当电流的方向发生变化时，电机会朝相反的方向转动。

显示控制模块将 C2407DSP 的通用 I/O 管脚 BFSR1 的输出接到直流电机控制电路，利用此管脚上输出的 PWM 波形的占空比变化来产生受控改变的电流，再利用另一个通用 I/O 管脚 BCLKXR0 上的电平信号控制电流的方向。

具体控制的方法是：首先设置全局控制寄存器中的 DCME 位为‘1’，再通过改变 BFSR1 的状态在此管脚上输出可以改变占空比的 PWM 波，同时设置 BCLKXR0 为高电平则顺时针转动，低电平为逆时针转动。

第三节 ICETEK DSP 教学实验箱操作手册

3.1 ICETEK DSP 教学实验箱的使用

3.1.1 连接电源

打开实验箱，取出三相电源连接线；将电源线的一端插入实验箱外部左侧箱壁上的电源插孔中；确认实验箱面板上电源总开关(位于实验箱底板左上角)处于“关”的位置；连接电源线的另一端至 220V 交流供电插座上，保证稳固连接。

3.1.2 连接各模块电源

确认断开实验箱总电源；使用电源连接线(两端均为带孔的插头)连接显示/控制模块上边插座到实验箱底板上+12V 电源插座;使用电源连接线(两端均为带孔的插头)连接显示/控制模块下边插座到实验箱底板上+5V 电源插座；如使用 PP(并口)型仿真器，则使用电源连接线(两端均为带孔的插头)连接仿真器上插座到实验箱底板上+5V 电源插座；使用电源连接线(两端均为带孔的插头)连接 DSP 评估板模块电源插座到实验箱底板上+5V 电源插座。注意各插头要插到底，防止虚接或接触不良。

3.1.3 连接 DSP 评估板信号线

当需要连接信号源输出到 A/D 输入插座时，使用信号连接线(两端均为单声道耳机插头)分别连接相应插座。

3.1.4 接通电源

检查实验箱上 220V 电源插座(箱体左侧)中保险管是否完好,在连接电源线以后，检查各模块供电连线是否正确连接，打开实验箱上的电源总开关(位于实验箱底板左上角)，使开关位于“开”的位置，实验箱底板上的电源指示灯亮。

3.2 ICETEK DSP 教学实验箱使用注意事项

- 拆卸各模块时请务必将实验箱总电源关闭；
- 不使用显示/控制模块时将相关电源开关关闭；
- 3．220V 交流电源线连接须牢靠，勿使发生虚接或接触不良，并保证良好地连接地线；
- 实验箱底板上标称值不同的直流电源不能直接跨接；
- 实验箱底板上直流电源不能直接跨接地线；
- 不要直接连接电源和信号插座；
- 显示/控制模块上的两个电源插座不要连接错误，上边插座为+12V，下面的为+5V；
- 连接不同类型的插座时，请再三确认无误后进行；
- 不要带电拔插各模块；
- 不要带电拔插仿真器和 DSP 评估板上 JTAG 插头的连接电缆；
- 如无特殊情况，请勿打开实验箱底板；
- 不要带电拔插键盘插头；
- 如遇实验箱冒烟等异常现象请立即关闭总电源，并查找原因。

3.3 ICETEK DSP 教学实验箱故障判断及排除

3.3.1 无法接通电源

请检查外接电缆是否完好；电缆是否与实验箱边插座连接妥当；电缆是否与外接插座连接紧密；检查实验箱上 220V 电源插座(箱体左侧)中保险管是否完好。

3.3.2 信号源没有输出

请确认相应信号源的开关是否打开；检查相应信号源的“幅值微调”旋钮是否处在最小位置；信号连接线是否连接好。

3.3.3 显示/控制模块上步进电机不转

请检查显示/控制模块的+12V 电源是否连接，开关是否打开；摸一下电机指针看是否有振动，如有试着拨动一下是否能恢复转动。

3.3.4 显示/控制模块上液晶没有显示

请调节显示/控制模块上液晶对比度调节电位器 R2。

3.3.5 直流电机不停转动

如果有的话退出 CCS 启动运行的程序；将显示/控制模块的电源开关关闭后再打开。

3.3.6 无法进入 CCS 软件仿真

- 断掉实验箱电源，拔掉 usb 电缆从仿真器上，重新插 usb 电缆，检查 usb 上的红灯和绿灯是按照先红后绿的次序来亮的。然后再打开实验箱电源。
- 请按照后面的实验一来配置软件和驱动。
- 请用 ctrl, alt delete 三个键盘键组合进入 win2000 操作系统的任务管理器，看看“进程”中是否有“CCS_app.exe”的进程，有的话，请先关闭，然后再点击进入 CCS 软件。

第四节　教学实验系统技术指标

（型号：ICETEK-LF2407-USB-EDU / ICETEK-LF2407-PP-EDU）

4.1 教学实验箱：ICETEK-EDU

- 实验箱自带双信号发生器，产生 10HZ-100K(±10%)信号，包括方波，正弦波和三角波。
- 自带电源转换设计，直接输入 220V 电源，分别产生多路+5V 和+12V 电源。
- 设计 A/D 和 D/A 的直接输入输出接口，实验连接方便，并可以连接示波器。

特点：双信号发生器设计，更加贴近 DSP 的实际应用。因为许多实际的情况都是需要对两个信号进行相关分析。

4.2 通用 DSP 开发系统：ICETEK5100-PP 或 ICETEK5100-USB

- 开发系统支持 C2000/VC33/C5000/C6000 DSP 的开发；
- 并口或 USB 口连接，适合于目前的各种计算机接口。

特点：通用开发系统和 DSP 控制板分离，有利于将来 DSP 的升级。同时，也可以脱离实验箱单独从事科研开发使用。

4.3 DSP 控制板：ICETEK-LF2407-A

- 主处理芯片：TMS320LF2407A，运行速度为 40MIPS，128K 存储空间；
- 16 路 10bit 片上 A/D 接口；
- 片上双重的管理器 PWM；
- 4 路的 DAC7625 转换；
- UART 串行接口，符合 RS232 标准；
- 32K 片上 FLASH；
- CAN 总线标准接口；

- 用户开关和测试指示灯；
- 数据、地址、I/O、控制，4 处扩展连接器；
- 具有 IEEE1149.1 相兼容的逻辑扫描电路，该电路仅用于测试和仿真；
- +5V 电源输入，内部 3.3V 电源管理；
- 1 带片内加密位。

4.4 通用控制板：ICETEK-CTR

- 液晶：128*64 图形显示
- 键盘：17 键财务数字键盘（标准 PS2 接口）
- LED 阵列：8*8
- 发光二极管控制：12 个（交通灯）
- 蜂鸣器：无源，1 个
- 步进电机：12V，四相八拍，>500PPS，1 个
- 直流电机：1.5V，0.12A，3050 转/分，1 个

特点：这是一个通用的控制模块，可以和多种 DSP 控制板连接，如 C2000 系列，C5000 系列，VC33 系列，C6000 系列等。

第三部分　ICETEK－LF2407-A 评估板软件实验指导

实验一 数据存取实验

一. 实验目的

了解 TMS320LF2407A 的内部存储器空间的分配及指令寻址方式。
了解 ICETEK-LF2407-A 板扩展存储器空间寻址方法，及其应用。
了解 ICETEK-2407-EDU 实验箱扩展存储器空间寻址方法，及其应用。
学习用 Code Composer Studio 修改、填充 DSP 内存单元的方法。
学习操作 TMS320C2xx 内存空间的指令。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱（或 ICETEK 仿真器+ICETEK-LF2407-A 系统板+相关连线及电源）。

三. 实验原理

TMS320LF240x DSP 内部存储器资源介绍

TMS320LF240x 系列 DSP 基于增强的哈佛结构，可以通过三组并行总线访问多个存储空间。它们分别是：程序地址总线（PAB）、数据读地址总线（DRAB）和数据写地址总线（DWAB）。由于总线工作是独立的，所以可以同时访问程序和数据空间。

TMS320LF240x 系列 DSP 的地址映象被组织为 3 个可独立选择的空间：程序存储器（64K 字），数据存储器（64K 字），输入/输出（I/O）空间（64K 字）。

*程序区：

00- 7FFFh： 如 DSP 工作在 MC 方式，片内 32k FlashMP 方式，片外扩 32k 展存储器
8000-87FFh： PON 位=1，片内 2k SARAMPON 位=0，片外 2k 扩展存储器
8800-FDFFh： 片外 2k 扩展存储器
FE00-FEFFh： CNF 位=1，保留空间
CNF 位=0，片外扩展存储器
FF00-FFFFh： CNF 位=1，为片内 DARAM（B0）
CNF 位=0，为片外扩展存储器

*数据区：

0000-005Fh： 寄存器映射地址，保留区
0060-007Fh： 片内 DARAM（B2）
0080-00FFh： 非法访问区
0100-01FFh： 保留区

0200-02FFh： CNF 位=0，片内 DARAM（B0）
CNF 位=1，保留区
0300-03FFh： 片内 DARAM（B1）
0400-04FFh： 保留区
0500-07FFh： 非法访问区
0800-0FFFh： DON 位=1，片内 2k SARAM
DON 位=0，保留区
1000-6FFFh： 非法访问区
7000-7FFFh： 外设寄存器映射地址区
8000-FFFFh： 片外扩展存储器

*I/O 区：
0000-FEFFh： 片外扩展区
FF00-FF0Eh： 保留区
FF0F-FF0Fh： Flash 控制模式寄存器
FF10-FFFEh： 保留区
FFFF-FFFFh： 等待状态寄存器

ICETEK-LF2407-A 板对 TMS320LF2407A DSP 存储空间的扩展
程序区：扩展所有片外扩展存储器
数据区：扩展所有片外扩展存储器
I/O 区：
0000-0004h： D/A 转换控制寄存器
0005-0007h： 保留
0008-0008h： 板上 DIP 开关控制寄存器
0009-000Bh： 保留
000C-000Ch： 板上指示灯控制寄存器
000D-7FFFh： 保留

ICETEK-LF2407-EDU 实验箱对 TMS320LF2407A DSP 存储空间的扩展
I/O 区：
8001-8001h： 读-键盘扫描值，写-液晶控制寄存器
8002-8002h： 液晶控制寄存器
8003-8004h： 液晶显示数据寄存器
8005-8005h： 发光二极管显示阵列控制寄存器
8006-9FFFh： 保留

四. 实验程序

```
main()
{
```

```
    int * pInt;
    int * pDes;
    int i;
    pInt= (int * ) 0x201;
    pDes= (int * ) 0x301;
    for(i = 0;i<4;i++)
    {
        * (pInt+i) = i+1;
    }
    for(i=0;i<4;i++)
    {
        * (pDes+i) = * (pInt+i);
    }
    while(1)
    {
    }

}
```

五. 实验步骤

1. 实验准备

⑴. 连接设备：

① 关闭计算机和实验箱电源；

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；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；

③ 关闭实验箱上的三个开关。

⑵. 开启设备：

① 打开计算机电源；

② 打开实验箱电源开关，打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；

③ 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。

⑶. 设置 Code Composer Studio 为 Emulator 方式。

⑷. 启动 Code Composer Studio：双击桌面上“  ”图标，启动 Code Composer Studio 2.2。

2. 打开工程文件

打开菜单“Project”的“Open”项；选择 D:\2407EDULab\Lab1-Memory 目录中的“Memory.pjt”。

右键单击工程管理窗口中的“GEL Files”，单击“Load GEL…”，选择工程目录中的 init.gel 文件。

3. 观察修改程序区

（1）显示程序：选择菜单“View”的“Memory…”项；在“Title”中输入 PROG，在“Address”项中输入 0x40，选择“Page”项为“Program”；单击“OK”按钮；“PROG” 窗口中显示了从地址 40H 开始的程序内存，由于 ICETEK-LF2407-A 板设置 DSP 工作在 MP 方式，窗口中显示的是片外扩展程序存储单元的内容；根据 cmd 文件中的设置，下载后的机器代码的入口应从 44H 处存放。

（2）改程序区存储单元

程序区单元的内容由 CCS 的下载功能填充，但也能用手动方式修改；双击“PROG”窗口地址“0x0040:”后的第一个数，显示“Edit Memory”窗口，在“Data”中输入 0x1234，单击“Done”按钮，观察“PROG”窗口中相应地址的数据被修改。

（3）观察修改数据区

① 显示片内数据存储区 B0：选择菜单“View”的“Memory…”项；在“Title”中输入 DARAM B0，在“Address”项中输入 0x200；单击“OK”按钮；“DARAM B0” 窗口中显示了从地址 200H 开始的数据内存；这片地址属于片内 DARAM B0。

② 显示片内数据存储区 B1：按照步骤⑴打开“DARAM B1”窗口显示从地址 0x300 开始的片内 DARAM B1 区的数据单元。

③ 修改数据单元：数据单元也可以单个进行修改，只需双击想要改变的数据单元即可；选择菜单“Edit”、“Memory”、“Fill…”，在“Address”项中输入 0x200，在“Length”中输入 16，在“Fill”中输入 0x11，单击“OK”按钮，可在 200H 开始的数据区中的头 16 个单元填充统一的数 0x11；观察“DARAM B0”窗口的变化；同样请将 0x300 开始的头 16 个单元的值用 0 填充。

④ 访问未扩展的区域：当访问未扩展的存储单元时，将不能正确修改内容；选择菜单“View”的“Memory…”项；在“Title”中输入 NO EXIST，在“Address”项中输入 0xA000，选择“Page”项为“I/O”；单击“OK”按钮； “NO EXIST” 窗口中显示了未扩展而不存在的 I/O 空间内存；试着修改其中的单元，可发现无法进行任何改动。

4. 运行程序观察结果

⑴ 编译和下载程序：单击菜单“Option”、“Program Load…”，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”、“Build All”，编译、连结和下载程序；观察“PROG”窗口中的变化。

⑵ 打开 CPU 寄存器观察窗口：选择菜单“View”、“CPU Registers”、“CPU Register”。

⑶ 单步执行程序并观察结果：按 F10 键单步运行，直到程序尾部的空循环语句；观察 CPU 寄存器窗口中 DP、ACCS、ST0、AR0、AR1、AR2 的变化；观察“DARAM B0”和“DARAM B1”中的显示；体会用程序修改数据区语句的使用方法。

六. 实验结果

实验程序运行之后，位于数据区地址 201H 开始的 4 个单元的数值被复制到了数据区 300H 开始的 4 个单元中。

* 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。

* 在 MC 方式时，程序区前 32k 均为 Flash 存储器区，只读，程序将无法下载。

* 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。

实验二 I/O 控制模块实验

一. 实验目的

1. 了解 TMS320LF2407A DSP 的数字 I/O 控制模块的使用方法；
2. 学习使用 I/O 管脚控制外围设备；
3. 学会用程序驱动简单外围设备。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱（或 ICETEK 仿真器+ICETEK-LF2407-A 系统板+相关连线及电源）。

三. 实验原理

1. TMS320LF2407A DSP 的数字 I/O 控制模块介绍

数字输入/输出模块是集成在 TMS320LF2407A 片内的外设之一，它主要对芯片的通用、双向的数字 I/O（GPIO）引脚进行控制。这些 I/O 引脚大多数是基本功能和一般 I/O 复用的引脚，数字 I/O 模块采用了一种灵活的方法，以控制专用 I/O 和复用 I/O 引脚的功能，所有 I/O 和复用引脚的功能可通过 9 个 16 位控制寄存器来设置，这些寄存器可分为两类：

• I/O 口复用控制寄存器（MCRx）：用于控制选择 I/O 口作为基本功能方式或一般 I/O 引脚功能；

• 数据和方向控制寄存器（PxDATDIR）：当 I/O 口用作一般 I/O 引脚功能时，用数据和方向控制寄存器可控制数据和到双向 I/O 引脚的数据方向，这些寄存器直接和双向 I/O 引脚相连。

具体控制寄存器的访问地址、定义请参见有关资料。

2. ICETEK-LF2407-A 板引出的 I/O 管脚及使用方法

ICETEK-LF2407-A 板使用了一些 I/O 管脚对 DSP 进行控制。例如：跳线 JP6 连接 DSP 上 MP/MC 管脚，在 DSP 复位时，DSP 可读回这一管脚的设置，当管脚接高电平时，DSP 采用微处理器（MP）方式工作，否则设置成微控制器（MC）方式。

ICETEK-LF2407-A 板在扩展插头上将未使用的 I/O 引脚接出，提供给用户连接使用。其定义见 ICETEK-LF2407-A 板说明。这些管脚支持 0-3.3V 逻辑电平操作，用户在进行相应设置后可以在 I/O 管脚上进行输入或输出操作，使用时须注意根据引脚本身的负载能力驱动相关设备。

3. ICETEK-LF2407-EDU 实验箱及控制模块使用的 I/O 管脚

ICETEK-LF2407-EDU 实验箱将引脚 ADCIN00-ADCIN03 连接到了实验箱底板上“A/D 输入”的四个插座上。

ICETEK-LF2407-EDU 实验箱控制模块使用如下引脚：

PWM12/IOPE6--指示灯

PWM11/IOPE5 和 TDIRB/IOPF4—步进电机

CANTX/IOPC6—蜂鸣器

四. 实验程序

实验程序通过相关寄存器设置，使用 PWM12/IOPE6 作为输出，控制实验箱控制模块上指示灯 J5 进行有规律地闪烁。方法是用程序定时地修改 PWM12/IOPE6 引脚的状态。

注：PWM12/IOPE6 引脚未连接驱动设备，通过一个 650Ω电阻限流后，直接连接到指示灯 J5。

源程序及注释：

```
main()
{
    unsigned int uWork;

    *WDCR=0x6f;
    *WDKEY=0x5555;
    *WDKEY=0xaaaa;        /* 关闭看门狗 */

    *SCSR1=0x81fe;        /* DSP 运行频率 40m */
    (*MCRB)=0;

    uWork=(*MCRC);       /*  将 PWM12/IOPE6 设置成通用 I/O 口 */
    uWork&=0x0ffbf;
    (*MCRC)=uWork;

    while ( 1 )
    {
        uWork=(*PEDATDIR);    /* PWM12/IOPE6 的控制寄存器 */
        uWork|=0x4000;        /* 输出 */
        uWork^=0x0040;        /* 输出状态 */
        (*PEDATDIR)=uWork;
        Delay(256);           /* 延时片刻 */
    }
}
```

五. 实验步骤

1.实验准备

⑴ 连接设备：

① 关闭计算机和实验箱电源；

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；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；

③关闭实验箱上的三个开关。

⑵ 开启设备：

① 打开计算机电源；

② 打开实验箱电源开关，打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；打开控制模块电源开关；

③ 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。

⑶ 设置 Code Composer Studio 为 Emulator 方式：

参见“Code Composer Studio 入门实验”之四.2。

⑷ 启动 Code Composer Studio

双击桌面上“  ”图标，启动 Code Composer Studio 2.2。

⑸ 打开工程文件

打开菜单“Project”的“Open”项；选择 D:\2407EDULab\Lab6-IOPin 目录中的“IOPin.pjt”。

2. 浏览程序

在项目浏览器中，双击 led.c，激活 led.c 文件，浏览该文件的内容，理解各语句作用。打开 led.cmd，浏览并理解各语句作用。

3. 编译工程

单击“Project”菜单，“Rebuild all”项，编译工程中的文件，生成 IOPin.out 文件。

4. 下载程序

单击“File”菜单，“Load program…”项，选择 D:\2407EDULab\Lab6-IOPin 目录中的 IOPin.out 文件，通过仿真器将其下载到 2407A DSP 上。

5. 运行程序观察结果

- 单击“Debug”菜单，“Run”项，运行程序。
- 观察实验箱控制模块上指示灯 J5 闪烁情况。
- 单击“Debug”菜单，“Halt”项，停止程序运行。

6. 修改程序重新运行

适当改变程序中“Delay(256);”语句中的延时参数，重复步骤 3-5，使指示灯约 1 秒闪烁一次。

六. 实验结果

实验程序可控制指示灯闪烁。通过 DSP 的通用 I/O 引脚可以输出状态，从而直接控制外围设备。

七. 问题与思考

* 如果需要控制较大电流驱动的设备或控制电平大于 3.3V 的设备应如何设计？
* 如果需要精确控制指示灯闪烁的时间，有什么方法？

实验三 定时器实验

一. 实验目的

1. 通过实验熟悉 LF2407A 的定时器；
2. 掌握 LF2407A 定时器的控制方法；
3. 掌握 LF2407A 的中断结构和对中断的处理流程；
4. 学会运用中断程序控制程序流程。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱（或 ICETEK 仿真器+ICETEK-LF2407-A 系统板+相关连线及电源）。

三. 实验原理

1. 通用定时器介绍及其控制方法

⑴.事件管理器模块（EV）

TMS320LF2407A DSP 片内包括两个事件管理模块 EVA 和 EVB，每个事件管理器模块包括通用定时器（GP）、比较单元以及正交编码脉冲电路。

每个事件管理模块都包含两个通用定时器,用以完成计数、同步、定时启动 ADC、定时中断等功能。

⑵.通用定时器（GP）

* 每个通用定时器包括：

-一个 16 位的定时器增/减计数的计数器 TxCNT，可读写；

-一个 16 位的定时器比较寄存器（双缓冲，带影子寄存器）TxCMPR,可读写；

-一个 16 位的定时器周期寄存器（双缓冲，带影子寄存器）TxPR,可读写；

-一个 16 位的定时器控制寄存器 TxCON，可读写；

-可选择的内部或外部输入时钟；

-用于内部或外部时钟输入的可编程的预定标器（Prescaler）；

-控制和中断逻辑，用于 4 个可屏蔽中断—下溢、溢出、定时器比较和周期中断；

-可选择方向的输入引脚 TDIRx，用于双向计数方式时选择向上或向下计数。

* 通用定时器之间可以彼此独立工作或相互同步工作,完成复杂的任务。

* 通用定时器在中断标志寄存器 EVAIFRA,EVAIFRB,EVBIFRA 和 EVBIFRB 中有 12 个中断标志位。每个通用定时器可根据以下事件产生 4 个中断：

-上溢—TxOFINF（x=1，2，3 或 4）；

-下溢—TxUFINF（x=1，2，3 或 4）；

-比较匹配--TxCINT（x=1，2，3 或 4）；

-周期匹配--TxPINT（x=1，2，3 或 4）。

* 每个通用定时器有 4 种可选择的操作模式：

-停止/保持模式；

-连续增计数模式；

-定向增/减计数模式；

-连续增/减计数模式。

相应的定时器控制寄存器 TxCON 中的位的形式决定了通用定时器的计数模式。

2. TMS320LF240x 中断结构

利用 CPU 支持的 6 个可屏蔽中断，采用集中化的中断扩展设计来满足大量的外设中断需求。

LF240x 内核提供一个不可屏蔽的中断 NMI 和 6 个按优先级获得服务的可屏蔽中断 INT1 至 INT6。而这 6 个中断级的每一个都可被很多外设中断请求共享。通过中断请求系统中的一个两级中断来扩展系统可响应的中断个数。

为了让 CPU 能区分引起中断的事件，在每个外设中断请求有效时都会产生一个唯一的外设中断向量，保存于外设中断向量寄存器（PIVR）中。实际上有两个中断向量表，CPU 的向量表用于得到一级通用中断服务子程序（GISR）；外设向量表指定外设中断子程序（SISR）。GISR 程序根据 PIVR 中的外设中断向量取值决定执行哪个 SISR。

3. 中断响应过程

外设事件要引起 CPU 中断，必须保证：外设事件的中断使能为被使能，CPU 内核级的 6 个可屏蔽中断中，相应中断也被使能。

在外设事件发生时，首先将其在外设中断控制器中的标志位置 1，从而引起 CPU 内核的 INT1—INT6 中的一个产生中断。中断服务过程中，其他可屏蔽中断将会自动被屏蔽，直到中断返回。

在软件中，当设置好相应中断标志后，开中断，进入等待中断发生的状态；外设（如定时器）中断发生时，首先跳转到相应中断高级的服务程序中（如：定时器 1 会引起 INT2 中断），在相应 GISR 子程序中，取出 PIVR 的值，根据其值再转向相应的 SISR；SISR 程序在进行服务操作之后，应将本外设的中断标志位清除以便能继续中断，然后返回。

4. 中断程序设计

用 C 语言设计中断服务程序需要用 intterupt 关键字修饰定义的中断服务函数，例如：

```
void interrupt gptime1(void);
```

中断服务函数应尽量短小，在中断服务函数中要注意对中断标志寄存器、中断屏蔽寄存器的设置，避免中断嵌套或中断丢失现象的发生。

四. 实验程序

本实验设计的程序是在上一个实验基础上修改得来，由于上一实验控制指示灯闪烁的延时控制是用循环计算方法得到的，延时不精确也不均匀，采用中断方式可

以实现指示灯的定时闪烁，时间更加准确。

对于定时器的周期寄存器为计数 40000 次产生 1 个中断，由于 DSP 工作在 40MHz 主频，正好是 1ms 中断一次，所以在中断服务程序中计算中断 500 次时改变指示灯状态，实现指示灯亮 0.5 秒，再灭 0.5 秒，即每秒闪烁 1 次。

实验程序的工程中包含了两种源代码，主程序采用 C 语言编制利于控制，中断向量表在 vector.asm 汇编语言文件中，利于直观地控制存储区分配。在工程中只需将它们添加进来即可，编译系统会自动识别分别处理完成整合工作。

实验程序的 C 语言主程序中包含了内嵌汇编语句，提供一种在需要更直接控制 DSP 状态的方法，同样的方法也能提高 C 语言部分程序的计算效率。

实验程序代码：

```
#include "2407c.h"
ioport unsigned int port000c;
#define T1MS        0x9c3f       /* 9c3fH=40000-1 */
void interrupt gptime1(void);    /* 中断服务程序,定时器计数 T1MS 次时中断
                                    调用 */
void gp_init(void);             /* 定时器初始化 */
unsigned int uWork,nCount,uWork1;
int * pf;
int j = 0;
main()
{
  int i;
  nCount=0;
  asm(" setc INTM");  /* 关中断,进行关键设置时不允许发生中断,以免干扰 */
  *WDCR=0x6f;
  *WDKEY=0x5555;
  *WDKEY=0xaaaa;          /* 关闭看门狗中断 */
  *SCSR1=0x81fe;          /* 设置 DSP 运行频率 40m */
  (*MCRB)=0;
  uWork=(*MCRC);/*将 PWM12/IOPE6 设置成通用 I/O 口,以控制实验箱上指示灯 */
  uWork&=0x0ffbf;
  (*MCRC)=uWork;
```

```
    gp_init();          /* 设置定时器 */
    *IMR=0x2;           /* 使能定时器中断(INT2) */
    *IFR=0xffff;          /* 清除中断标志 */
    asm(" clrc INTM");  /* 开中断 */
    while(1)
    {
    }
  }


void interrupt gptime1(void)        /* 中断服务程序定义，须使用 interrupt
声 */
  {
     uWork=(*PIVR);       /* 读外设中断向量寄存器*/
     switch(uWork)
     {
         case 0x27           /* T1PINT，0x27 为定时器 1 的周期中断的向量值 */
          {
          (*EVAIFRA)=0x80;       /* 清除中断标志 T1PINT */
          nCount++;
          if ( nCount>=100 )     /* 计数 100 此 100ms=0.1 秒*/
          {
              uWork=(*PEDATDIR);    /* 设置指示灯状态翻转一次 */
              uWork|=0x4000;
              uWork^=0x0040;
              (*PEDATDIR)=uWork;
              j = !j;
              (*PCDATDIR) = j + 0x100;
              port000c=uWork1++;
              uWork1%=0x100;
```

```
            nCount=0;
        }
            break;
        }
    }
}
void gp_init(void)
{
    *EVAIMRA = 0x80;    /* 使能 T1PINT 即通用定时器 1 周期中断 */
    *EVAIFRA = 0xffff;  /* 清除中断标志 */
    *GPTCONA = 0x0000;
    *T1PR    = T1MS;    /* 周期寄存器=40000 */
    *T1CNT   = 0;       /* 计数初值=0 */
    *T1CON   = 0x1040;  /* 启动计数器 */
}
```

五. 实验步骤

1. 实验准备

⑴.连接设备

① 关闭计算机和实验箱电源；

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；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；

③ 关闭实验箱上的三个开关。

⑵ 开启设备：

① 打开计算机电源

② 打开实验箱电源开关，打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；打开控制模块的电源开关；

③ 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。

⑶ 启动 Code Composer Studio

双击桌面上“  ”图标，启动 Code Composer Studio 2.2。

2. 打开工程文件,浏览程序

打开菜单“Project”的“Open”项；选择 D:\2407EDULab\Lab3-Timer 目录中的“Timer.pjt”。

在项目浏览器中，双击 led.c，激活 led.c 文件，浏览该文件的内容，理解各语句作用。打开 led.cmd，浏览并理解各语句作用，对照 C 源程序学习中断向量表的写法。

3. 编译工程

单击“Project”菜单，“Rebuild all”项，编译工程中的文件，生成 Timer.out 文件。

4. 下载程序

单击“File”菜单，“Load program…”项，选择 D:\2407EDULab\Lab3-Timer 目录中的 Timer.out 文件，通过仿真器将其下载到 2407A DSP 上。

5. 运行程序观察结果

- 单击“Debug”菜单，“Run”项，运行程序。
- 观察实验箱控制模块上指示灯 J5 闪烁情况。
- 单击“Debug”菜单，“Halt”项，停止程序运行。

5. 修改程序重新运行

适当改变程序中“#define T1MS 0x9c3f”语句中的延时参数，重复步骤 3-5，使指示灯约 1 秒闪烁两次、三次、四次。

六. 实验结果

- 指示灯在定时器的定时中断中按照设计定时闪烁。

- 使用定时器和中断服务程序可以完成许多需要定时完成的任务，比如 DSP 定时启动 A/D 转换，日常生活中的计时器计数、空调的定时启动和关闭等。

- 在调试程序时，有时需要指示程序工作的状态，可以利用指示灯的闪烁来达到，指示灯灵活的闪烁方式可表达多种状态信息。

实验四 模数转换实验

一. 实验目的

1. 掌握 A/D 转换的基本过程；
2. 熟悉 TMS320LF2407A 片内 A/D 转换模块的技术指标和常用方法。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱（或 ICETEK 仿真器+ICETEK-LF2407-A 系统板+信号源+示波器+相关连线及电源）。

三. 实验原理

1. TMS320LF2407A 模数转换模块特性

- 带内置采样和保持的 10 位模数转换模块 ADC，最小转换时间为 500ns。
- 多达 16 个的模拟输入通道（ADCIN0—ADCIN15）。
- 自动排序的能力。一次可执行最多 16 个通道的“自动转换”，而每次要转换的通道都可通过编程来选择。
- 两个独立的最多可选择 8 个转换通道的排序器可以独立工作，也可以级连后工作。
- 排序控制器可决定模拟通道转换的顺序。
- 可单独访问的 16 个转换结果寄存器。
- 多个触发源启动转换：
  - 软件设置启动标志；
  - 事件管理器（共两个）提供多个事件源；
  - 外部 ADCSOC 引脚。
- 灵活的中断控制。
- 采样和保持获取时间窗口有单独的预定标控制。
- 内置校验模式和自测试模式。

2. 模数转换工作过程

- 模数转换模块接到启动转换信号后，按照排序器的设置，开始转换第一个通道的数据；
- 经过一个采样时间的延迟后，将采样结果放入转换结果寄存器保存；
- 按顺序进行下一个通道的转换；
- 转换结束，设置标志，也可发出中断；
- 如果为连续转换方式则从新开始转换过程；否则等待下一个启动信号。

3. 模数转换的程序控制

模数转换相对于计算机来说是一个较为缓慢的过程。一般采用中断方式启动转换或保存结果，这样在 CPU 忙于其他工作时可以少占用处理时间。设计转换程序应首先考虑处理过程如何与模数转换的时间相匹配，根据实际需要选择适当的触发转换的手段，也要能及时地保存结果。

由于 TMS320LF2407A DSP 芯片内的 A/D 转换精度是 10 位的，转换结果的高 10 位为所需数值，所以在保留时应注意将结果的低 6 位去除，取出高 10 位有效数字。

四. 实验程序

本实验程序设置 DSP 采用连续采集的方式工作，同时采集两个通道（ADCIN0，ADCIN1）的模拟量输入；使用片内通用定时器 1 产生定时中断，用以定时保存转换数据。

```
#include "2407c.h"

#define ADCNUMBER 256

void interrupt gptime1(void); /* 中断服务程序，用于设置保存标志 */

void ADInit(void);            /* 初始化 A/D 转换模块和通用定时器 1 */

ioport unsigned char port000c;    /* I/O 端口用于设置 ICETEK-2407-A 板上
                                     指示灯 */

unsigned int uWork,uWork1,nADCount,nLed,*pResult1,*pResult2;

int nNewConvert,nWork;

unsigned int nADCIn0[ADCNUMBER]; /* 存储区 1，保存通道 ADCIN0 的转换结果，
循环保存 */

unsigned int nADCIn1[ADCNUMBER]; /* 存储区 2，保存通道 ADCIN1 的转换结果，
循环保存 */

int *z;

main()

{

 asm(" CLRC    SXM"); /* 清标志，关中断 */

 asm(" CLRC    OVM");

 asm(" CLRC    CNF");


 pResult1=RESULT0;

 pResult2=RESULT1;

 nNewConvert=0;

 *WDCR=0x6f;

 *WDKEY=0x5555;
```

```
*WDKEY=0xaaaa;    /* 关闭看门狗中断 */
*SCSR1=0x81fe;    /* 打开所有外设，设置时钟频率为 40MHz */
uWork=(*WSGR);    /* 设置 I/O 等待状态为 0  */
uWork&=0x0fe3f;
(*WSGR)=uWork;
ADInit();         /* 初始化 A/D 相关设备 */
*IMR=3;           /* 使能定时器中断 */
*IFR=0xffff;      /* 清所有中断标志     */
asm(" clrc INTM");   /* 开中断 */
while ( 1 )
{
   if ( nNewConvert )     /* 如果保存标志置位，以下开始转换和保存转换
                             结果 */
   {
      nNewConvert=0;        /* 清保存标志 */
      uWork=(*pResult1);    /* 取 ADCINT0 通道转换结果 */
      uWork>>=6;            /* 移位去掉低 6 位 */
      nADCIn0[nADCount]=uWork;/* 保存结果 */
      uWork=(*pResult2);    /* 取 ADCINT1 通道转换结果 */
      uWork>>=6;            /* 移位去掉低 6 位 */
      nADCIn1[nADCount]=uWork;/* 保存结果 */
      nADCount++;
        if ( nADCount>=ADCNUMBER )  /* 缓冲区满后设置指示灯闪烁 */
      {
         nADCount=0;              /* 中断位置 */
         nWork++;
         if ( nWork>=16 )
         {
            nWork=0;
```

```
                nLed++; nLed&=0x0f;
                port000c=nLed;
            }
        }
    }
  }
}
void ADInit(void)              /* 初始化设置 */
{
  int i;
  for ( i=0;i<ADCNUMBER;i++ ) /* 缓冲区清 0 */
     nADCIn0[i]=nADCIn1[i]=0;
  port000c=0;                  /* 关指示灯 */
  *ADCTRL1= 0x2040;            /* 设置连续转换模式 */
  *MAXCONV = 0x1;              /* 每次完成转换两个通道 */
  *CHSELSEQ1=0x10;             /* 转换次序，先 ADCIN0，再 ADCIN1 */
  *ADCTRL2= 0x2000;            /* 启动转换 */
  nADCount=nLed=nWork=0;
            /* 以下设置通用定时器参数 */
  *EVAIMRA = 0x80;             /* 使能 T1PINT */
  *EVAIFRA = 0xffff;           /* 清中断标志 */
  *GPTCONA = 0x0100;
  *T1PR    = 2000;         /* 保存结果周期=2000*25ns=50us=20KHz */
  *T1CNT   = 0;            /* 计数器从 0 开始计数 */
  *T1CON   = 0x1040;       /* 连续增计数方式，启动计数器 */
}

void interrupt gptime1(void)
{
```

```
    uWork1=(*PIVR);
    switch ( uWork1 )
    {
        case 0x27:
        {
        nNewConvert=1;     /* 设置保存标志 */
        (*EVAIFRA)=0x80;  /* 清中断标志位 */
            break;
        }
    }
}
```

五. 实验步骤

1. 实验准备

⑴ 连接设备：

① 关闭计算机和实验箱电源；

② 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；

③ 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；

④ 用实验箱附带的信号连接线（两边均为单声道耳机插头）连接第一信号源的波形输出端到“A/D 输入”的 ADCIN0 插座（位于实验箱底板中部，第二信号源旁边）；用信号连线连接第二信号源的输出端到“A/D 输入”的 ADCIN1 插座；

⑤ 打开实验箱上三个开关。

⑵ 开启设备：

① 开计算机电源；

② 开实验箱电源开关，打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；

③ 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。

④ 设置 Code Composer Studio 为 Emulator 方式。

⑶ 启动 Code Composer Studio

双击桌面上“  ”图标，启动 Code Composer Studio 2.2。

2. 打开工程并浏览程序

- 打开菜单“Project”的“Open”项；选择 D:\2407EDULab\Lab4-AD 目录中的“adc.pjt”。

- 在项目浏览器中，双击 ad.c，激活 ad.c 文件，浏览该文件的内容，理解各语句作用。打开 ad.cmd，浏览并理解各语句作用。

4. 编译工程

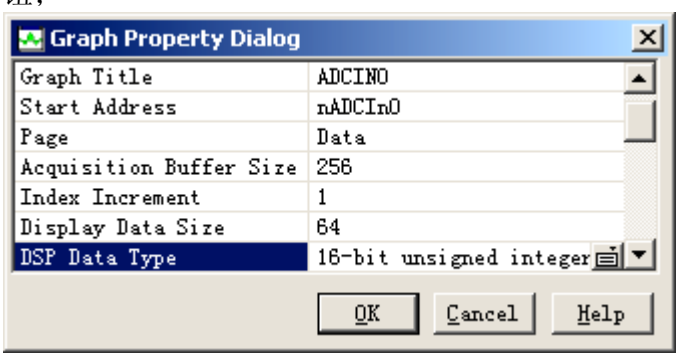
单击“Project”菜单，“Rebuild all”项，编译工程中的文件，生成 adc.out 文件。

5. 下载程序

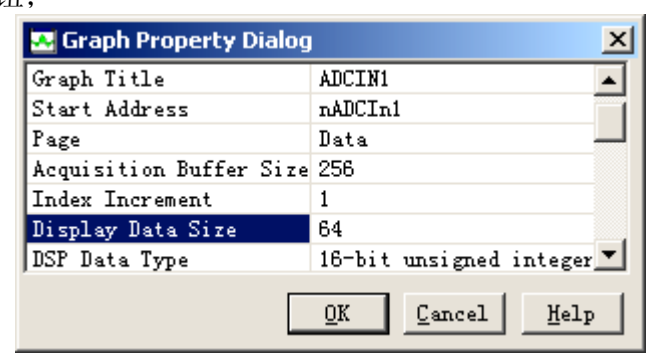
单击“File”菜单，“Load program…”项，选择 D:\2407EDULab\Lab4-AD 目录中的 adc.out 文件，通过仿真器将其下载到 2407A DSP 上。

6. 打开观察窗口

- 选择菜单“View”、“Graph”、“Time/Frequency…”做如下设置，然后单击“OK”按钮；



- 选择菜单“View”、“Graph”、“Time/Frequency…”做如下设置，然后单击“OK”按钮；



- 在弹出的图形窗口中单击鼠标右键，选择“Clear Display”。

- 在有“中断位置”注释的语句上加上软件跟踪断点（Toggle Breakpoint），即单击语句后按 F9 键；

通过设置，我们打开了两个图形窗口观察两个通道模数转换的结果。

7. 设置信号源

由于模数输入信号未经任何转换就进入 DSP，所以必须保证输入的模拟信号的幅度在 0-3.3V 之间。实验箱上信号源输出为 0-3.3V。但如果使用外接信号源，则必须

用示波器检测信号范围，保证最小值 0V 最大值 3.3V，否则容易损坏 DSP 芯片的模数采集模块。

首先设置一号信号源（上部）开关为“关”。设置实验箱上一号信号源的“频率选择”在“100Hz—1KHz”档，“波形选择”在“三角波”档，“频率微调”选择较大位置靠近最大值，“幅值微调”选择最大。开启一号信号源开关，一号信号源电源指示灯亮。

首先设置二号信号源（下部）开关为“关”。设置实验箱上二号信号源的“频率选择”在“100Hz—1KHz”档，“波形选择”在“正弦波”档，“频率微调”选择适中位置，“幅值微调”选择最大。开启二号信号源开关，二号信号源电源指示灯亮。

8. 运行程序观察结果

- 单击“Debug”菜单，“Run”项，运行程序；

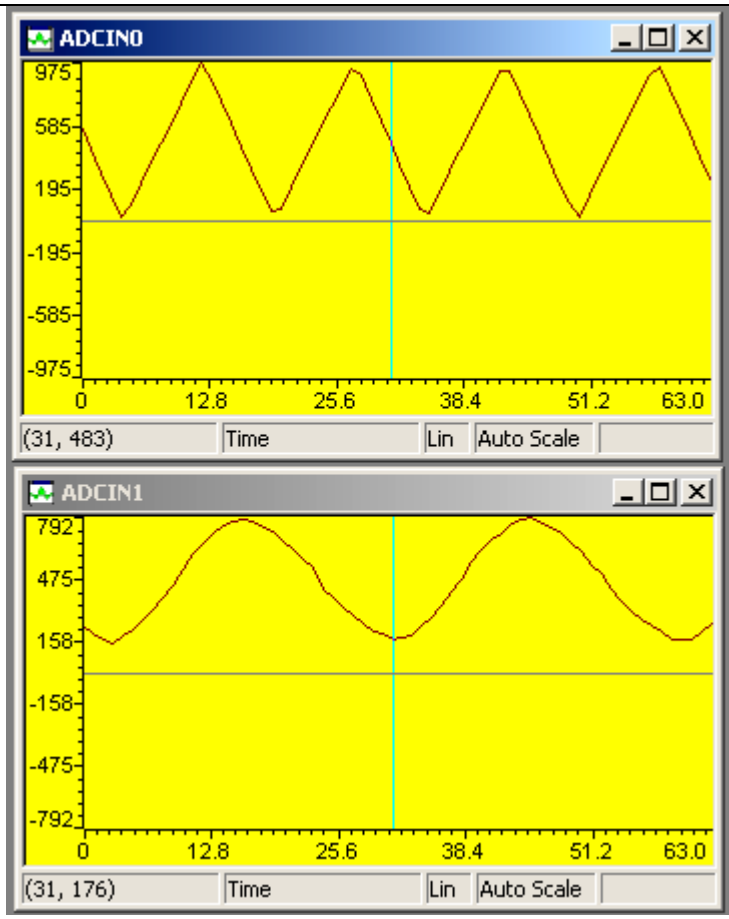
- 当程序停在所设置的软件断点上时，观察“ADCIN0”、“ADCIN1”窗口中的图形显示；

- 适当改变信号源的四个调节旋钮的位置，按 F5 健再次运行到断点位置，观察图形窗口中的显示。注意：输入信号的频率不能大于 10KHz，否则会引起混叠失真，而无法观察到波形，如果有兴趣，可以试着做一下，观察采样失真后的图形。

9. 停止运行结束实验

六. 实验结果

- 用实验中的设置，我们可以看到如下结果



- 实验程序使用定时器中断去读取模数转换的结果，这是一种较为简单的方法。用这种方法，没有考虑到 A/D 转换的精确时钟，必然会造成保存的结果中发生多点（重复）、丢点等不精确的结果。在要求较高的场合，一般采用使用定时器中断启动 A/D 转换，再相应转换完成后的中断信号，将结果保存这种方案。

七. 问题与思考

- 请修改实验程序，完成同时采集四路输入信号功能。
- 如何设置能做到用定时器中断启动 A/D 转换，再由 A/D 转换完成后的中断中断 CPU 保存结果？

实验五 数模转换实验

一. 实验目的

1. 了解数模转换的基本操作；
2. 了解 ICETEK-LF2407-A 板扩展数模转换方式；
3. 掌握数模转换程序设计方法。

二. 实验设备

计算机,示波器,ICETEK-LF2407-EDU 实验箱(或 ICETEK 仿真器+ICETEK-LF2407-A 系统板+相关连线及电源)。

三. 实验原理

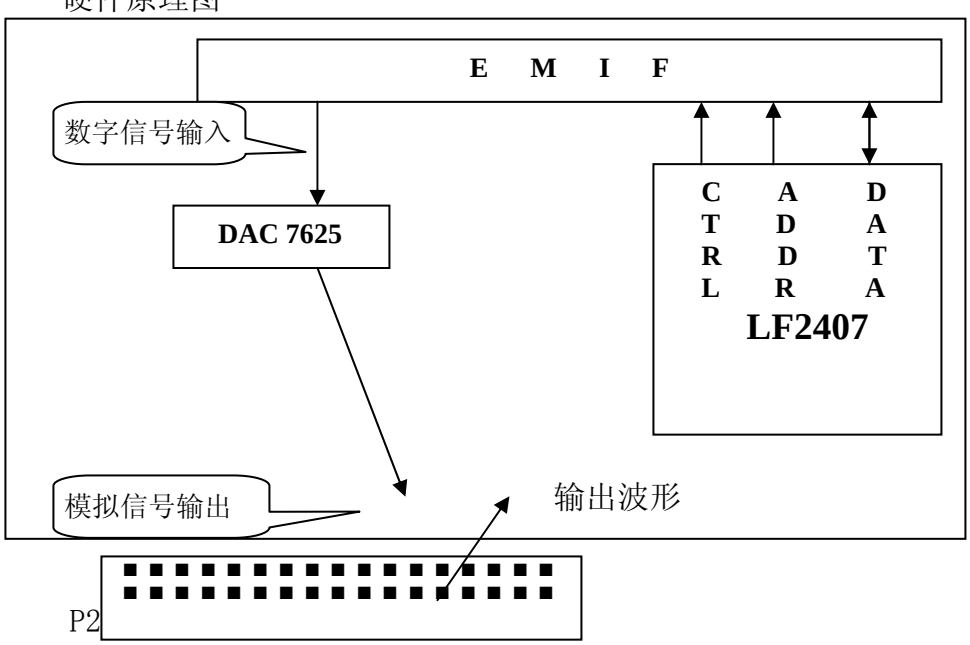
1. 数模转换操作

利用专用的数模转换芯片，可以实现将数字信号转换成模拟量输出的功能。在 ICETEK-LF2407-A 板上，使用的是 DAC7625 数模芯片，它可以实现同时转换四路模拟信号数出，并有 12 位精度，转换时间 10μs。其控制方式较为简单：首先将需要转换的数值通过数据总线传送到 DAC7625 上相应寄存器，再发送转换信号，经过一个时间延迟，转换后的模拟量就从 DAC7625 输出引脚输出。

2. DAC7625 与 TMS320LF2407A 的连接

由于 TMS320LF2407A DSP 没有数模转换输出设备，采用外扩数模转换芯片的方法。在 ICETEK-LF2407-A 板上选用的是 DAC7625。DAC7625 的转换寄存器被映射到了 DSP 的 I/O 空间，地址是 0-3，控制转换由 I/O 端口 4 的写信号控制，这部分在硬件上由译码电路（GAL 芯片）完成。在 DAC7625 的输出端，为了增加输出功率，经过一级运放再输出到板上插座上。

硬件原理图



四. 实验程序

```
#include "2407c.h"
#include "math.h"

void interrupt gptime1 (void);
void gp_init(void);
ioport unsigned int port0000;
ioport unsigned int port0001;
ioport unsigned int port0002;
ioport unsigned int port0003;
ioport unsigned int port0004;
unsigned int data[4][20];
unsigned int cnt, uWork;
main()
{
    int i;
    cnt = 0;
    for (i = 0; i < 20; i ++)
    {
        data[0][i] = 4096 / 20 * i;
        if (i < 10)
        {
            data[1][i] = 4096 / 10 * i;
            data[2][i] = 4095;
        }
        else
        {
            data[1][i] = data[1][19 - i];
            data[2][i] = 0;
        }
        data[3][i] = 4096 - data[0][i];
    }

    asm(" setc INTM");  /* 关中断，进行关键设置时不允许发生中断，以免
干扰 */
    *WDCR=0x6f;
    *WDKEY=0x5555;
    *WDKEY=0xaaaa;        /* 关闭看门狗中断 */
```

```
    *SCSR1=0x81fe;        /* 设置 DSP 运行频率 40m */
    (*MCRB)=0;
    uWork=(*MCRC);/*将 PWM12/IOPE6 设置成通用 I/O 口，以控制实验箱上指示
灯 */
    uWork&=0x0ffbf;
    (*MCRC)=uWork;
    gp_init();          /* 设置定时器 */
    *IMR=0x2;           /* 使能定时器中断(INT2) */
    *IFR=0xffff;          /* 清除中断标志 */
    asm(" clrc INTM");  /* 开中断 */
    while(1)
    {
    }
}
void gp_init(void)
{
   *EVAIMRA = 0x80;           //使能 T1PINT
   *EVAIFRA = 0xffff;         //清中断标志
   *GPTCONA = 0x0000;
   *T1PR    = 2000;
   *T1CNT   = 0;           //计数器从 0 开始计数
   *T1CON   = 0x1040;
}
void interrupt gptime1()
{
   uWork=(*PIVR);             /* 读外设中断向量寄存器*/
   switch(uWork)
   {
       case 0x27:     /* T1PINT，0x27 为定时器 1 的周期中断的向量值 */
       {
           (*EVAIFRA)=0x80;  /* 清除中断标志 T1PINT */
           port0000 = data[0][cnt];
           port0001 = data[1][cnt];
           port0002 = data[2][cnt];
           port0003 = data[3][cnt];
           port0004 = 0;
           cnt ++;
           if (cnt == 20) cnt = 0;
```

```
                break;
            }
        }
    }
```

五. 实验步骤

1. 实验准备

⑴.连接设备

① 关闭计算机和实验箱电源；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 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；

③ 关闭实验箱上三个开关。

⑵.开启设备：

④ 打开计算机电源；

⑤ 打开实验箱电源开关，打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；

⑥ 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。

⑶.设置 Code Composer Studio 为 Emulator 方式：

参见“Code Composer Studio 入门实验”之四.3。

⑷.启动 Code Composer Studio

双击桌面上“  ”图标，启动 Code Composer Studio 2.2。

2. 打开工程并浏览程序

-打开菜单“Project”的“Open”项；选择 D:\2407EDULab\Lab9-DA 目录中的“dac.pjt”。

-在项目浏览器中，双击 lfdac.asm，激活 lfdac.asm 文件，浏览该文件的内容，理解各语句作用。

3. 编译工程

单击“Project”菜单，“Rebuild all”项，编译工程中的文件，生成 dac.out 文件。

4. 下载程序

单击“File”菜单，“Load program…”项，选择 D:\2407EDULab\Lab9-DA 目录中的 dac.out 文件，通过仿真器将其下载到 2407A DSP 上。

5. 连接示波器

用信号线从实验箱底板上右侧“D/A 输出”的四个插座引线到示波器。

也可以用控制模块右侧的测试勾连接示波器。

6. 运行并观察结果

- 单击“Debug”菜单，“Run”项，运行程序；

- 观察示波器上的波形。

7. 结束运行退出 Code Composer Studio 2.2。

六. 实验结果

用示波器察看 D/A 四路输出波形，可以看到第一路输出锯齿波，第二路输出三角波，第三路输出方波，第四路输出反锯齿波。

七. 问题与思考

程序采用查表法输出波形，这样做的优点是速度快，缺点是占用存储空间较多。请考虑使用别的方法产生同样波形输出。

实验六 PWM 实验

一. 实验目的

1. 了解 TMS320LF2407A DSP 片内事件管理器模块的脉宽调制电路 PWM 的特性参数；
2. 掌握 PWM 电路的控制方法
3. 学会用程序控制产生不同占空比的 PWM 波形。

二. 实验设备

计算机，示波器，ICETEK-LF2407-EDU 实验箱（或 ICETEK 仿真器+ICETEK-LF2407-A 系统板+相关连线及电源）。

三. 实验原理

1. 脉宽调制电路 PWM 的特性

每个事件管理器模块（TMS320LF2407A DSP 片内有两个）可同时产生多达 8 路的 PWM 波形输出。由 3 个带可编程死区控制的比较单元产生独立的 3 对（即 6 个输出），以及由通用定时器比较产生的 2 个独立的 PWM 输出。

PWM 的特性如下：

- 16 位寄存器；
- 有从 0 到 16μs 的可编程死区发生器控制 PWM 输出对；
- 最小的死区宽度为 1 个 CPU 时钟周期；
- 对 PWM 频率的变动可根据需要改变 PWM 的载波频率；
- 在每个 PWM 周期内和以后可根据需要改变 PWM 脉冲的宽度；
- 外部可屏蔽的功率驱动保护中断；
- 脉冲形式发生器电路，用于可编程对称、非对称以及 4 个空间矢量 PWM 波形产生；
- 自动重装载的比较和周期存器使 CPU 的负荷最小。

2. PWM 电路的设置

在电机控制和运动控制的应用中，PWM 电路被设计为减少产生 PWM 波形的 CPU 开销和减少用户的工作量。与比较单元相关的 PWM 电路其 PWM 波形的产生由以下器存器控制：对于 EVA 模块，T1CON、COMCONA、ACTRA 和 DBTCONA；对于 EVB 模块，T3CON、COMCONB、ACTRB 和 DBTCONB。

产生 PWM 的器存器设置：

- 设置和装载 ACTRx 寄存器；
- 如果使能死区，则设置和装载 DBTCONx 寄存器；
- 设置和装载 T1PR 或 T3PR 寄存器，即规定 PWM 波形的周期；
- 初始化 CMPRX 寄存器；
- 设置和装载 COMCONx 寄存器；
- 设置和装载 T1CON 或 T3CON 寄存器，来启动比较操作；

- 更新 CMPRx 寄存器的值，使输出的 PWM 波形的占空比发生变化。

四. 实验程序

```
#include "regs240x.h"

main()
{
  unsigned int uWork;

  asm(" setc INTM");    /* 关中断 */
  asm(" setc SXM");     /* 符号位扩展有效 */
  asm(" clrc OVM");     /* 累加器中结果正常溢出 */
  asm(" clrc CNF");     /* B0 被配置为数据存储空间 */

  WDCR=0x6f;
  WDKEY=0x5555;
  WDKEY=0xaaaa;         /* 关闭看门狗中断 */

  SCSR1=0x81fe;         /* DSP 工作在 40MHz */
  IMR=0;                /* 屏蔽所有可屏蔽中断 */
  IFR=0x0ffff;          /* 清除中断标志 */
  uWork=WSGR;           /* I/O 引脚 0 等待 */
  uWork&=0x0fe3f;
  WSGR=uWork;

  MCRA=MCRA|0x0c0;      /* IOPA6-7 被配置为基本功能方式，PWM1-2 */
  ACTRA=0x06;           /* PWM2 低有效，PWM1 高有效 */
  DBTCONA=0x00;         /* 不使能死区控制 */
  CMPR1=0x1000;         /* 比较单元 1 设置 */
```

```
    CMPR2=0x3000;           /* 比较单元 2 设置 */
    T1PER=0x6000;           /* 设置定时器 1 的周期寄存器，以确定不同的输出占
                               空比 */
    COMCONA=0x8200;         /* 使能比较操作 */
    T1CON=0x1000;           /* 定时器 1 为连续增计数模式 */


    T1CON=T1CON|0x0040;       /* 启动定时器 1 */
    while ( 1 )
    {
    }
  }
```

五. 实验步骤

1. 实验准备

(1).连接设备

① 关闭计算机和实验箱电源；

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；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；

③ 关闭实验箱上三个开关。

(2).开启设备

① 打开计算机电源；

② 打开实验箱电源开关，打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；

③ 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。

(3).启动 Code Composer Studio

双击桌面上“  ”图标，启动 Code Composer Studio 2.2。

2. 打开工程并浏览程序

- 打开菜单“Project”的“Open”项；选择 D:\2407EDULab\Lab6-PWM 目录中的“pwm.pjt”。

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

3. 编译工程

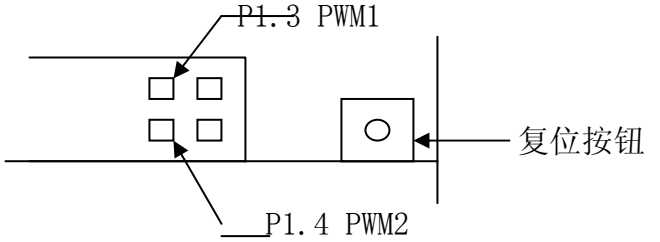
单击“Project”菜单，“Rebuild all”项，编译工程中的文件，生成 pwm.out 文件。

4. 下载程序

单击“File”菜单，“Load program…”项，选择 D:\2407EDULab\Lab6-PWM 目录中的 pwm.out 文件，通过仿真器将其下载到 2407A DSP 上。

5. 连接示波器

连接示波器探头的地线与实验箱地线输出；测试 ICETEK-LF2407-A 板扩展插座 P1（位于板上复位按钮旁边，未与实验箱连接的插座）的第 3 和第 4 脚，即 PWM1 和 PWM2 的输出，如图：



注：P1 插座的地线为 P1.17 和 P1.18。

6. 运行并观察结果

- 单击“Debug”菜单，“Run”项，运行程序；
- 观察示波器上的波形。

7. 重新设置参数

- 停止运行；
- 修改程序的“T1CON=0x1000;”语句，将计时器设置为“连续增/减计数方式”；
- 再次运行程序观察示波器上的波形的改变；
- 再次修改程序调整 PWM 的参数：载波频率、占空比，运行并观察得到的波形是否与设计的一致。

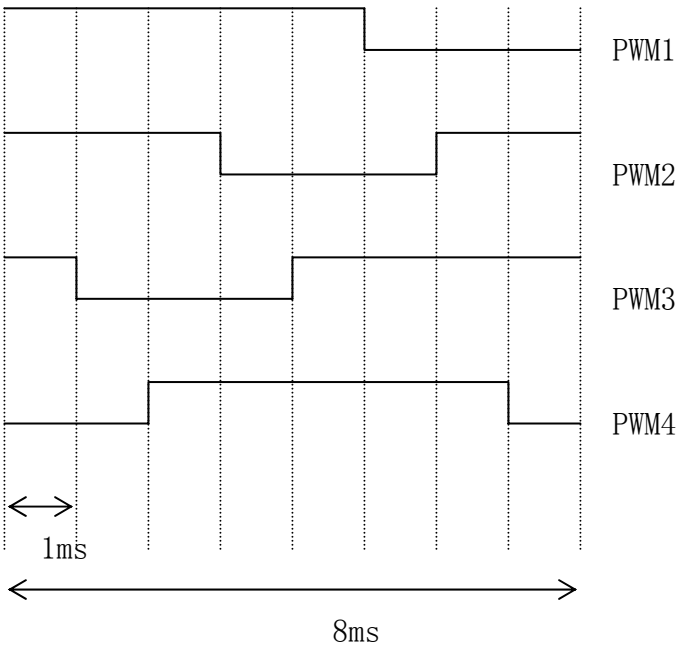
8. 结束运行退出 Code Composer Studio。

四. 实验结果

通过示波器可观察到不同占空比的 PWM 输出波形，其载波频率、占空比与程序中对控制寄存器的设置相关。

五. 问题与思考

试设计四路（PWM1,PWM2,PWM3,PWM4）输出，载波频率为 8ms，波形关系如下（一个周期）：


PWM1
PWM2
PWM3
PWM4
1ms
8ms

实验七 外设控制实验—发光二极管阵列显示实验

一. 实验目的

通过实验学习使用 2407A DSP 的扩展 I/O 端口控制外围设备的方法，了解发光二极管阵列的控制编程方法。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱。

三. 实验原理

ICETEK-LF2407-A 是一块以 TMS320LF2407ADSP 为核心的 DSP 扩展评估板，它通过扩展接口与实验箱的显示/控制模块连接，可以控制其各种外围设备。

发光二极管显示阵列的显示是由 I/O 扩展端口控制，DSP 须将显示的图形按列的顺序存储起来(8×8 点阵，8 个字节，高位在下方，低位在上方)，然后定时刷新控制显示。具体方法是，将以下控制字按先后顺序、每两个为一组发送到端口 0x8005，发送完毕后，隔不太长的时间(以人眼观察不闪烁的时间间隔)再发送一遍。由于位值为“0”时点亮，所以需要将显示的数据取反。

0x01,第 8 列数据取反，0x02,第 7 列数据取反，
0x04,第 6 列数据取反，0x08,第 5 列数据取反，
0x10,第 4 列数据取反，0x20,第 3 列数据取反，
0x40,第 2 列数据取反，0x80,第 1 列数据取反，

四. 实验程序

```
#include "2407c.h"

ioport unsigned int port8000;

ioport unsigned int port8005;

ioport unsigned int port8007;


void Delay(unsigned int nTime);      // 延时子程序

void RefreshLEDArray();           // 刷新显示

void SetLEDArray(int nNumber);       // 修改显示内容


unsigned char ledbuf[8],ledx[8];

unsigned char ledkey[10][8]=

{
```

```
  {0x00,0x00,0x7C,0x82,0x82,0x82,0x7C,0x00},     //0
  {0x00,0x00,0x00,0x84,0xFE,0x80,0x00,0x00},     //1
  {0x00,0x00,0x84,0xC2,0xA2,0x92,0x8C,0x00},     //2
  {0x00,0x00,0x44,0x92,0x92,0x92,0x6C,0x00},
  {0x00,0x00,0x30,0x28,0x24,0xFE,0x20,0x00},
  {0x00,0x00,0x4E,0x92,0x92,0x92,0x62,0x00},
  {0x00,0x00,0x7C,0x92,0x92,0x92,0x64,0x00},
  {0x00,0x00,0x02,0xC2,0x32,0x0A,0x06,0x00},
  {0x00,0x00,0x6C,0x92,0x92,0x92,0x6C,0x00},
  {0x00,0x00,0x4C,0x92,0x92,0x92,0x7C,0x00}
};

main()
{
  int nCount;
  asm(" setc INTM");  /* 关中断,进行关键设置时不允许发生中断,以免干扰 */

  *WDCR=0x6f;
  *WDKEY=0x5555;
  *WDKEY=0xaaaa;        /* 关闭看门狗中断 */

  *SCSR1=0x81fe;        /* 设置 DSP 运行频率 40m */
  (*MCRB)=0;

  port8000=0;           // 初始化 ICETEK-CTR
  port8000=0x80;
  port8000=0;
  port8007=0;           // 关闭东西方向的交通灯
  port8007=0x40;        // 关闭南北方向的交通灯
```

```
  port8007=0x0c1;
  for ( nCount=0;nCount<8;nCount++ )
  {
      ledbuf[nCount]=0x0ff;        // 显示为空白
      ledx[nCount]=(nCount<<4);    // 生成显示列控制字
  }
  RefreshLEDArray();
  nCount=0;
  for (;;)
  {
      SetLEDArray(nCount);
      Delay(256);
      nCount++;
      nCount%=10;
  }
}

void Delay(unsigned int nDelay)
{
  int ii,jj,kk=0;
  for ( ii=0;ii<nDelay;ii++ )
  {
      for ( jj=0;jj<64;jj++ )
      {
          RefreshLEDArray();
          kk++;
      }
  }
}
```

```
void RefreshLEDArray()
{
  int i;
  for ( i=0;i<8;i++ )
  {
      port8000=ledx[i];
      port8005=ledbuf[i];
  }
}

void SetLEDArray(int nNumber)
{
  int i;
  for ( i=0;i<8;i++ )
      ledbuf[i]=~ledkey[nNumber][7-i];
}
```

五. 实验步骤

1. 实验准备

⑴.连接设备

① 关闭计算机和实验箱电源；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 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；

③ 关闭实验箱上三个开关；

⑵.开启设备

① 打开计算机电源；

② 打开实验箱电源开关，ICETEK-CTR 板上 J2、J3 灯亮；

③ 打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；

④ 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。

⑶.设置 Code Composer Studio 为 Emulator 方式

(4).启动 Code Composer Studio

2. 打开工程并浏览程序，工程目录为 D:\2407EDULab\Lab12-LedArray
3. 编译并下载程序
4. 运行程序，观察结果

如果显示出现乱码，可退出 CCS 后关闭实验箱，再打开重新做一次。

5. 停止程序运行并退出

六. 实验结果与分析

实验结果：可以观察到发光二极管阵列显示从 0 到 9 的计数。

分析：本程序使用循环延时的方法，如果想实现较为精确的定时，可使用通用计时器，在通用计时器中断中取得延时，改变显示内容。另外本程序中 DSP 一直在做刷新显示的工作，如果使用通用计时器定时刷新显示，将能减少 DSP 用于显示的操作。适当更新显示可取得动画效果。

七. 问题与思考

试设计用定时器定时刷新的程序，并显示秒计数的最低位。

实验八 外设控制实验—液晶显示器控制显示实验

一. 实验目的

通过实验学习使用 2407ADSP 的扩展 I/O 端口控制外围设备的方法，了解液晶显示器的显示控制原理及编程方法。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱。

三. 实验原理

ICETEK-LF2407-A 是一块以 TMS320LF2407ADSP 为核心的 DSP 扩展评估板，它通过扩展接口与实验箱的显示/控制模块连接，可以控制其各种外围设备。

液晶显示模块的访问、控制是由 2407ADSP 对扩展 I/O 接口的操作完成。

控制 I/O 口的寻址：命令控制 I/O 接口的地址为 0x8001，数据控制 I/O 接口的地址为 0x8003 和 0x8004，辅助控制 I/O 接口的地址为 0x8002。

显示控制方法：

- 液晶显示模块中有两片显示缓冲存储器，分别对应屏幕显示的象素，向其中写入数值将改变显示，写入“1”则显示一点，写入“0”则不显示。其地址与象素的对应方式如下：

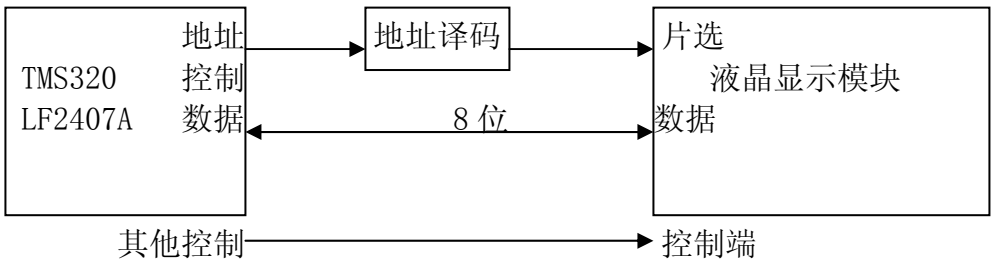
左侧显示内存						右侧显示内存					
Y=	0	1	•••	62	63	0	1	•••	62	63	行号
X=0	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	0 ↓ 7
↓	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	8 ↓ 55
X=7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	DB0 ↓ DB7	56 ↓ 63

-发送控制命令：向液晶显示模块发送控制命令的方法是通过向命令控制 I/O 接口写入命令控制字，然后再向辅助控制接口写入 0。下面给出的是基本命令字、解释和 C 语言控制语句举例。

- 显示开关：0x3f 打开显示；0x3e 关闭显示；
- 设置显示起始行：0x0c0+起始行取值，其中起始行取值为 0 至 63；
- 设置操作页：0x0b8+页号，其中页号取值为 0-7；
- 设置操作列：0x40+列号，其中列号为取值为 0-63；

- 写显示数据：在使用命令控制字选择操作位置(页数、列数)之后，可以将待显示的数据写入液晶显示模块的缓存。将数据发送到相应数据控制 I/O 接口即可。

* 液晶显示器与 DSP 的连接：



* 数据信号的传送：由于液晶显示模块相对运行在 40MHz 主频下的 DSP 属于较为慢速设备，连接时需要考虑数据线上信号的等待问题；

* 电平转换：由于 DSP 为 3.3V 设备，而液晶显示模块属于 5V 设备，所以在连接控制线、数据线时需要加电平隔离和转换设备，如：ICETEK-CTR 板上使用了 74LS245。

四. 实验程序

```
include "2407c.h"


#define LCDDELAY 1
#define LCDCMDTURNON 0x3f
#define LCDCMDTURNOFF 0x3e
#define LCDCMDSTARTLINE 0xc0
#define LCDCMDPAGE 0xb8
#define LCDCMDVERADDRESS 0x40


ioport unsigned int port8001;
ioport unsigned int port8002;
ioport unsigned int port8003;
ioport unsigned int port8004;


void Delay(unsigned int nTime);      // 延时子程序
void TurnOnLCD();                    // 打开显示
void LCDCLS();                       // 清除屏幕显示内容
```

```
unsigned char ledkey[10][8]=
{
  {0x00,0x00,0x7C,0x82,0x82,0x82,0x7C,0x00},     //0
  {0x00,0x00,0x00,0x84,0xFE,0x80,0x00,0x00},     //1
  {0x00,0x00,0x84,0xC2,0xA2,0x92,0x8C,0x00},     //2
  {0x00,0x00,0x44,0x92,0x92,0x92,0x6C,0x00},
  {0x00,0x00,0x30,0x28,0x24,0xFE,0x20,0x00},
  {0x00,0x00,0x4E,0x92,0x92,0x92,0x62,0x00},
  {0x00,0x00,0x7C,0x92,0x92,0x92,0x64,0x00},
  {0x00,0x00,0x02,0xC2,0x32,0x0A,0x06,0x00},
  {0x00,0x00,0x6C,0x92,0x92,0x92,0x6C,0x00},
  {0x00,0x00,0x4C,0x92,0x92,0x92,0x7C,0x00}
};

main()
{
  int i,nCount=0;
  unsigned int uWork;

  *WDCR=0x6f;
  *WDKEY=0x5555;
  *WDKEY=0xaaaa;
  *SCSR1=0x81fe;
  *IMR=0x0;
  *IFR=0xffff;
  uWork=(*WSGR);
  uWork&=0x0fe3f;
  (*WSGR)=uWork;
```

```
  LCDCLS();                          // 清除显示内存
  TurnOnLCD();                       // 打开显示
  port8001=LCDCMDSTARTLINE;          // 设置显示起始行
  port8002=0;
  Delay(LCDDELAY);
  for (;;)
  {
      port8001=LCDCMDPAGE;           // 设置操作页=0
      port8002=0;
      Delay(LCDDELAY);
      port8001=LCDCMDVERADDRESS;     // 起始列=0
      port8002=0;
      Delay(LCDDELAY);
      for ( i=0;i<8;i++ )
      {
          port8003=ledkey[nCount][i]; // 屏幕左侧第 1 至 8 行第 i 列赋值
          port8002=0;        // (赋值后当前操作列自动加 1，所以不需设置)
          port8002=0;
          Delay(LCDDELAY);
      }
      Delay(2048);
      nCount++;
      nCount%=10;
  }
}

void Delay(unsigned int nDelay)
{
  int ii,jj,kk=0;
```

```
  for ( ii=0;ii<nDelay;ii++ )
  {
      for ( jj=0;jj<64;jj++ )
      {
          kk++;
      }
  }
}

void TurnOnLCD()
{
  port8001=LCDCMDTURNON;
  port8002=0;
  Delay(1024);
  port8001=LCDCMDSTARTLINE;
  port8002=0;
  Delay(LCDDELAY);
}

void LCDCLS()
{
  int i,j;
  port8001=LCDCMDSTARTLINE;
  port8002=0;
  Delay(LCDDELAY);
  for ( i=0;i<8;i++ )
  {
      port8001=LCDCMDPAGE+i;
      port8002=0;
```

```
        Delay(LCDDELAY);
        port8001=LCDCMDVERADDRESS;
        port8002=0;
        Delay(LCDDELAY);
        for ( j=0;j<64;j++ )
        {
            port8003=0;
            port8002=0;
            Delay(LCDDELAY);
        }
        port8001=LCDCMDPAGE+i;
        port8002=0;
        Delay(LCDDELAY);
        port8001=LCDCMDVERADDRESS;
        port8002=0;
        Delay(LCDDELAY);
        for ( j=0;j<64;j++ )
        {
            port8004=0;
            port8002=0;
            Delay(LCDDELAY);
        }
    }
  }
```

五. 实验步骤

1.实验准备

⑴连接设备

① 关闭计算机和实验箱电源；

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近

DSP 芯片端），即设置 DSP 工作在 MP 方式；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；

③ 关闭实验箱上三个开关。

⑵开启设备

④ 打开计算机电源；

⑤ 打开实验箱电源开关，ICETEK-CTR 板上 J2、J3 灯亮；

⑥ 打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；

⑦ 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。

⑶设置 Code Composer Studio 为 Emulator 方式。

⑷启动 Code Composer Studio

2.打开工程并浏览程序，工程目录为 D:\2407EDULab\Lab8-LCD

3.编译并下载程序

4.运行程序，观察结果

5. 将内层循环中的“port8003=ledkey[nCount][i];”语句改为“port8004=ledkey[nCount][i];”，重复步骤 3-4，实现在屏幕右侧显示。

6.更改程序中对页、列的设置，实现不同位置的显示。

7.自己设计一些控制语句，实现不同显示效果。

8.停止程序运行并退出

六. 实验结果与分析

实验结果：可以观察到液晶显示从 0 到 9 的计数。

分析：灵活使用控制字，可以实现复杂多变的显示。当使用点阵图形显示时需要在 DSP 内存中建立图形存储缓冲；适当更新显示可取得动画效果。在实际生活中观察点阵显示的霓虹灯广告、交通指示牌、报站牌等领会这种控制的具体应用。

七. 问题与思考

试设计程序在液晶显示屏上显示计时时钟，精确到秒，形式为“时时：分分：秒秒”。

实验九 外设控制实验—键盘输入实验

一. 实验目的

通过实验学习使用 2407ADSP 的扩展 I/O 端口接收外围设备信息的方法，了解键盘的使用原理及编程方法。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱。

三. 实验原理

ICETEK-LF2407-A 是一块以 TMS320LF2407ADSP 为核心的 DSP 扩展评估板，它通过扩展接口与实验箱的显示/控制模块连接，可以控制其各种外围设备，也可以接收外设发送的各种数据、信息。

键盘的扫描码由 DSP 的 I/O 扩展地址 0x8001 给出，当有键盘输入时，读此端口得到扫描码，当无键被按下时读此端口的结果为 0。各按键的扫描码排列如下所示。

0x18,0x14,0x12,0x11
0x28,0x24,0x22,0x21
0x48,0x44,0x42,0x41
0x88,0x84,0x82,0x81

四. 实验程序

```
#include "2407c.h"

#include "scancode.h"

ioport unsigned int port8000;
ioport unsigned int port8001;
ioport unsigned int port8002;
ioport unsigned int port8005;
ioport unsigned int port8007;

void Delay(unsigned int nTime);      // 延时子程序
void RefreshLEDArray();             // 刷新显示
void SetLEDArray(int nNumber);       // 修改显示内容
```

```
char ConvertScanToChar(unsigned char cScanCode);  // 将键盘扫描码转换为字符


unsigned int nScreenBuffer[1024];
unsigned char ledbuf[8],ledx[8];
unsigned char ledkey[10][8]=
{
  {0x00,0x00,0x7C,0x82,0x82,0x82,0x7C,0x00},
  {0x00,0x00,0x00,0x84,0xFE,0x80,0x00,0x00},    //1
  {0x00,0x00,0x84,0xC2,0xA2,0x92,0x8C,0x00},    //2
  {0x00,0x00,0x44,0x92,0x92,0x92,0x6C,0x00},
  {0x00,0x00,0x30,0x28,0x24,0xFE,0x20,0x00},
  {0x00,0x00,0x4E,0x92,0x92,0x92,0x62,0x00},
  {0x00,0x00,0x7C,0x92,0x92,0x92,0x64,0x00},
  {0x00,0x00,0x02,0xC2,0x32,0x0A,0x06,0x00},
  {0x00,0x00,0x6C,0x92,0x92,0x92,0x6C,0x00},
  {0x00,0x00,0x4C,0x92,0x92,0x92,0x7C,0x00}
};
unsigned int uPort8000;

main()
{
  int nCount,nCursorCount;
  unsigned int nScanCode,nKeyCode;
  unsigned char cKey;

  asm("setc INTM");  /* 关中断,进行关键设置时不允许发生中断,以免干扰 */

  *WDCR=0x6f;
```

```
*WDKEY=0x5555;
*WDKEY=0xaaaa;        /* 关闭看门狗中断 */

*SCSR1=0x81fe;        /* 打开所有外设，设置时钟频率 40MHz */
(*MCRB)=0;

port8000=0;               // 初始化 ICETEK-CTR
port8000=0x80;
port8000=0;
port8007=0;               // 关闭东西方向的交通灯
port8007=0x40;            // 关闭南北方向的交通灯
port8007=0x0c1;
uPort8000=port8002;       // 清键盘缓冲区
for ( nCount=0;nCount<8;nCount++ )
{
    ledbuf[nCount]=0x0ff;         // 显示为空白
    ledx[nCount]=(nCount<<4);     // 生成显示列控制字
}
RefreshLEDArray();
nCount=nCursorCount=0;

for (;;)
{
    nScanCode=port8001;
    nScanCode&=0x0ff;
    uPort8000=port8002;
    if ( nScanCode!=0 && nScanCode!=0x0ff )
    {
        if ( nScanCode==SCANCODE_Enter )   break;
```

```
            else
            {
                cKey=ConvertScanToChar(nScanCode);
                if ( cKey!=0 )
                {
                    nKeyCode=cKey-'0';
                    SetLEDArray(nKeyCode);
                }
            }
        }
        Delay(16);
        nCursorCount++; nCursorCount%=4;
        if ( nCursorCount==0 )  ledbuf[7]^=0x80;
    }
    for ( nCount=0;nCount<8;nCount++ )
    {
        ledbuf[nCount]=0x0ff;        // 显示为空白
        ledx[nCount]=(nCount<<4);    // 生成显示列控制字
    }
    RefreshLEDArray();
    exit(0);
}

void Delay(unsigned int nDelay)
{
    int ii,jj,kk=0;
    for ( ii=0;ii<nDelay;ii++ )
    {
        for ( jj=0;jj<64;jj++ )
```

```
        {
            RefreshLEDArray();
            kk++;
        }
    }
}

void RefreshLEDArray()
{
    int i;
    for ( i=0;i<8;i++ )
    {
        port8000=ledx[i];
        port8005=ledbuf[i];
    }
}

void SetLEDArray(int nNumber)
{
    int i;
    for ( i=0;i<8;i++ )
        ledbuf[i]=~ledkey[nNumber][7-i];
}

char ConvertScanToChar(unsigned char cScanCode)
{
    char cReturn;

    cReturn=0;
```

```
    switch ( cScanCode )
    {
        case SCANCODE_0: cReturn='0'; break;
        case SCANCODE_1: cReturn='1'; break;
        case SCANCODE_2: cReturn='2'; break;
        case SCANCODE_3: cReturn='3'; break;
        case SCANCODE_4: cReturn='4'; break;
        case SCANCODE_5: cReturn='5'; break;
        case SCANCODE_6: cReturn='6'; break;
        case SCANCODE_7: cReturn='7'; break;
        case SCANCODE_8: cReturn='8'; break;
        case SCANCODE_9: cReturn='9'; break;
    }

    return cReturn;
  }
```

五. 实验步骤

1.实验准备

⑴连接设备

① 关闭计算机和实验箱电源。

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式。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。

③ 关闭实验箱上三个开关。

④ 设置 ICETEK-CTR 板上拨动开关（位于板子左上角）的第 1 位 BS3 为“OFF”状态。

⑵开启设备

① 打开计算机电源。

② 打开实验箱电源开关，ICETEK-CTR 板上 J2、J3 灯亮。

③ 打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮。

④ 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。

⑶设置 Code Composer Studio 为 Emulator 方式

⑷启动 Code Composer Studio

2.打开工程并浏览程序，工程目录为 D:\2407EDULab\Lab9-Key

3.编译并下载程序

4.运行程序，观察结果

按下键盘上一些健，观察显示是否正确。

5.停止程序运行并退出

六. 实验结果

实验结果：可以观察到发光二极管阵列显示键盘输入字符。

分析：在程序中加入分支语句实现对不同键盘输入值的处理或支持控制型按键；修改程序中键值查找表可实现按键的重新布局或修改。

实验十 外设控制实验—步进电机控制实验

一. 实验目的

通过实验学习使用 2407ADSP 的扩展 I/O 端口控制外围设备信息的方法，掌握使用 2407ADSP 通用计时器的控制原理及中断服务程序的编程方法；了解步进电机的控制方法。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱。

三. 实验原理

ICETEK-LF2407-A 是一块以 TMS320LF2407ADSP 为核心的 DSP 扩展评估板，它通过扩展接口与实验箱的显示/控制模块连接，可以控制其各种外围设备。

步进电机是由 DSP 通用 I/O 管脚输出直接控制。步进电机的起动频率大于 500PPS(拍每秒)，空载运行频率大于 900PPS。2407A 的通用 I/O 口 PWM11/IOPE5 控制电机的转动频率，TDIRB/IOPF4 控制转动方向。

控制的方法是使用 DSP 通用定时器设置 PWM11/IOPE5 以一定的频率改变高低状态，输出方波，设置 TDIRB/IOPF4 为高电平则顺时针转动，低电平为逆时针转动。

四. 实验程序

```
#include "2407c.h"
#include "scancode.h"

#define T46uS     0x0d40
void gp_init(void);
void Delay(unsigned int nTime);
void interrupt time(void);
char ConvertScanToChar(unsigned char cScanCode);
ioport unsigned int port8000;
ioport unsigned int port8001;
ioport unsigned int port8002;
ioport unsigned int port8007;
unsigned int uWork,nCount;
unsigned int pwm1[8]={ 0x8e,0x8c,0x8d,0x89,0x8b,0x83,0x87,0x86 };
unsigned int pwm2[8]={ 0x86,0x87,0x83,0x8b,0x89,0x8d,0x8c,0x8e };
int nAddStep,nStep;

main()
{
```

```
    unsigned int nScanCode,uWork1,i;
    unsigned char cKey;

    asm(" setc INTM");  /* 关中断，进行关键设置时不允许发生中断，以免
干扰 */
     (*PIRQR0)=0;
     (*EVAIFRA)=0x80;
    *EVAIFRA = 0xffff;
     *WDCR=0x6f;
     *WDKEY=0x5555;
     *WDKEY=0xaaaa;        /* 关闭看门狗中断 */

     *SCSR1=0x81fe;        /* 设置 DSP 运行频率 40m */
     (*MCRB)=0;

     nStep=0; nAddStep=1;
     port8000=0;           // 初始化 ICETEK-CTR
     port8000=0x80;
     port8000=0x0;
     port8007=0;           // 关闭东西方向的交通灯
     port8007=0x40;        // 关闭南北方向的交通灯
     uWork1=port8002;
     port8007=0x0c3;
     gp_init();
     *IMR=0x3;
     *IFR=0xffff;
     asm("  clrc   INTM");
     while ( 1 )
     {
        if ( nCount>16 )
        {

           nCount=0;
           nScanCode=port8001;
           nScanCode&=0x0ff;
           uWork=port8002;
           if ( nScanCode!=0 )
           {
              if ( nScanCode==SCANCODE_Enter )   break;
```

```
                else
                {
                    cKey=ConvertScanToChar(nScanCode);
                }
                cKey=ConvertScanToChar(nScanCode);
                if ( cKey!=0 )
                {
                    if ( cKey=='4' ) nAddStep=1;
                    else if ( cKey=='6' )    nAddStep=-1;
                }
            }
        }
        Delay(10);
    }
}

void Delay(unsigned int nDelay)
{
    int i,j,k=0;
    for ( i=0;i<nDelay;i++ )
        for ( j=0;j<64;j++ )
            k++;
}

void interrupt gptime1(void)
{
   uWork=(*PIVR);
   switch(uWork)
   {
        case 0x27:
        {
            (*EVAIFRA)=0x80;
        port8007=pwm2[nStep];
        nStep+=nAddStep;
        if ( nStep<0 )    nStep=7;
        else if ( nStep>7 )  nStep=0;
        nCount++;
            break;
        }
```

```
    }

}
void gp_init(void)
{
   *EVAIMRA = 0x80;
   *EVAIFRA = 0xffff;
   *GPTCONA = 0x0100;
   *T1PR    = T46uS*9/5;
   *T1CNT   = 0;
   *T1CON   = 0x1340;
}

char ConvertScanToChar(unsigned char cScanCode)
{
    char cReturn;

    cReturn=0;
    switch ( cScanCode )
    {
        case SCANCODE_0: cReturn='0'; break;
        case SCANCODE_1: cReturn='1'; break;
        case SCANCODE_2: cReturn='2'; break;
        case SCANCODE_3: cReturn='3'; break;
        case SCANCODE_4: cReturn='4'; break;
        case SCANCODE_5: cReturn='5'; break;
        case SCANCODE_6: cReturn='6'; break;
        case SCANCODE_7: cReturn='7'; break;
        case SCANCODE_8: cReturn='8'; break;
        case SCANCODE_9: cReturn='9'; break;
    }

    return cReturn;
}
```

五. 实验步骤

1.实验准备

⑴连接设备

① 关闭计算机和实验箱电源。

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式。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。

③ 关闭实验箱上三个开关。

⑵开启设备

① 打开计算机电源。

② 打开实验箱电源开关，ICETEK-CTR 板上 J2、J3 灯亮；打开位于实验箱中部的电机电源开关，ICETEK-CTR 板上 J4 灯亮。

③ 打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮。

④ 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。

⑶设置 Code Composer Studio 为 Emulator 方式

⑷启动 Code Composer Studio

2.打开工程并浏览程序，工程目录为 D:\2407EDULab\Lab16-Motor

3.编译并下载程序

4.运行程序，观察结果

电机转动时按下 ICETEK-CTR 板上的键盘“4”和“6” 键，控制电机转动方向

5.停止程序运行并退出

六. 实验结果

实验结果：可以看到显示/控制模块上的电机指针在转动，使用“4”和“6”键可控制其转动方向。

分析：使用优化的程序可控制电机更平滑地转动，平滑地改变频率可使马达的摆动减到最小。

实验十一 直流电机控制实验

一. 实验目的

1. 学习用 C 语言编制中断程序，控制 LF2407 DSP 的通用 I/O 管脚产生不同占空比的 PWM 信号。

2. 学习 LF2407DSP 的通用 I/O 管脚的控制方法。

3. 学习直流电机的控制原理和控制方法。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱。

三. 实验原理

1. TMS320LF2407DSP 的通用 I/O 引脚

TMS320LF2407DSP 可以提供超过 40 个通用 I/O 引脚。

每个 IO 均有一组控制寄存器设置复用状态，这一组寄存器的访问是通过映射在 DSP 数据区的地址进行。

通过设置各管脚的工作方式和状态，可以实现将它们当成通用 I/O 引脚使用。

2. 直流电机控制

直流电动机是最早出现的电动机，也是最早能实现调速的电动机。近年来，直流电动机的结构和控制方式都发生了很大的变化。随着计算机进入控制领域，以及新型的电力电子功率元器件的不断出现，使采用全控型的开关功率元件进行脉宽调制(Puls Width Modulation，简称 PWM)控制方式已成为绝对主流。

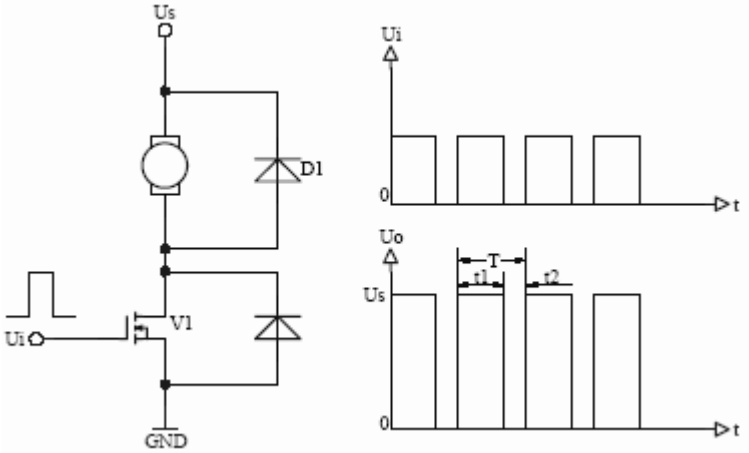
PWM 调压调速原理

直流电动机转速 n 的表达式为：

$$n = \frac{U - IR}{K\Phi}$$

其中，U 为电枢端电压；I 为电枢电流；R 为电枢电路总电阻；Φ为每极磁通量；K 为电动机结构参数。

所以直流电动机的转速控制方法可分为两类：对励磁磁通进行控制的励磁控制法和对电枢电压进行控制的电枢控制法。其中励磁控制法在低速时受磁极饱和的限制，在高速时受换向火花和换向器结构强度的限制，并且励磁线圈电感较大，动态响应较差，所以这种控制方法用得很少。现在，大多数应用场合都使用电枢控制法。绝大多数直流电机采用开关驱动方式。开关驱动方式是使半导体功率器件工作在开关状态，通过脉宽调制 PWM 来控制电动机电枢电压，实现调速。



上图是利用开关管对直流电动机进行 PWM 调速控制的原理图和输入输出电压波形。图中，当开关管 MOSFET 的栅极输入高电平时，开关管导通，直流电动机电枢绕组两端有电压 Us。t1 秒后，栅极输入变为低电平，开关管截止，电动机电枢两端电压为 0。t2 秒后，栅极输入重新变为高电平，开关管的动作重复前面的过程。这样，对应着输入的电平高低，直流电动机电枢绕组两端的电压波形如图中所示。电动机的电枢绕组两端的电压平均值 Uo 为

$$U_O = \frac{t_1 U_S + 0}{t_1 + t_2} = \frac{t_1}{T} U_S = \alpha U_S$$

式中 α 为占空比， α =t1/T

占空比 α 表示了在一个周期 T 里，开关管导通的时间与周期的比值。α 的变化范围为 0≤ α ≤1。由此式可知，当电源电压 Us 不变的情况下，电枢的端电压的平均值 Uo 取决于占空比 α 的大小，改变 α 值就可以改变端电压的平均值，从而达到调速的目的，这就是 PWM 调速原理。

PWM 调速方法

在 PWM 调速时，占空比 α 是一个重要参数。以下 3 种方法都可以改变占空比的值：

(1)定宽调频法：这种方法是保持 t1 不变，只改变 t2，这样使周期 T(或频率)也随之改变。

(2)调宽调频法：这种方法是保持 t2 不变，只改变 t1，这样使周期 T(或频率)也随之改变。

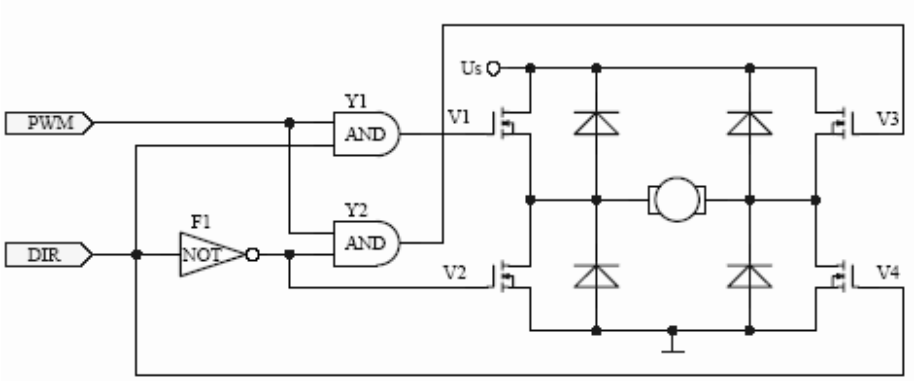
(3)定频调宽法：这种方法是使周期 T(或频率)保持不变，而改变 t1 和 t2。

前两种方法由于在调速时改变了控制脉冲的周期(或频率)，当控制脉冲的频率与系统的固有频率接近时，将会引起震荡，因此这两种方法用得很少。目前，在直流电动机的控制中，主要使用定频调宽法。

3．ICETEK-CTR 直流电机模块

原理图

ICETEK-CTR 即显示/控制模块上直流电机部分的原理图见下图。



图中 PWM 输入对应 ICETEK-LF2407-A 板上 P4 外扩插座第 26 引脚的 IOPE5 信号，DSP 将在此引脚上给出 PWM 信号用来控制直流电机的转速；图中的 DIR 输入对应 ICETEK-LF2407-A 板上 P4 外扩插座第 29 引脚的 IOPF4 信号，DSP 将在此引脚上给出高电平或低电平来控制直流电机的方向。从 DSP 输出的 PWM 信号和转向信号先经过 2 个与门和 1 个非门再与各个开关管的栅极相连。

控制原理

当电动机要求正转时，IOPE5 给出高电平信号，该信号分成 3 路：第 1 路接与门 Y1 的输入端，使与门 Y1 的输出由 PWM 决定，所以开关管 V1 栅极受 PWM 控制；第 2 路直接与开关管 V4 的栅极相连，使 V4 导通；第 3 路经非门 F1 连接到与门 Y2 的输入端，使与门 Y2 输出为 0，这样使开关管 V3 截止；从非门 F1 输出的另一路与开关管 V2 的栅极相连，其低电平信号也使 V2 截止。

同样，当电动机要求反转时，IOPE4 给出低电平信号，经过 2 个与门和 1 个非门组成的逻辑电路后，使开关管 V3 受 PWM 信号控制，V2 导通，V1、V4 全部截止。

四. 实验程序

程序中采用定时器中断产生固定频率的 PWM 波，100 次中断为一个周期，在每个中断中根据当前占空比判断应输出波形的高低电平。

主程序用轮询方式读入键盘输入，得到转速和方向控制命令。

在改变电机方向时为减少电压和电流的波动采用先减速再反转的控制顺序。

```
#include "2407c.h"

#include "scancode.h"

#define T46uS     0x0d40

ioport unsigned int port8000;

ioport unsigned int port8007;

void Delay(unsigned int nTime);

void interrupt none(void);

void interrupt gptime1(void);
```

```
void gp_init(void);
unsigned int uWork,nCount,uN,nCount1;
unsigned int cnt = 0;
unsigned int datacnt = 0, data[2]={10,30};
main()
{
    asm(" setc INTM");  //关中断
    port8000=0;
     port8000=0x80;
    port8000=2;
    port8000=1;        //使能直流电机
    port8007=0;           // 关闭东西方向的交通灯
    port8007=0x40;    // 关闭南北方向的交通灯
    uN=10; nCount=nCount1=0; //cKey=cOldKey=0;
    *WDCR=0x6f;
    *WDKEY=0x5555;
    *WDKEY=0xaaaa;     //关闭看门狗中断
    *SCSR1=0x81fe;     //打开所有外设，设置时钟频率为 40MHz
    uWork=(*MCRC);
    uWork&=0x0efdf;   /* 将 PWM11/IOPE5,TDIR2/IOPF4 设置成通用 I/O 口 */
    (*MCRC)=uWork;
    gp_init();
    *IMR=0x2;     //使能定时器中断
    *IFR=0xffff;  //清所有中断标志
    uWork=(*WSGR);     //（以下三句）设置 I/O 等待状态为 0
    uWork&=0x0fe3f;
    (*WSGR)=uWork;
    uWork=(*PFDATDIR);//（以下三句）将 direct 置为 0
    uWork|=0x1000;
```

```
    uWork&=0xffef;
    (*PFDATDIR)=uWork;
    asm(" clrc INTM"); //开中断
    Delay(128);
    *T1PR=T46uS; //保存结果周期=0xd40*25ns
    for(;;)
    {
        if ( nCount>50 )
        {
            nCount=0;
        }

    }
     port8000=0;
     port8000=0x80;
    port8000=0;
    exit(0);
}
void interrupt gptime1(void)
{
   uWork=(*PIVR);
   switch(uWork)
   {
        case 0x27:
        {
            (*EVAIFRA)=0x80;////////////////////here:)
            uWork=(*PEDATDIR);
            uWork|=0x2000;
            if ( nCount>uN )
```

```
                uWork|=0x20;
            else
                uWork&=0x0ffdf;
            (*PEDATDIR)=uWork;
        nCount++;
        nCount1++; nCount1%=100;
        cnt++;
        if(cnt>10000)
        {
            cnt = 0;
            datacnt ++;
            if (datacnt > 1)
                datacnt = 0;
            uN=data[datacnt];
        }
            break;
        }
    }
}
void gp_init(void)
{
    *EVAIMRA = 0x80;          //使能 T1PINT
    *EVAIFRA = 0xffff;        //清中断标志
    *GPTCONA = 0x0000;  //setting of period interrupt flag starts ADC
    *T1PR    = T46uS*9/5;  //保存结果周期=0xd40*200*9/5ns=1.22ms=820Hz
    *T1CNT   = 0;          //计数器从 0 开始计数
    *T1CON   = 0x1340;
}
void Delay(unsigned int nDelay)
```

```
{
    int i,j,k;
    for ( i=0;i<nDelay;i++ )
        for ( j=0;j<16;j++ )
            k++;
}
void interrupt none(void)
{}
```

五. 实验步骤

1. 实验准备

(1)连接设备

① 关闭计算机和实验箱电源。

② 检查 ICETEK-LF2407-A 板上跳线 JP6（MP/MC）的位置，应设置在“1－2”位置，即设置 DSP 工作在 MP 方式。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。

③ 关闭实验箱上三个开关。

④ 将小键盘接头插入显示/控制模块上相应插座 P8。

⑵开启设备

① 打开计算机电源。

② 打开实验箱全部电源开关，包括两个信号源及 ctr 控制模块的电源。

③ 注意：ICETEK-LF2407-A 板上指示灯 D1 和 D2 亮，ICETEK-CTR 板上 J2、J3 灯亮。

④ ICETEK-LF2407-A 板上指示灯 D1、D2 亮。

⑤ 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。

⑶设置 Code Composer Studio 为 Emulator 方式。

⑷启动 Code Composer Studio 2.0

2.打开工程并浏览程序,工程目录为 D:\ICETEK-LF2407-EDULab\Lab21-DCMotor

3. 编译并下载程序

4. 运行并观察程序运行结果

开始运行程序后，电机以较高的转速转动，经过大概 5s 后，电机转速变慢，再经过 5s 电机转速又加快，依次循环下去。

5. 停止运行

如果程序退出或中断时电机不停转动，可以将控制 ICETEK-CTR 模块的电源开关关闭再开启一次。

六．实验结果

通过实验可以发现，直流电机可以程控的改变转速和方向。

七．问题与思考

电动机是一个电磁干扰源。电动机的启停还会影响电网电压的波动，它周围的电器开关也会引发火花干扰。因此，除了采用必要的隔离、屏蔽和电路板合理布线等措施外，看门狗的功能就会显得格外重要。看门狗在工作时不断地监视程序运行的情况，一旦程序“跑飞”，会立刻使 DSP 复位。

实验十二 异步串口通信实验

一. 实验目的

1. 了解 TMS320LF2407A DSP 片内串行通信接口（SCI）的特点。
2. 学会设置 SCI 接口进行通信。
3. 了解 ICETEK-LF2407-A 板上对 SCI 接口的驱动部分设计。
4. 学习设计异步通信程序。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱（或 ICETEK 仿真器+ICETEK-LF2407-A 系统板+相关连线及电源）。

三. 实验原理

1. TMS320LF2407A DSP 串行通信接口模块

TMS320LF240x 器件包括串行通信接口 SCI 模块。SCI 模块支持 CPU 与其他使用标准格式的异步外设之间的数字通信。SCI 接收器和发送器是双缓冲的，每一个都有它自己单独的使能和中断标志位。两者都可以独立工作，或者在全双工的方式下同时工作。

2. ICETEK-LF2407-A 板异步串口设计

由于 DSP 内部包含了异步串行通信控制模块，所以在板上只需加上驱动电路部分即可。驱动电路主要完成将 SCI 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，但由于它是 5V 器件所以它同 DSP 间的信号线必须有电平转换，此板采用的是 74LS245。

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，SCI 的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：

  SCI 异步波特率=SYSCLK/ [(BRR+1)*8]

  其中，BRR=SYSCLK/(SCI 异步波特率*8)-1；
- BRR=0 时的串行通信接口异步波特率：

  SCI 异步波特率=SYSCLK/16

这里 BRR 等于波特率选择寄存器的 16 位值。

四. 实验程序

```
/* 2407A 板：将 JP11 和 JP14 短接到 2-3;
   使用直连的串口通信电缆;
   启动串口调试助手.exe;
   PC 机发送一个"."为结束标志*/
#include "2407c.h"

void wait(int nWait);

char cString[17]={"Hello PC!, Over|" }, cReceive, cBuffer[17], cAnswer[16]
= {"Oh, you say"};
int bReceive,nLen;

main()
{
  unsigned int uWork;
  int i,k;

  bReceive=0;
  asm(" setc INTM");
  asm(" clrc SXM");
  asm(" clrc OVM");
  *WDCR=0x6f;
  *WDKEY=0x5555;
  *WDKEY=0xaaaa;     /*关闭看门狗中断*/
  *SCSR1=0x81fe;     /* 打开所有外设，设置时钟频率为 40MHz */
  uWork=(*MCRA);
  uWork|=0x03;       /* use SCITXD,SCIRXD */
```

```
(*MCRA)=uWork;
(*SCICCR)=0x07;       /* 8 位字符，1 停止位，无校验*/
(*SCICTL1)=0x03;  /* 使能发送和接收 */
(*SCICTL2)=0x00;  /* 禁止接收和发送中断 */
(*SCIHBAUD)=0x02; /* 波特率=208H,40MHz */
(*SCILBAUD)=0x08; /* 208h=40*10^6/(9600*8)-1 */
(*SCICTL1)=0x23;  /* 使能发送和接收,复位 SCI */
while ( 1 )
{
    if ( bReceive==0 )
    {
        for ( i=0;i<16;i++ )
        {
            do
            {
                uWork=(*SCICTL2);
            } while ( uWork&0x0c0 != 0x0c0 );
            (*SCITXBUF)=cString[i];
            wait(1024);
        }
    }
    else
    {
        for ( i=0;i<10;i++ )
        {
            do
            {
                uWork=(*SCICTL2);
            } while ( uWork&0x0c0 != 0x0c0 );
```

```
    (*SCITXBUF)=cAnswer[i];
    wait(1024);
}
do
{
    uWork=(*SCICTL2);
} while ( uWork&0x0c0 != 0x0c0 );
(*SCITXBUF)='\"';
for ( i=0;i<nLen;i++ )
{
    do
    {
        uWork=(*SCICTL2);
    } while ( uWork&0x0c0 != 0x0c0 );
    (*SCITXBUF)=cBuffer[i];
    wait(1024);
}
do
{
    uWork=(*SCICTL2);
} while ( uWork&0x0c0 != 0x0c0 );
(*SCITXBUF)='\"';
wait(1024);
for ( i=9;i<16;i++ )
{
    do
    {
        uWork=(*SCICTL2);
    } while ( uWork&0x0c0 != 0x0c0 );
```

```
            (*SCITXBUF)=cString[i];
            wait(1024);
        }
    }
    k=0; bReceive=0;
    while ( 1 )
    {
        do
        {
            uWork=(*SCIRXST);
        } while ( (uWork&0x40)==0 );
        cReceive=(*SCIRXBUF);
        cBuffer[k]=cReceive;
        if ( cReceive=='.' )
        {
            cBuffer[k+1]='\0';
            nLen=k+1;
            bReceive=1;
            break;
        }
        k++; k%=16;
    }
  }
}

void wait(int nWait)
{
  int i,j,k=0;
  for ( i=0;i<nWait;i++ )
```

```
        for ( j=0;j<64;j++ )
            k++;
    }
```

五. 实验步骤

1. 实验准备

⑴.连接设备

① 关闭计算机和实验箱电源；

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；

③ 关闭实验箱上三个开关；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；

④ 用附带的串行通信电缆连接计算机 COM 端口和 ICETEK-LF2407-A 板上 P6 九针接头。

⑵.开启设备

① 打开计算机电源；

② 打开实验箱电源开关，打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；

③ 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。

⑶.设置 Code Composer Studio 为 Emulator 方式：

参见“Code Composer Studio 入门实验”之四.3。

⑷.启动 Code Composer Studio

2. 打开工程，浏览程序，工程目录为 D:\2407EDULab\Lab20-SCI

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 D:\2407EDULab\Lab20-SCI 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。

六. 实验结果

通过 DSP 传送到 PC 机上的信息，可以看出：SCI 正确工作。

七. 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。

实验十三 快速傅立叶变换（FFT）算法实验

一. 实验目的

1.掌握用窗函数法设计 FFT 快速傅里叶的原理和方法。

2.熟悉 FFT 快速傅里叶特性。

3.了解各种窗函数对快速傅里叶特性的影响。

二. 实验设备

计算机，ICETEK-LF2407-EDU 实验箱（或 ICETEK 仿真器+ICETEK-LF2407-A 系统板+相关连线及电源）。

三. 实验原理

1.FFT 快速傅里叶基础理论。

2.FFT 快速傅里叶确定方法。

四. 实验程序

```
#include "math.h"
#include "2407c.h"

float pi=3.1415926;
int r=4;
int N=16;
int k,i,j,bfsize,p,t;
float u_real,u_imag,v_real,v_imag;
float x1_real[16],x1_imag[16],x2_real[16],x2_imag[16];
float wk_real[16],wk_imag[16];
float y_real[16],y_imag[16],z_imag[16],z_real[16],x_real[16],x_imag[16];

void interrupt none(void)
{}
main()
{
    *WDCR=0x6f;
    *WDKEY=0x5555;
    *WDKEY=0xaaaa;    /* 关闭看门狗中断 */
    *SCSR1=0x81fe;    /* 打开所有外设，设置时钟频率为 40MHz */
   for(i=0;i<N/2;i++)
   {
      wk_real[i]=cos(-2*pi*i/N);
```

```
    wk_imag[i]=sin(-2*pi*i/N);
}
for(i=0;i<N;i++)
{
    x1_real[i]=exp(-i);
    x1_imag[i]=0;
}
for(i=0;i<N;i++)
{
    y_real[i]=x1_real[i];
    y_imag[i]=x1_imag[i];
    z_real[i]=x2_real[i];
    z_imag[i]=x2_imag[i];
}
for(k=0;k<r;k++)
{
    for(j=0;j<1<<k;j++)
    {
        bfsize=1<<(r-k);
        for(i=0;i<bfsize/2;i++)
        {
            p=j*bfsize;
            t=i*(1<<k);

            u_real=y_real[i+p]-y_real[i+p+bfsize/2];
            u_imag=y_imag[i+p]-y_imag[i+p+bfsize/2];

            v_real=u_real*wk_real[t]-u_imag*wk_imag[t];
            v_imag=u_real*wk_imag[t]+u_imag*wk_real[t];

            z_real[i+p]=y_real[i+p]+y_real[i+p+bfsize/2];
            z_imag[i+p]=y_imag[i+p]+y_imag[i+p+bfsize/2];

            z_real[i+p+bfsize/2]=v_real;
            z_imag[i+p+bfsize/2]=v_imag;
        }
    }
    for(i=0;i<N;i++)
    {
```

```
            x_real[i]=y_real[i];
            x_imag[i]=y_imag[i];
            y_real[i]=z_real[i];
            y_imag[i]=z_imag[i];
            z_real[i]=x_real[i];
            z_imag[i]=x_imag[i];
        }
    }
    for(j=0;j<N;j++)
    {
        p=0;
        for(i=0;i<r;i++)
        {
            if(j&(1<<i))
                p+=1<<(r-i-1);
        }
        x2_real[j]=y_real[p];
        x2_imag[j]=y_imag[p];
    }
    for(i=0;i<N;i++)
        x1_real[i]=sqrt(pow(x2_real[i],2)+pow(x2_imag[i],2));
    i = i;
    while (1)
     {}
}
```

五. 实验步骤

1. 实验准备

(1) 连接设备

① 关闭计算机和实验箱电源；

② 检查 ICETEK-LF2407-A 板上 JP6 的位置，应连接在 1-2 位置（靠近 DSP 芯片端），即设置 DSP 工作在 MP 方式；

③ 关闭实验箱上三个开关。

(2) 开启设备

① 打开计算机电源；

② 打开实验箱电源开关，打开 ICETEK-LF2407-A 板上电源开关，注意板上指示灯 DS1 灭、DS2 和 DS3 亮；

③ 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 Emulator 方式：

(4) 启动 Code Composer Studio

2. 打开工程，浏览程序，工程目录为 D:\2407EDULab\Lab18-FFT

3. 编译并下载程序

六. 实验结果

双击变量 x1_real，单击右健，点击“add to watch window” ，察看数组 x1_real[16]的值,在屏幕下方显示运行结果。如下所示：

x1_real[16]={1.58，1.48，1.28，1.08，0.93，0.84，0.78，0.74，0.73，0.74，
0.78，0.84，0.93，1.08，1.27，1.48}

七. 问题与思考

试选用不同点数的 fft 运算法则来运算。