**深 圳 大 学 实 验 报 告**

|  |  |  |
| --- | --- | --- |
| **课程名称:** | **DSP原理与应用技术** |  |
| **实验名称:** | **模拟信号采集实验** |  |
| **学 院:** | **机电与控制工程学院** |  |
| **专 业:** | **自动化** |  |
| **指导教师:** |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **姓名:** |  | **学号:** |  | **班级:** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **分组编号:** |  | **实验台号:** |  |  |

|  |  |  |
| --- | --- | --- |
| **实验时间:** | **2021年 月 日** |  |
| **提交时间:** | **2021年 月 日** |  |

**教务部制**

**实验目的与要求**

熟练掌握DSP ADC模块的功能配置和程序设计思路

练习面向DSP的C语言编程

尝试基于中断方法的程序设计思路（提高）

任务L1

将ADC配置成级联模式，每轮采样两个通道（第一个通道为A0，第二个通道为学号末位(6)对应数字通道）；

若第一通道模拟量大于1.5V，则二极管D8点亮，否则熄灭；第二通道控制D9的亮灭状态；

测试要求：调节滑动变阻器，通过观察窗口观察结果。

任务L2

采用与L1相同的ADC配置

将A0通道采集到的数字量换算回对应的模拟量，并且在数码管实时显示；另一通道若大于1.5V，则二极管D9点亮，否则熄灭

测试要求：调节滑动变阻器，观察数码管的显示变化

难点：1. 何时循环点亮4只数码管？何时读取ADC的最新读数？

2. 如何将0~4095转换成0.000V~3.000V电压并显示?

任务L3

采用中断法设计程序，实现上述功能

注1：参考中断管理的3层守卫流程进行初始化；参考定时器中断的程序设计框架设计主程序和中断函数

**实验原理**

**ADC的背景概述**

物理世界是模拟量的，计算机只能处理数字量，所以需要先ADC(A模拟量；D数字量；C转换)

物理信号首先转换至电压信号（模拟量）

ADC：将电压信号转换至数字量（非电流信号！）

ADC核心指标：精度及速度

精度-位数：12位（2^12=4096等分, 分辨率=1/4096）

速度-SPS：最快12.5百万次采样每秒

转换准备：信号需要调理成0~3V电压信号。

例. 汽车测速区间0~200km，通过信号调理：0km→0V; 200km→3V

转换过程：启动转换→ 开启采样窗口（=摄影中的曝光）→保持→开始转换→ 转换结束（告知、转存结果）

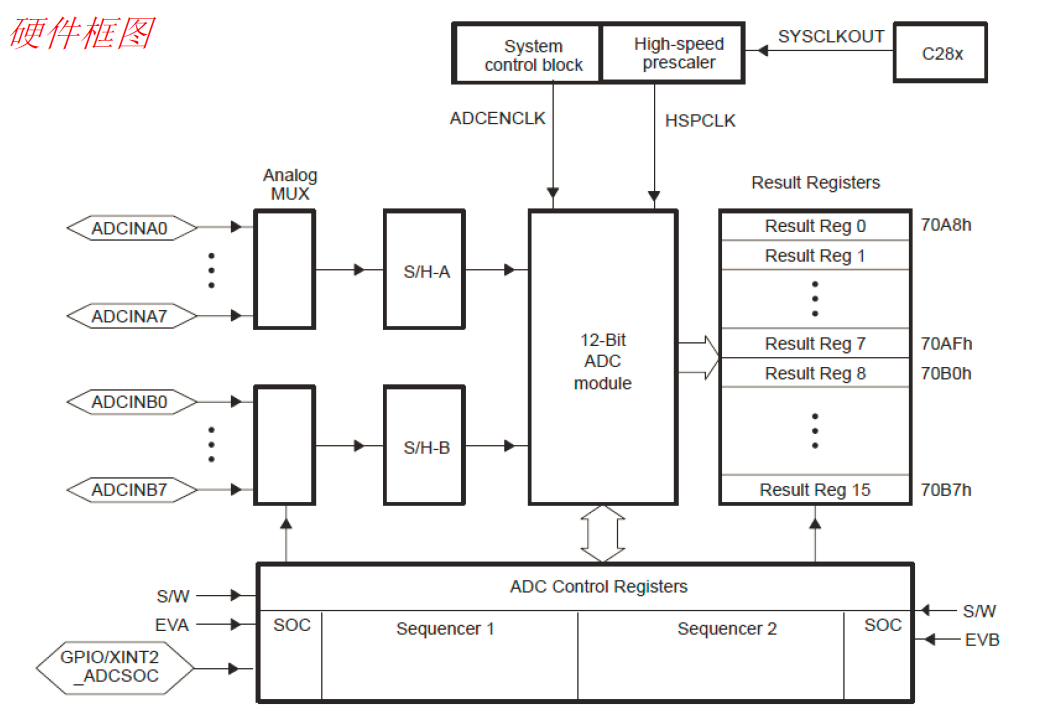
可以是一个通道的采样和转换，也可以是一个多个通道按约定次序采样和转换

**ADC的工作原理**

ADC共有16个输入通道引脚，分成为两组。

ADCINA0~ADCINA7，使用采样保持器S/H-A，对应于排序器SEQ1

ADCINB0~ADCINB7，使用采样保持器S/H-B，对应于排序器SEQ2



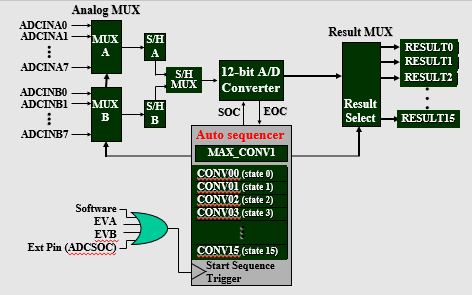
**自动转换排序器的工作原理**

自动排序器的作用是为需要转换的模拟输入通道安排转换的顺序，即确定先采哪个通道，后采哪个通道。ADC的排序器由2个8状态排序器SEQ1和SEQ2组成，它们也可以级联成1个16状态排序器SEQ。

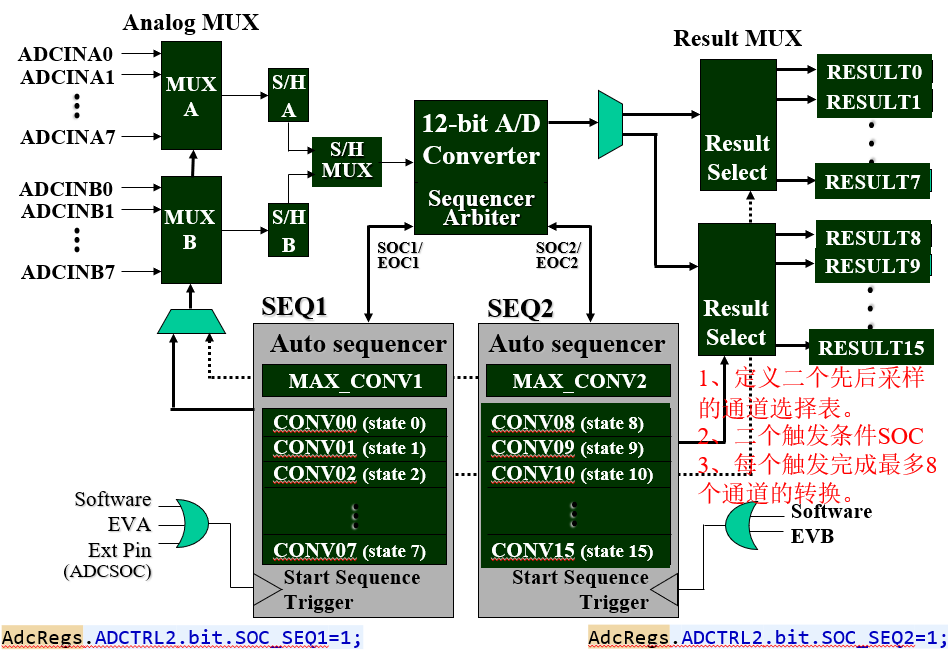
这里所说的“状态”是指排序器中能够完成A/D转换通道的个数。

排序器又可以分为级联排序器SEQ（级联构成16状态）模式和双排序器SEQ1/SEQ2（2个相互独立的8状态）模式。

**级联排序器（16状态）模式下自动排序ADC结构框图**



**双排序器（两个独立的8状态）模式下自动排序ADC结构框图**



**ADC单操作模式和级联操作模式比较**



**ADC的寄存器**

片上ADC模块有：

3个控制寄存器（ADCTRL1~3)

1个状态寄存器（ADCST）

4个输入通道选择排序控制寄存器（ADCCHSELSEQ1~4)

1个自动排序状态寄存器（ADCASEQSR）

1个最大转换通道寄存器（ADCMAXCONV）

16个结果寄存器（ADCRESULT0~15）。

ADC模块寄存器如表8-1所示。该表中的寄存器映射到外设帧PF2中，这个空间只允许16位访问，32位的访问会产生未定义的结果。

**ADC控制寄存器1：ADCTRL1**



位14 RESET: 模数转换模块软件复位位。为1，复位模块。

位13~12 SUSMOD1~SUSMOD0：仿真悬挂模式。

位11~8 ACQ\_PS3~ACQ\_PS0：采样时间窗宽度位。

位7 CPS：内核时钟预分频器 。设定对外设高速时钟(HSPCLK)的分频。

位6 CONT RUN：连续运行位。 0：启动/停止模式。 1：连续转换模式。

位5 SEQ OVRD：排序器超越模式位。 1：使能超越模式。

位4 SEQ CASC：级联排序器工作模式位。 0：双排序器工作模式 。l：级联模式 。

**ADC控制寄存器2：ADCTRL2**



位15 EVB SOC SEQ：级联排序器模式下EVB SOC使能位。1：EVB的信号启动级联的排序器SEQ。

位14 RST SEQ1：复位排序器1位。 1：立即将排序器复位到CONV00。

位13 SOC SEQ1：开始转换SOC触发SEQ1/级联16状态SEQ。

位11 INT ENA SEQ1：排序器SEQ1的中断使能位。1：使能 。

位10 INT MOD SEQ1：排序器SEQ1的中断模式控制位。0：每个SEQ1排序结束时置1。 1：每隔一个SEQ1排序结束时置1。

位8 EVA SOC SEQ1：EVA对SEQ1产生SOC信号的屏蔽位。1：使能EVA的触发信号源启动SEQ1/SEQ。

位7 EXT SOC SEQ1：外部信号启动SEQ1转换位。1：使能 。

位6 RST SEQ2：复位排序器2。l：立即复位SEQ2 。

位5 SOC SEQ2：启动SEQ2转换位 。

位3 INT ENA SEQ2：SEQ2的中断使能控制位。1：使能 。

位2 INT MOD SEQ2：SEQ2的中断模式控制位。 0：每个SEQ2排序结束时置1。 1：每隔一个SEQ2排序结束时置1。

位0 EVB SOC SEQ2：EVB对SEQ2产生SOC信号的屏蔽位。

**ADC控制寄存器3：ADCTRL3**

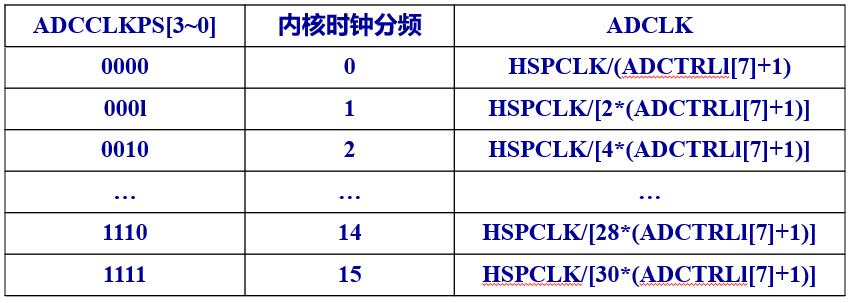


位7~6 ADCBGRFDN1、ADCBGRFDN0：模数转换内部带隙(Bandgap)和参考电压源电路的电源上电。这两位控制ADC内部参考电压源电路的上电与断电。

位5 ADCPWDN：模数转换模块掉电控制位。

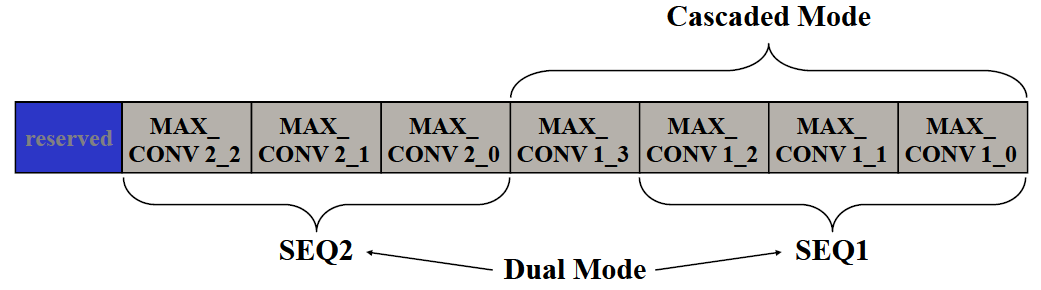
位0 SMODE\_SEL：采样模式选择位。0:顺序采样。1：同时采样。

**位4~1 ADCCLKPS[3~0]：内核时钟分频器**



ADCCLKPS[3~0]确定ADC的转换速度，数值越大转换越慢

**最大通道转换寄存器：ADCMAXCONV**



位6~0 MAX CONVn：定义一次自动转换最多可以转换的通道个数。一次转换的个数为MAX CONVn+1

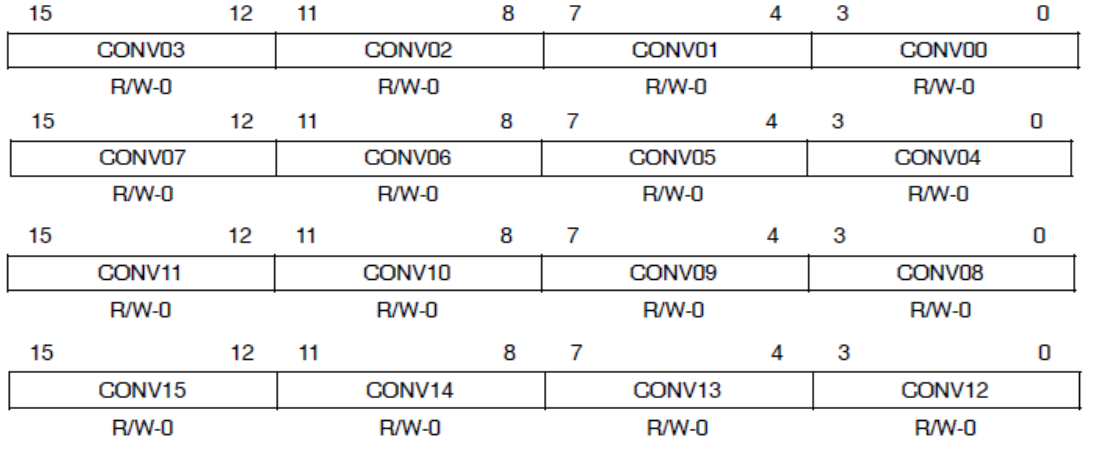
**自动排序状态寄存器：ADCASSR**



位11~8 SEQ CNTR[3~0]：排序计数器状态位。在转换排序开始时，SEQ CNTR[3~0]初始化为MAX CONV中的值。在一个自动转换排序的每一个转换之后，排序器的计数器减1。

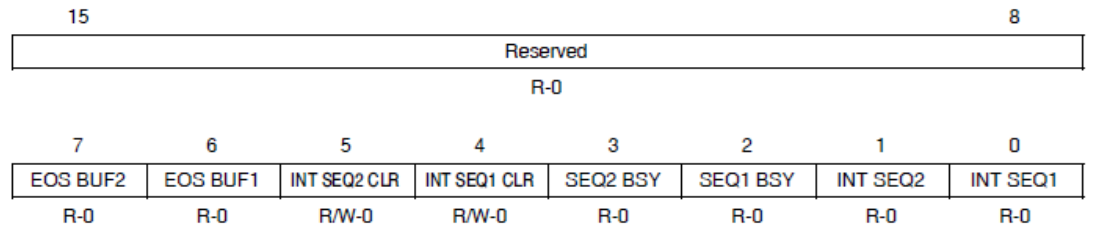
位6~0 是排序器SEQ2和SEQ1的指针。作为TI测试用。

**ADC输入通道排序寄存器:ADCCHSELSEQ1~4**





**ADC状态和标志寄存器：ADCST**



位7 EOS BUF2：SEQ2的排序缓冲器结束位。

位6 EOS BUF1：SEQ1的排序缓冲器结束位。

位5 INT SEQ2 CLR：SEQ2中断清零位。

位4 INT SEQ1 CLR：SEQ1中断清零位。

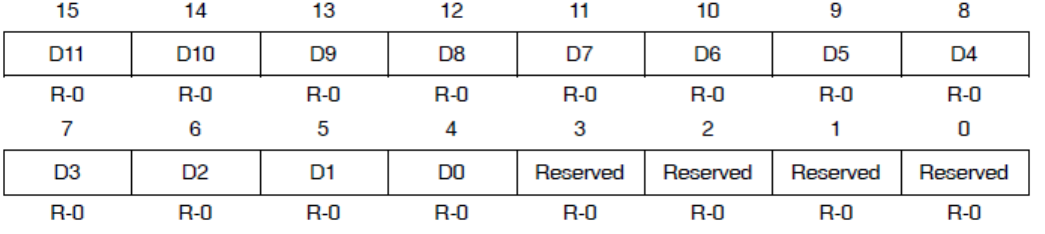
位3 SEQ2 BSY：SEQ2忙状态位。

位2 SEQ1 BSY：SEQ1忙状态位。

位1 INT SEQ2：SEQ2中断标志位 。

位0 INT SEQ1：SEQ1中断标志位 。

**ADC转换结果缓冲寄存器: ADCRESULT0~15**



寄存器ADCRESULTn中12位转换结果是左对齐的，即存放在16位寄存器的高12位。需要移位！

AD1 = AdcRegs.ADCRESULT0>>4;

**连续自动排序模式**

1. 如果CONT RUN位＝1，转换排序自动再次启动（即SEQ CNTRn重载MAX CONV1中的初始值，且SEQ1状态被置于CONV00）。在这种情况下，必须确保在下一次转换排序之前读取结果寄存器。在ADC模块向结果寄存器写入数据而用户却想从结果寄存器读取数据时，ADC的仲裁逻辑确保结果寄存器不会崩溃。
2. 如果CONT RUN位＝0，则排序器工作于启动/停止模式，排序器会停留在最后的状态（例如CONV06），并且SEQ CNTRn继续保持0值。因为每次SEQ CNTRn达到0时，中断标志会被置1。如果需要，用户可使用ADCTRL2寄存器的RST SEQn位，在中断服务程序中复位排序器，以便下一次转换启动时，SEQ CNTRn可以重载MAX CONV1的初始值，且SEQ1状态被设置为CONV00。这一特性在排序器的启动/停止操作中非常有用。

**实验过程**

基本任务

要求：

1. 将ADC配置成级联模式，每轮采样两个通道（第一个通道为A0，第二个通道为学号末位对应数字通道）；
2. 若第一通道模拟量大于1.5V，则二极管D8点亮，否则熄灭；第二通道控制D9的亮灭状态；
3. 测试要求：调节滑动变阻器，通过观察窗口观察结果。

代码分析

头文件

|  |
| --- |
| **代码：** |
| **说明：**  引入头文件 |

设置宏定义

|  |
| --- |
| **代码：** |
| **说明：**  GPIO端口宏定义 |

ADC启动参数

|  |
| --- |
| **代码：** |
| **说明：**  使用宏定义设置ADC时钟分频系数为3（#define ADC\_MODCLK 0x3），也就是设置ADC时钟频率为25MHz；使用宏定义设置ADC模块时钟参数为1（#define ADC\_CKPS 0x1），也就是设置ADC模块时钟为12.5MHz；使用宏定义设置采样窗口的长度为16个ADC 时钟（#define ADC\_SHCLK 0xf）；设置每次采样的个数参数为5（平均采样限值）； |

函数声明

|  |
| --- |
| **代码：** |
| **说明：**  声明所用的函数 |

全局变量

|  |
| --- |
| **代码：** |
| **说明：**  为了能保存读取到的两个通道的ADC数据量，这里定义两个全局变量数组，其中SampleTable0 用于保存第一通道传来的ADC数据量，SampleTable1用于保存第二通道传来的ADC数据量。SampleTable 用于存放采样数据，为了求采样的平均值，已消除采样过程的抖动 |

主函数初始化系统

|  |
| --- |
| **代码：** |
| **说明：**  InitPieCtrl() 函数。初始化PIE，将寄存器都置0 |

设置ADC时钟

|  |
| --- |
| **代码：** |
| **说明：**    设置分频系数为3（SysCtrlRegs.HISPCP.all = ADC\_MODCLK），也就是设置ADC时钟为25MHz，因为要操作底部寄存器。调用EALLOW后才能操作底部寄存器，调用EDIS禁止调用底部寄存器； |

初始化中断

|  |
| --- |
| **代码：** |
| **说明：**    DINT 是同一个汇编语句的宏定义，用于禁止CPU全局中断  InitPieCtrl() 用于初始化PIE中断  InitPieVectTable() 用于初始化中断向量表 |

初始化片内外设

|  |
| --- |
| **代码：** |
| **说明：**  需要调用InitAdc()初始化ADC，  调用Init\_LEDS\_Gpio初始化所用到的LED的GPIO；  设置本任务需要设置的ADC参数  AdcRegs.ADCTRL1.bit.ACQ\_PS = ADC\_SHCLK；  AdcRegs.ADCTRL3.bit.ADCCLKPS = ADC\_CKPS  设置ADC 模块时钟为12.5MHz；  调用AdcRegs.ADCTRL1.bit.SEQ\_CASC = 1  设置为题目要求的级联模式（为0时为双排序器模式）；  AdcRegs.ADCTRL3.bit.SMODE\_SEL = 0  设置为顺序采样模式；  AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0  设置A0为第一采样通道，  AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x3  设置A3为第三采样通道；  AdcRegs.ADCTRL1.bit.CONT\_RUN = 1  设置ADC模块为连续运行模式  AdcRegs.ADCMAXCONV.bit.MAX\_CONV1 = 0x1  设置最大数据存储器单元为2个； |

初始化LED

|  |
| --- |
| **代码：** |
| **说明：**  将控制LED灯的GPIO口清零 |

初始化采样表

|  |
| --- |
| **代码：** |
| **说明：**  将采样表清零 |

采样

|  |
| --- |
| **代码：** |
| **说明：**  while (AdcRegs.ADCST.bit.INT\_SEQ1== 0) {}等待SEQ1转换完后产生的中断；  接着调用AdcRegs.ADCST.bit.INT\_SEQ1\_CLR = 1清空中断标志位； |

采样数据转换

|  |
| --- |
| **代码：** |
| **说明：**  读取两个采样表中的数据，进行计算、转换。  进行四舍五入的处理 |

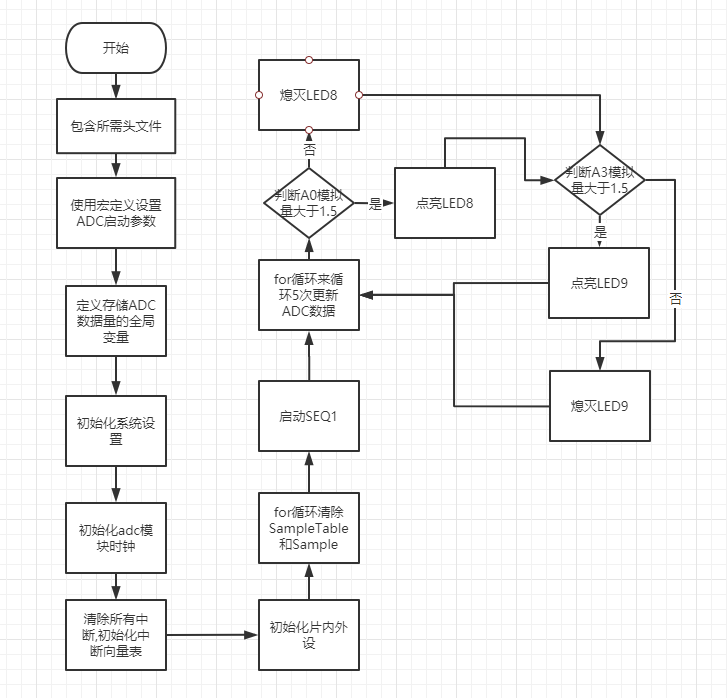
LED灯控制逻辑

|  |
| --- |
| **代码：** |
| **说明：**  判断读取到的电压是否大于1.5V，如果大于1.5V 则点亮LED灯，否则熄灭LED灯 |

数码管扫描显示

|  |
| --- |
| **代码：** |
| **说明：**  判断读取到的电压是否大于1.5V，如果大于1.5V 则点亮LED灯，否则熄灭LED灯 |

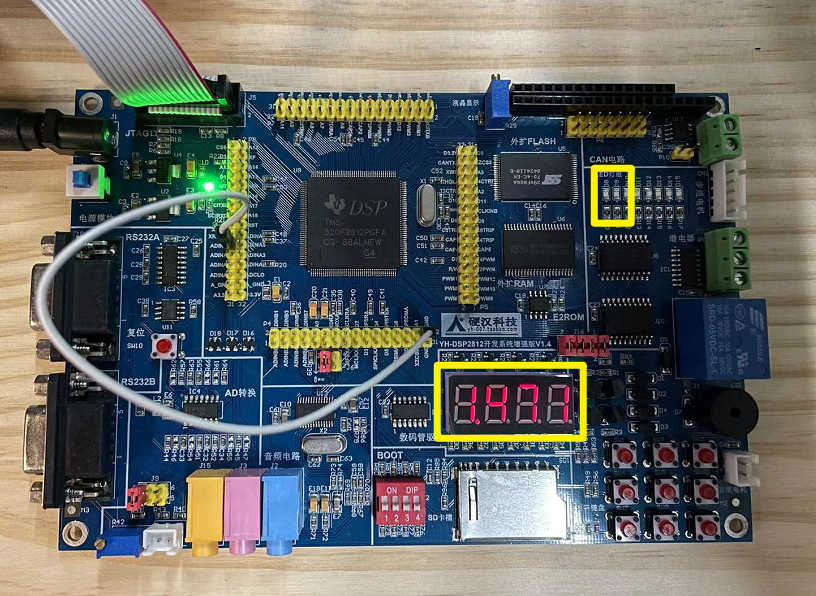
程序流程框图



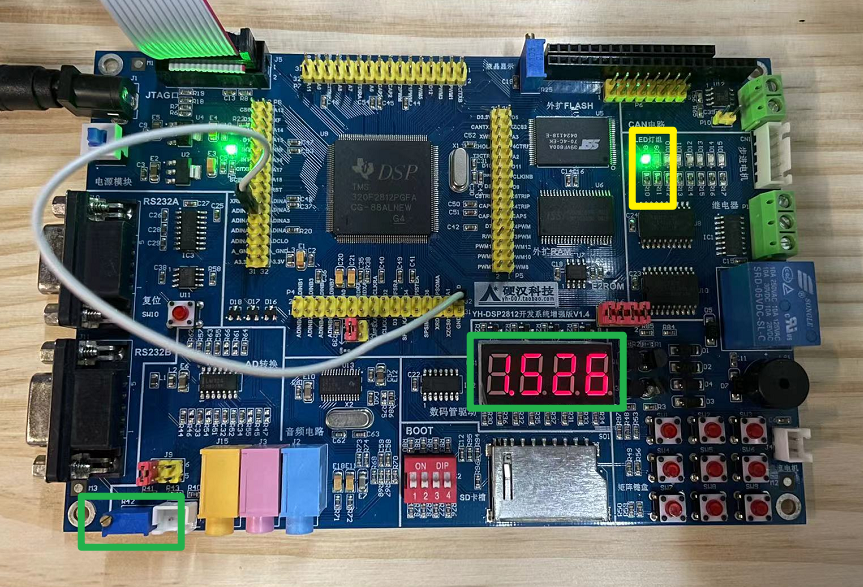
**实验结果：**

1. 第一个通道A0采集的电压显示在数码管上，为1.471V。

电压小于1.5V，LED灯D8不亮



1. 逆时针转动左下角的可变电阻，可以观察到数码管上的数字发生变化。当电压超过1.5V时，可以看到LED灯D8亮起。

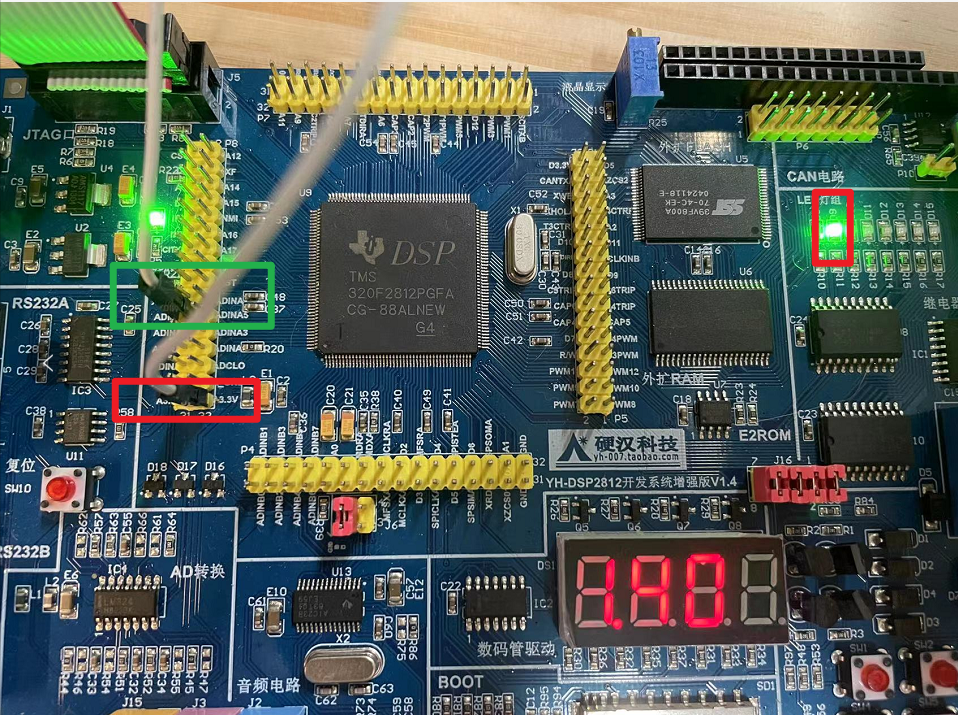


1. 通道2 (ADIN6) 检测电压。使用跳线连接ADIN6 和 GND ，可以看到LED灯 D9 不亮，

与预期相符合 （检测电压如果超过1.5V将会点亮LED灯 D9，否则D9不亮）



1. 通道2 (ADIN6) 检测电压。 使用跳线连接ADIN6 和 3.3V端口 ，可以看到LED灯 D9 点亮，与预期相符合。（检测电压如果超过1.5V将会点亮LED灯 D9，否则D9不亮）



代码附录

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* 文件名： 数码管显示程序/电压表采样实验

\* 描述: 电压表采样实验 ，D8 和数码管是对应着的，数码管过了1.5 D9就会亮，而D9对应着另一个通道

\* Author:icarusH

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 功能描述:

// 用跳线帽将J9插件的1,2引脚短接

// 通过ADC通道A0采集模拟电压值，然后通过SPI传输给数码管

// 通过数码管显示电压值

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "DSP281x\_Device.h" // DSP281x Headerfile Include File

#include "DSP281x\_Examples.h" // DSP281x Examples Include File

#define stuNum 0x06 // 学号末位，用作第二通道

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*宏定义数码管位选 IO 接口\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define SET\_BIT4 GpioDataRegs.GPBSET.bit.GPIOB8 = 1

//与外设板 8\_LEDS 端子的 IOB8 对应

#define RST\_BIT4 GpioDataRegs.GPBCLEAR.bit.GPIOB8 = 1

//与外设板 8\_LEDS 端子的 IOB8 对应

#define SET\_BIT3 GpioDataRegs.GPBSET.bit.GPIOB9 = 1

//与外设板 8\_LEDS 端子的 IOB9 对应

#define RST\_BIT3 GpioDataRegs.GPBCLEAR.bit.GPIOB9 = 1

//与外设板 8\_LEDS 端子的 IOB9 对应

#define SET\_BIT2 GpioDataRegs.GPBSET.bit.GPIOB10 = 1

//与外设板 8\_LEDS 端子的 IOB10 对应

#define RST\_BIT2 GpioDataRegs.GPBCLEAR.bit.GPIOB10 = 1

//与外设板 8\_LEDS 端子的 IOB10 对应

#define SET\_BIT1 GpioDataRegs.GPBSET.bit.GPIOB13 = 1

//与外设板 8\_LEDS 端子的 IOB13 对应

#define RST\_BIT1 GpioDataRegs.GPBCLEAR.bit.GPIOB13 = 1

//与外设板 8\_LEDS 端子的 IOB13 对应

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*函数申明\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//void delay(Uint32 t);

static void DisData\_Trans**(**Uint16 data**);**

static void Sellect\_Bit**(**Uint16 i**);**

static void Init\_LEDS\_Gpio**(**void**);**

static void spi\_xmit**(**Uint16 a**);**

static void spi\_fifo\_init**(**void**);**

static void spi\_init**(**void**);**

static void delay**(**Uint16 t**);**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*定义相关变量\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static unsigned char msg**[**10**]={**0xC0**,**0xf9**,**0xA4**,**0xB0**,**0x99**,**0x92**,**0x82**,**0xF8**,**0x80**,**0x90**};** //段码：0~9

static unsigned char DisData\_Bit**[**4**]** **=** **{**0**};** //存放拆分后的四位数字

static Uint16 DisData **=** 1234**;** //显示的数字

static Uint16 Loop **=** 0**;**

static Uint16 LedBuffer**[**2**];**

static Uint16 showdata**;**

// ADC启动参数

#define ADC\_MODCLK 0x3

// HSPCLK = SYSCLKOUT/2\*ADC\_MODCLK2 = 150/(2\*3) = 25MHz

#define ADC\_CKPS 0x0

// ADC 模块时钟 = HSPCLK/1 = 25MHz/(1) = 25MHz

#define ADC\_SHCLK 0x1

// 采样窗口时间 = 2 ADC 周期

#define AVG 1000

// 平均采样限值

#define ZOFFSET 0x00

#define BUF\_SIZE 1024

// 采样缓冲器尺寸

// 该例程使用的全局变量

//Uint16 SampleTable[BUF\_SIZE];

static Uint16 SampleTable0**[**BUF\_SIZE**];**

static Uint16 SampleTable1**[**BUF\_SIZE**];**

static Uint16**\*** SampleTable**[**2**]** **=** **{**SampleTable0**,** SampleTable1**};**

//循环扫描变量

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*数码管位选 IO 接口初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void Init\_LEDS\_Gpio**(**void**)**

**{**

EALLOW**;**

GpioMuxRegs**.**GPAMUX**.**all**=**0x0000**;**

GpioMuxRegs**.**GPADIR**.**all**=**0xFFFF**;** // GPIO PORTs as output

GpioMuxRegs**.**GPAQUAL**.**all**=**0x0000**;** // Set GPIO input qualifier values

GpioDataRegs**.**GPBSET**.**bit**.**GPIOB8 **=** 1**;**

// 输出高电平

GpioMuxRegs**.**GPBMUX**.**bit**.**CAP4Q1\_GPIOB8 **=** 0**;**

// GPIOB8 = GPIO

GpioMuxRegs**.**GPBDIR**.**bit**.**GPIOB8 **=** 1**;**

// GPIOB8 = output

GpioDataRegs**.**GPBSET**.**bit**.**GPIOB9 **=** 1**;**

// 输出高电平

GpioMuxRegs**.**GPBMUX**.**bit**.**CAP5Q2\_GPIOB9 **=** 0**;**

// GPIOB9 = GPIO

GpioMuxRegs**.**GPBDIR**.**bit**.**GPIOB9 **=** 1**;**

// GPIOB9 = output

GpioDataRegs**.**GPBSET**.**bit**.**GPIOB10 **=** 1**;**

// 输出高电平

GpioMuxRegs**.**GPBMUX**.**bit**.**CAP6QI2\_GPIOB10 **=** 0**;**

// GPIOB10 = GPIO

GpioMuxRegs**.**GPBDIR**.**bit**.**GPIOB10 **=** 1**;**

// GPIOB10 = output

GpioDataRegs**.**GPBSET**.**bit**.**GPIOB13 **=** 1**;**

// 输出高电平

GpioMuxRegs**.**GPBMUX**.**bit**.**C4TRIP\_GPIOB13 **=** 0**;**

// GPIOB13 = GPIO

GpioMuxRegs**.**GPBDIR**.**bit**.**GPIOB13 **=** 1**;**

// GPIOB13 = output

EDIS**;**

RST\_BIT1**;**//关闭数码管显示

RST\_BIT2**;**

RST\_BIT3**;**

RST\_BIT4**;**

**}**

static void Gpio\_select**(**void**)**

**{**

Uint16 var1**;**

Uint16 var2**;**

Uint16 var3**;**

var1**=** 0x0000**;** // sets GPIO Muxs as I/Os

var2**=** 0xFFFF**;** // sets GPIO DIR as outputs

var3**=** 0x0000**;** // sets the Input qualifier values

EALLOW**;**

GpioMuxRegs**.**GPAMUX**.**all**=**var1**;**

// GpioMuxRegs.GPBMUX.all=var1;

GpioMuxRegs**.**GPDMUX**.**all**=**var1**;**

GpioMuxRegs**.**GPFMUX**.**all**=**var1**;**

GpioMuxRegs**.**GPEMUX**.**all**=**var1**;**

GpioMuxRegs**.**GPGMUX**.**all**=**var1**;**

GpioMuxRegs**.**GPADIR**.**all**=**var2**;**

// GPIO PORTs as output

// GpioMuxRegs.GPBDIR.all=0xFFDF;

// GPIO的B组全部定义为输出，除GPIOB5定义为输入

GpioMuxRegs**.**GPDDIR**.**all**=**var2**;**

GpioMuxRegs**.**GPEDIR**.**all**=**var2**;**

GpioMuxRegs**.**GPFDIR**.**all**=**var2**;**

GpioMuxRegs**.**GPGDIR**.**all**=**var2**;**

GpioMuxRegs**.**GPAQUAL**.**all**=**var3**;**

// Set GPIO input qualifier values

// GpioMuxRegs.GPBQUAL.all=var3;

GpioMuxRegs**.**GPDQUAL**.**all**=**var3**;**

GpioMuxRegs**.**GPEQUAL**.**all**=**var3**;**

EDIS**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*数码管位选函数（从低位到高位扫描）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void Sellect\_Bit**(**Uint16 i**)**

**{**

**switch(**i**)**

**{**

**case** 0**:**

RST\_BIT4**;** //关断数码管第四位

SET\_BIT1**;** //选通数码管第一位

**break;**

**case** 1**:**

RST\_BIT1**;** //关断数码管第一位

SET\_BIT2**;** //选通数码管第二位

**break;**

**case** 2**:**

RST\_BIT2**;** //关断数码管第二位

SET\_BIT3**;** //选通数码管第三位

**break;**

**case** 3**:**

RST\_BIT3**;** //关断数码管第三位

SET\_BIT4**;** //选通数码管第四位

**break;**

**default:**

**break;**

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\* 拆分要显示的四位数保存到数组DisData\_Trans\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void DisData\_Trans**(**Uint16 data**)**

**{**

DisData\_Bit**[**3**]** **=** data **/** 1000**;** //千位数

DisData\_Bit**[**2**]** **=** data **%** 1000 **/** 100 **;** //百位数

DisData\_Bit**[**1**]** **=** data **%** 100 **/** 10**;** //十位数

DisData\_Bit**[**0**]** **=** data **%** 10**;** //个位数

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*延时函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void delay**(**Uint16 t**)**

**{**

Uint16 i **=** 0**;**

**for** **(**i **=** 0**;** i **<** t**;** i**++);**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Spi初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void spi\_init**()**

**{**

SpiaRegs**.**SPICCR**.**all **=**0x004F**;** // Reset on, rising edge, 16-bit char bits

//0x000F对应Rising Edge，0x004F对应Falling Edge

SpiaRegs**.**SPICTL**.**all **=**0x0006**;** // Enable master mode, normal phase,

// enable talk, and SPI int disabled.

SpiaRegs**.**SPIBRR **=**0x007F**;**

SpiaRegs**.**SPICCR**.**all **=**0x00DF**;** // Relinquish SPI from Reset

SpiaRegs**.**SPIPRI**.**bit**.**FREE **=** 1**;** // Set so breakpoints don't disturb xmission

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Spi模块FIFO设置\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void spi\_fifo\_init**()**

**{**

// 初始化 SPI FIFO 寄存器

SpiaRegs**.**SPIFFTX**.**all**=**0xE040**;**

SpiaRegs**.**SPIFFRX**.**all**=**0x204f**;**

SpiaRegs**.**SPIFFCT**.**all**=**0x0**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Spi发送\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void spi\_xmit**(**Uint16 a**)**

**{**

SpiaRegs**.**SPITXBUF**=**a**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//延迟函数

static void delay\_loop**()**

**{**

long i**;**

**for** **(**i **=** 0**;** i **<** 45000**;** i**++)** **{}**

**}**

void demo8**(**void**)**

**{**

Uint16 i**;**

Uint16 j**;**

Uint16 k**;**

Uint32 Sum**=**0**;**

Uint32 Vin**;**

// 步骤 1. 初始化系统控制:

// 设置PLL, WatchDog, 使能外设时钟

// 下面这个函数可以从DSP281x\_SysCtrl.c文件中找到.

InitSysCtrl**();**

// 步骤 2. 初始化通用输入输出多路复用器GPIO:

// 这个函数在DSP281x\_Gpio.c源文件中被定义了

// InitGpio(); // 本例直接跳过该步骤

// 仅设置相应GPIO为为SPI功能引脚

// 本例特定的时钟设置

EALLOW**;**

SysCtrlRegs**.**HISPCP**.**all **=** ADC\_MODCLK**;**

// HSPCLK = SYSCLKOUT/ADC\_MODCLK

EDIS**;**

//初始化SPI

EALLOW**;**

GpioMuxRegs**.**GPFMUX**.**all**=**0x000F**;**

// 选择GPIO为SPI引脚

// 端口F MUX - x000 0000 0000 1111

EDIS**;**

// Gpio\_select();

Init\_LEDS\_Gpio**();**

// 步骤 3. 清除所有中断,初始化中断向量表:

// 禁止CPU全局中断

DINT**;**

// 初始化PIE控制寄存器到他们的默认状态.

// 这个默认状态就是禁止PIE中断及清除所有PIE中断标志

// 这个函数放在DSP281x\_PieCtrl.c源文件里

InitPieCtrl**();**

// 禁止CPU中断，清除CPU中断标志位

IER **=** 0x0000**;**

IFR **=** 0x0000**;**

//初始化PIE中断向量表，并使其指向中断服务子程序（ISR）

// 这些中断服务子程序被放在了DSP281x\_DefaultIsr.c源文件中

// 这个函数放在了DSP281x\_PieVect.c源文件里面.

InitPieVectTable**();**

// 步骤 4.初始化片内外设:

InitAdc**();** // For this example, init the ADC

spi\_fifo\_init**();** // 初始化Spi FIFO

spi\_init**();** // 初始化 SPI

// 本例需要设置ADC参数

AdcRegs**.**ADCTRL1**.**bit**.**ACQ\_PS **=** ADC\_SHCLK**;**

AdcRegs**.**ADCTRL3**.**bit**.**ADCCLKPS **=** ADC\_CKPS**;**

// ADC 模块时钟 = HSPCLK/1 = 25MHz/(1) = 25MHz

AdcRegs**.**ADCTRL1**.**bit**.**SEQ\_CASC **=** 1**;**

// 1 级联模式

AdcRegs**.**ADCCHSELSEQ1**.**bit**.**CONV00 **=** 0x0**;**

//ADC通道 按顺序 先 采样通道ADCIN0

AdcRegs**.**ADCCHSELSEQ1**.**bit**.**CONV01 **=** stuNum**;**

//ADC通道 按顺序 接着 采样通道ADCINx, 这里x指的是学号末尾数

AdcRegs**.**ADCTRL1**.**bit**.**CONT\_RUN **=** 1**;**

// 设置为连续运行

AdcRegs**.**ADCMAXCONV**.**all **=** 0x0001**;**

// 最多采样 AdcRegs.ADCMAXCONV.all+1 个

// Step 5.用户指定代码，使能中断:

GpioDataRegs**.**GPADAT**.**all **=**0x0000**;**

//GPIOA0-A7输出清0，使LED1灯灭

// GpioDataRegs.GPBDAT.all =0xFFFF;

// GpioDataRegs.GPBDAT.all &=0xFFFE;//GPIOB0输出清0，其他置高

// 采样表清0

**for** **(**i**=**0**;** i**<**BUF\_SIZE**;** i**++)**

**{**

SampleTable0**[**i**]** **=** 0**;**

SampleTable1**[**i**]** **=** 0**;**

**}**

//关数码管；

spi\_xmit**(**0xFFFF**);**

//延迟

delay\_loop**();**

// 软件启动SEQ1,SEQ2

AdcRegs**.**ADCTRL2**.**all **=** 0x2020**;**

// 取ADC数据并写入采样数据表

**for(;;)**

**{**

**for** **(**i**=**0**;** i**<**AVG**;** i**++)**

**{**

// while (AdcRegs.ADCST.bit.INT\_SEQ1== 0) {} // 等待中断

// AdcRegs.ADCST.bit.INT\_SEQ1\_CLR = 1;

// SampleTable[i] =((AdcRegs.ADCRESULT0>>4) );

**while** **(**AdcRegs**.**ADCST**.**bit**.**INT\_SEQ1**==** 0 **)** **{}** // 等待中断

AdcRegs**.**ADCST**.**bit**.**INT\_SEQ1\_CLR **=** 1**;**

SampleTable0**[**i**]** **=((**AdcRegs**.**ADCRESULT0**>>**4**)** **);** // AdcRegs.ADCCHSELSEQ1.bit.CONV00的结果

SampleTable1**[**i**]** **=((**AdcRegs**.**ADCRESULT1**>>**4**)** **);** // AdcRegs.ADCCHSELSEQ1.bit.CONV01的结果

**}**

**for** **(**i**=**0**;**i**<**2**;++**i**){**

Sum **=** 0**;**

**for** **(**j**=**0**;**j**<**AVG**;**j**++)**

**{**

Sum**+=**SampleTable**[**i**][**j**];**

**}**

Sum**=**Sum**/**AVG**;**

//输入电压和AD值之间的关系Vin/Sum=3/4096；

Vin**=**Sum**\***3**\***10000**/**4096**;**

//将输入电压放大100倍，以便于第2位有效小数的四舍五入计算；

**if(**Vin**%**10**>=**5**)** //最后一位整数>=5时，要五入；

showdata**=**Vin**/**10**+**1**;**

**else**

showdata**=**Vin**/**10**;**//要四舍；

**if** **(**i **==** 1**){**

**if** **(** showdata **>** 1500**){**

GpioDataRegs**.**GPADAT**.**bit**.**GPIOA1 **=**1**;** // 点亮D9

**}else{**

GpioDataRegs**.**GPADAT**.**bit**.**GPIOA1 **=**0**;** //GPIOA0-A7输出清0，使LED1灯灭

**}**

**}**

**if** **(**i**==**0**)**

**{**

**if** **(** showdata **>** 1500**){**

GpioDataRegs**.**GPADAT**.**bit**.**GPIOA0 **=**1**;**

// 点亮D8

**}else{**

GpioDataRegs**.**GPADAT**.**bit**.**GPIOA0 **=**0**;**

//GPIOA0-A7输出清0，使LED1灯灭

**}**

**for(**k**=**0**;**k**<**100**;**k**++)**

**{**

DisData\_Trans**(**showdata**);** //拆分四位数

**for(**Loop**=**0**;**Loop**<**4**;**Loop**++)** //分别显示四位

**{**

Sellect\_Bit**(**Loop**);** //选择要扫描的数码管位

**if(**Loop**==**3**)**

spi\_xmit**(**msg**[**DisData\_Bit**[**Loop**]]+**0x80**);**

**else**

spi\_xmit**(**msg**[**DisData\_Bit**[**Loop**]]);** //串行输出要显示的数字

delay**(**1500**);** //延时配合人眼反应时间

**}**

**}**

**}**

**}**

**}**

**}**

//===================================================================

// No more.

//===================================================================

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 文件名： demo9

\* 描述: 电压表采样实验 中断法

\* Author:team0ne

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 功能描述:

// 用跳线帽将J9插件的1,2引脚短接

// 通过ADC通道A0采集模拟电压值，然后通过SPI传输给数码管

// 通过数码管显示电压值

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "DSP281x\_Device.h" // DSP281x Headerfile Include File

#include "DSP281x\_Examples.h" // DSP281x Examples Include File

#include <stdio.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*宏定义数码管位选 IO 接口\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define SET\_BIT4 GpioDataRegs.GPBSET.bit.GPIOB8 = 1

//与外设板 8\_LEDS 端子的 IOB8 对应

#define RST\_BIT4 GpioDataRegs.GPBCLEAR.bit.GPIOB8 = 1

//与外设板 8\_LEDS 端子的 IOB8 对应

#define SET\_BIT3 GpioDataRegs.GPBSET.bit.GPIOB9 = 1

//与外设板 8\_LEDS 端子的 IOB9 对应

#define RST\_BIT3 GpioDataRegs.GPBCLEAR.bit.GPIOB9 = 1

//与外设板 8\_LEDS 端子的 IOB9 对应

#define SET\_BIT2 GpioDataRegs.GPBSET.bit.GPIOB10 = 1

//与外设板 8\_LEDS 端子的 IOB10 对应

#define RST\_BIT2 GpioDataRegs.GPBCLEAR.bit.GPIOB10 = 1

//与外设板 8\_LEDS 端子的 IOB10 对应

#define SET\_BIT1 GpioDataRegs.GPBSET.bit.GPIOB13 = 1

//与外设板 8\_LEDS 端子的 IOB13 对应

#define RST\_BIT1 GpioDataRegs.GPBCLEAR.bit.GPIOB13 = 1

//与外设板 8\_LEDS 端子的 IOB13 对应

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void demo9**(**void**);**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*函数申明\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void DisData\_Trans**(**Uint16 data**);**

static void Sellect\_Bit**(**Uint16 i**);**

static void Init\_LEDS\_Gpio**(**void**);**

static void Init\_Light\_Gpio**(**void**);**

static void spi\_xmit**(**Uint16 a**);**

static void spi\_fifo\_init**(**void**);**

static void spi\_init**(**void**);**

static int getVin**(**Uint32 Res**);**

static void showLEDs**(**int data**);**

static void selectLights**(**void**);**

static \_\_interrupt void adc\_int**(**void**);**

static \_\_interrupt void cpu\_timer0\_isr**(**void**);**

//static void delay(Uint32 t);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*定义相关变量\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static unsigned char msg**[**10**]={**0xC0**,**0xf9**,**0xA4**,**0xB0**,**0x99**,**0x92**,**0x82**,**0xF8**,**0x80**,**0x90**};** //段码：0~9

static unsigned char DisData\_Bit**[**4**]** **=** **{**0**};** //存放拆分后的四位数字

static Uint16 DisData **=** 1234**;** //显示的数字

static Uint16 Loop **=** 0**;**

static volatile int RES1**=**0**;**

static volatile int RES2**=**0**;**

// ADC启动参数

#define ADC\_MODCLK 0x3 // HSPCLK = SYSCLKOUT/2\*ADC\_MODCLK2 = 150/(2\*3) = 25MHz

#define ADC\_CKPS 0x0F // ADC 模块时钟 = HSPCLK/1 = 25MHz/(1) = 25MHz

#define ADC\_SHCLK 0x1 // 采样窗口时间 = 2 ADC 周期

#define AVG 1000 // 平均采样限值

#define ZOFFSET 0x00

#define BUF\_SIZE 1024 // 采样缓冲器尺寸

// 该例程使用的全局变量

//static Uint16 SampleTable[BUF\_SIZE];

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void Init\_Light\_Gpio**(**void**){**

EALLOW**;**

GpioMuxRegs**.**GPAMUX**.**all**=**0x0000**;**

GpioMuxRegs**.**GPADIR**.**all**=**0xFFFF**;**

GpioMuxRegs**.**GPAQUAL**.**all**=**0x0000**;**

EDIS**;**

GpioDataRegs**.**GPADAT**.**all **=**0x0000**;**//GPIOA0-A7输出清0，使LED1灯灭

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*数码管位选 IO 接口初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void Init\_LEDS\_Gpio**(**void**)**

**{**

EALLOW**;**

GpioDataRegs**.**GPBSET**.**bit**.**GPIOB8 **=** 1**;**

// 输出高电平

GpioMuxRegs**.**GPBMUX**.**bit**.**CAP4Q1\_GPIOB8 **=** 0**;**

// GPIOB8 = GPIO

GpioMuxRegs**.**GPBDIR**.**bit**.**GPIOB8 **=** 1**;**

// GPIOB8 = output

GpioDataRegs**.**GPBSET**.**bit**.**GPIOB9 **=** 1**;**

// 输出高电平

GpioMuxRegs**.**GPBMUX**.**bit**.**CAP5Q2\_GPIOB9 **=** 0**;**

// GPIOB9 = GPIO

GpioMuxRegs**.**GPBDIR**.**bit**.**GPIOB9 **=** 1**;**

// GPIOB9 = output

GpioDataRegs**.**GPBSET**.**bit**.**GPIOB10 **=** 1**;**

// 输出高电平

GpioMuxRegs**.**GPBMUX**.**bit**.**CAP6QI2\_GPIOB10 **=** 0**;**

// GPIOB10 = GPIO

GpioMuxRegs**.**GPBDIR**.**bit**.**GPIOB10 **=** 1**;**

// GPIOB10 = output

GpioDataRegs**.**GPBSET**.**bit**.**GPIOB13 **=** 1**;**

// 输出高电平

GpioMuxRegs**.**GPBMUX**.**bit**.**C4TRIP\_GPIOB13 **=** 0**;**

// GPIOB13 = GPIO

GpioMuxRegs**.**GPBDIR**.**bit**.**GPIOB13 **=** 1**;**

// GPIOB13 = output

EDIS**;**

RST\_BIT1**;**//关闭数码管显示

RST\_BIT2**;**

RST\_BIT3**;**

RST\_BIT4**;**

**}**

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//

//

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*数码管位选函数（从低位到高位扫描）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void Sellect\_Bit**(**Uint16 i**)**

**{**

**switch(**i**)**

**{**

**case** 0**:**

RST\_BIT4**;** //关断数码管第四位

SET\_BIT1**;** //选通数码管第一位

**break;**

**case** 1**:**

RST\_BIT1**;** //关断数码管第一位

SET\_BIT2**;** //选通数码管第二位

**break;**

**case** 2**:**

RST\_BIT2**;** //关断数码管第二位

SET\_BIT3**;** //选通数码管第三位

**break;**

**case** 3**:**

RST\_BIT3**;** //关断数码管第三位

SET\_BIT4**;** //选通数码管第四位

**break;**

**default:**

**break;**

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\* 拆分要显示的四位数保存到数组DisData\_Trans\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void DisData\_Trans**(**Uint16 data**)**

**{**

DisData\_Bit**[**3**]** **=** data **/** 1000**;** //千位数

DisData\_Bit**[**2**]** **=** data **%** 1000 **/** 100 **;** //百位数

DisData\_Bit**[**1**]** **=** data **%** 100 **/** 10**;** //十位数

DisData\_Bit**[**0**]** **=** data **%** 10**;** //个位数

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*延时函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void delay1**(**Uint32 t**)**

**{**

Uint32 i **=** 0**;**

**for** **(**i **=** 0**;** i **<** t**;** i**++);**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Spi初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void spi\_init**()**

**{**

SpiaRegs**.**SPICCR**.**all **=**0x004F**;**

// Reset on, rising edge, 16-bit char bits

//0x000F对应Rising Edge，0x004F对应Falling Edge

SpiaRegs**.**SPICTL**.**all **=**0x0006**;**

// Enable master mode, normal phase,

// enable talk, and SPI int disabled.

SpiaRegs**.**SPIBRR **=**0x007F**;**

SpiaRegs**.**SPICCR**.**all **=**0x00DF**;**

// Relinquish SPI from Reset

SpiaRegs**.**SPIPRI**.**bit**.**FREE **=** 1**;**

// Set so breakpoints don't disturb xmission

**}**

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Spi模块FIFO设置\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void spi\_fifo\_init**()**

**{**

// 初始化 SPI FIFO 寄存器

SpiaRegs**.**SPIFFTX**.**all**=**0xE040**;**

SpiaRegs**.**SPIFFRX**.**all**=**0x204f**;**

SpiaRegs**.**SPIFFCT**.**all**=**0x0**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Spi发送\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

static void spi\_xmit**(**Uint16 a**)**

**{**

SpiaRegs**.**SPITXBUF**=**a**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//延迟函数

static void delay\_loop8**()**

**{**

long i**;**

**for** **(**i **=** 0**;** i **<** 4500**;** i**++)** **{}**

**}**

static int getVin**(**Uint32 Res**){**

// printf("Res:%d\n", Res);

int Vin**;**

int showdata **=** 0**;**

// for (i=0;i<AVG;i++)

// {

// Sum+=SampleTable[i];

// Sum=Sum/2;

// }

//输入电压和AD值之间的关系Vin/Sum=3/4096；

Vin**=**Res**\***3**\***10000**/**4096**;** //将输入电压放大100倍，以便于第2位有效小数的四舍五入计算；

// printf("Vin\*10:%d\n", Vin);

**if(**Vin**%**10**>=**5**)**//最后一位整数>=5时，要五入；

showdata**=**Vin**/**10**+**1**;**

**else**

showdata**=**Vin**/**10**;**//要四舍；

// printf("getVin:%d\n", showdata);

**return** showdata**;**

// AdcRegs.ADCTRL2.bit.SOC\_SEQ1 = 1;

**}**

static void showLEDs**(**int data**){**

int i**;**

DisData\_Trans**(**data**);**

**for(**i**=**0**;**i**<**16**;**i**++)**

**{**

**for(**Loop**=**0**;**Loop**<**4**;**Loop**++)**

//分别显示四位

**{**

Sellect\_Bit**(**Loop**);**

//选择要扫描的数码管位

**if(**Loop**==**3**)**

spi\_xmit**(**msg**[**DisData\_Bit**[**Loop**]]+**0x80**);**

**else**

spi\_xmit**(**msg**[**DisData\_Bit**[**Loop**]]);**

//串行输出要显示的数字

delay1**(**1000**);**

//延时配合人眼反应时间

**}**

**}**

**}**

static void selectLights**(**void**){**

**if(**RES1**>**1500**){**

printf**(**"(RES1):%d\n"**,**RES1**);**

printf**(**"(RES2):%d\n"**,**RES2**);**

printf**(**"LightA8:up\n"**);**

GpioDataRegs**.**GPASET**.**bit**.**GPIOA0 **=** 1**;**

**}else{**

printf**(**"(RES1):%d\n"**,**RES1**);**

printf**(**"(RES2):%d\n"**,**RES2**);**

printf**(**"LightA8:down\n"**);**

GpioDataRegs**.**GPACLEAR**.**bit**.**GPIOA0 **=** 1**;**

**}**

**if(**RES2**>**1500**){**

printf**(**"(RES1):%d\n"**,**RES1**);**

printf**(**"(RES2):%d\n"**,**RES2**);**

printf**(**"LightA9:up\n"**);**

GpioDataRegs**.**GPASET**.**bit**.**GPIOA1 **=** 1**;**

**}else{**

printf**(**"(RES1):%d\n"**,**RES1**);**

printf**(**"(RES2):%d\n"**,**RES2**);**

printf**(**"LightA9:down\n"**);**

GpioDataRegs**.**GPACLEAR**.**bit**.**GPIOA1 **=** 1**;**

**}**

**}**

static \_\_interrupt void adc\_int**(**void**){**

printf**(**"adc\_int\n"**);**

printf**(**"(AdcRegs.ADCRESULT0>>4):%d\n"**,(**AdcRegs**.**ADCRESULT0**>>**4**));**

printf**(**"(AdcRegs.ADCRESULT1>>4):%d\n"**,(**AdcRegs**.**ADCRESULT1**>>**4**));**

RES1 **=** getVin**((**AdcRegs**.**ADCRESULT0**>>**4**));**

RES2 **=** getVin**((**AdcRegs**.**ADCRESULT1**>>**4**));**

selectLights**();**

// AdcRegs.ADCTRL2.bit.RST\_SEQ1 = 1; // Reset SEQ1

AdcRegs**.**ADCST**.**bit**.**INT\_SEQ1\_CLR **=** 1**;** // Clear INT SEQ1 bit

PieCtrlRegs**.**PIEACK**.**all **=** PIEACK\_GROUP1**;** // Acknowledge interrupt to PIE

**}**

static \_\_interrupt void cpu\_timer0\_isr**(**void**)**

**{**

showLEDs**(**getVin**((**AdcRegs**.**ADCRESULT0**>>**4**)));**

// Acknowledge this interrupt to receive more interrupts from group 1

PieCtrlRegs**.**PIEACK**.**all **=** PIEACK\_GROUP1**;**

**}**

void demo9**(**void**)**

**{**

// 步骤 1. 初始化系统控制:

// 设置PLL, WatchDog, 使能外设时钟

// 下面这个函数可以从DSP281x\_SysCtrl.c文件中找到.

InitSysCtrl**();**

// 步骤 2. 初始化通用输入输出多路复用器GPIO:

// 这个函数在DSP281x\_Gpio.c源文件中被定义了

// InitGpio(); // 本例直接跳过该步骤

// 仅设置相应GPIO为为SPI功能引脚

// 本例特定的时钟设置

EALLOW**;**

SysCtrlRegs**.**HISPCP**.**all **=** ADC\_MODCLK**;** // HSPCLK = SYSCLKOUT/ADC\_MODCLK

EDIS**;**

//初始化SPI

EALLOW**;**

GpioMuxRegs**.**GPFMUX**.**all**=**0x000F**;** // 选择GPIO为SPI引脚

// 端口F MUX - x000 0000 0000 1111

EDIS**;**

Init\_LEDS\_Gpio**();**

Init\_Light\_Gpio**();**

// 步骤 3. 清除所有中断,初始化中断向量表:

// 禁止CPU全局中断

DINT**;**

// 初始化PIE控制寄存器到他们的默认状态.

// 这个默认状态就是禁止PIE中断及清除所有PIE中断标志

// 这个函数放在DSP281x\_PieCtrl.c源文件里

InitPieCtrl**();**

// 禁止CPU中断，清除CPU中断标志位

IER **=** 0x0000**;**

IFR **=** 0x0000**;**

//初始化PIE中断向量表，并使其指向中断服务子程序（ISR）

// 这些中断服务子程序被放在了DSP281x\_DefaultIsr.c源文件中

// 这个函数放在了DSP281x\_PieVect.c源文件里面.

InitPieVectTable**();**

// Interrupts that are used in this example are re-mapped to

// ISR functions found within this file.

EALLOW**;** // This is needed to write to EALLOW protected registers

PieVectTable**.**ADCINT **=** **&**adc\_int**;**

PieVectTable**.**TINT0 **=** **&**cpu\_timer0\_isr**;**

InitCpuTimers**();**

ConfigCpuTimer**(&**CpuTimer0**,** 150**,** 200000**);**

StartCpuTimer0**();**

// 步骤 4.初始化片内外设:

InitAdc**();** // For this example, init the ADC

spi\_fifo\_init**();** // 初始化Spi FIFO

spi\_init**();** // 初始化 SPI

// 本例需要设置ADC参数

AdcRegs**.**ADCTRL1**.**bit**.**ACQ\_PS **=** ADC\_SHCLK**;**

AdcRegs**.**ADCTRL3**.**bit**.**ADCCLKPS **=** ADC\_CKPS**;** // ADC 模块时钟 = HSPCLK/1 = 25MHz/(1) = 25MHz

AdcRegs**.**ADCTRL1**.**bit**.**SEQ\_CASC **=** 1**;** // 1 级联模式

AdcRegs**.**ADCCHSELSEQ1**.**bit**.**CONV00 **=** 0x0**;** //ADC通道选择ADCIN0 SEQ1

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*学号尾号通道\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

AdcRegs**.**ADCCHSELSEQ1**.**bit**.**CONV01 **=** 0x6**;** //ADC通道选择ADCINx

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

AdcRegs**.**ADCMAXCONV**.**all **=** 0x0001**;** // 最多采样 AdcRegs.ADCMAXCONV.all+1 个

AdcRegs**.**ADCTRL1**.**bit**.**CONT\_RUN **=** 1**;** // 设置为 连续运行

AdcRegs**.**ADCTRL2**.**bit**.**INT\_ENA\_SEQ1 **=** 1**;**

AdcRegs**.**ADCTRL2**.**bit**.**INT\_MOD\_SEQ1 **=** 0**;**

AdcRegs**.**ADCTRL2**.**bit**.**RST\_SEQ1 **=** 1**;**

AdcRegs**.**ADCTRL3**.**bit**.**SMODE\_SEL **=** 0**;**

// Enable CPU INT1

IER **|=** M\_INT1**;**

// Enable TINT0 in the PIE: Group 1 interrupt 7

PieCtrlRegs**.**PIEIER1**.**bit**.**INTx6 **=** 1**;** //adc

PieCtrlRegs**.**PIEIER1**.**bit**.**INTx7 **=** 1**;** //timer

// Enable global Interrupts and higher priority real-time debug events:

EINT**;** // Enable Global interrupt INTM

ERTM**;** // Enable Global realtime interrupt DBGM

// Step 5.用户指定代码，使能中断:

// 采样表清0

// for (i=0; i<BUF\_SIZE; i++)

// {

// SampleTable[i] = 0;

// }

//关数码管；

spi\_xmit**(**0xFFFF**);**

//延迟

delay\_loop8**();**

// 软件启动SEQ1

AdcRegs**.**ADCTRL2**.**bit**.**SOC\_SEQ1 **=** 1**;**

**while(**1**){}**

**}**

//===================================================================

// No more.

//===================================================================

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指导教师批阅意见：  验证性实验（报告）评分细则表   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 评分项目 | | 满分标准 | | | 成绩 | | 权重 | | 原始数据 | | 准确、真实可信，记录完整 | | |  | | 40％ | | 实验数据分析与处理 | | 分析与处理正确，有必要的过程，能恰当运用图表，分析全面、正确结论合理。 | | |  | | 20％ | | 实验报告 | | 实验报告格式规范，内容完整 | | |  | | 10％ | | 撰写质量 | | 撰写认真、报告整洁、清晰 | | |  | | 10％ | | 实验心得与思考题 | | 有心得体会，完成思考题 | | |  | | 20％ | | A（100~85） | B（84~75） | | C（74~65） | D（64~60） | | F（<60） | |   成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。