**深 圳 大 学 实 验 报 告**

|  |  |  |
| --- | --- | --- |
| **课程名称:** | **DSP原理与应用技术** |  |
| **实验名称:** | **串口与SPI** |  |
| **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**

# 实验目标

* **基础：看懂程序，在原程序基础上进行修改，通过8位SPI传输，在数码管0、3位显示“86”，1、2位不亮；**
* **提高：实现二位数码管从00-99递增循环。**

# 实验说明

## 背景

**SPI的基本概念**

串行外设接口SPI是原摩托罗拉公司推出的一种同步串行通讯接口，用于微处理器和外围扩展芯片之间的串行连接，目前已发展为一种工业标准。

目前各半导体公司推出大量带有SPI接口的芯片，为用户的外围扩展提供灵活和廉价的选择。

SPI一般使用4条线:串行时钟线SCK、主机输入/从机输出数据线MISO、主机输出/从机输入数据线MOSI和从机选择线SS。

**主机与从机：**

一个SPI系统，由一个主机和一个或多个从机构成，主机启动一个与从机的同步通讯，从而完成数据的交换。提供SPI串行时钟线的SPI设备称为SPI主机或主设备，其他设备则称为SPI从机或从设备。

**主出从入引脚MOSI与主入从出引脚MISO：**

主出从入引脚MOSI即主机输出、从机输入数据线。

主入从出引脚MISO即主机输入、从机输出数据线。

**SPI串行时钟引脚SCK：**

控制主机与从机之间的数据传输。串行时钟信号由主机的内部总线时钟分频获得，主机的SCK引脚输出给从机的SCK引脚，控制整个数据的传输速度。

**时钟极性与时钟相位：**

时钟极性表示时钟信号在空闲时是高电平还是低电平。

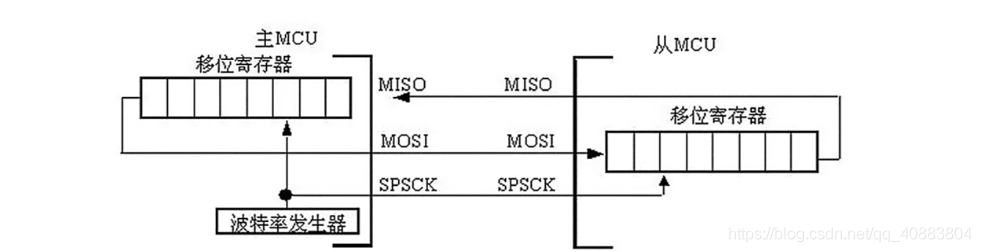
时钟相位表示时钟信号SCK的第一个边沿出现在第一位数据传输周期的开始位置还是中央位置。

**从机选择引脚SS：**

一些芯片带有从机选择引脚SS，也称为片选引脚。若一个MCU的SPI工作于主机方式，则该MCU的引脚设为高电平。若一个MCU的SPI工作于从机方式，当SS=0时表示主机选中了该从机，反之则未选中该从机。

## 原理

**一个SPI的基本连接图**



从主机CPU发出启动传输信号开始，将要传输的信号装入8位移位寄存器，并同时产生8位时钟信号依次从SCK引脚送出。在SCK信号的控制下，主机中8位移位寄存器中的数据一次从MOSI引脚送出，到从机的MOSI引脚后送入他的8位移位寄存器；在此过程中，从机的数据也可通过MISO引脚传送到主机中。

**SPI的时序**

SPI的数据传输是在时钟信号SCK的控制下完成的，该过程涉及时钟极性和时钟相位，主机和从机必须使用一致的时钟极性和相位才能正常通信

**对时钟极性于时钟相位的设置总体要求：**

确保发生数据在一周期开始的时刻上线，接收方在½周期的时刻从线上取数，这样是最稳定的通信方式。

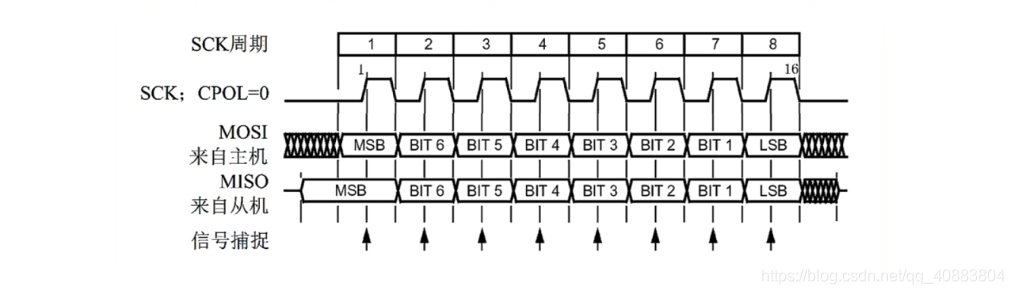
**对发送方编程必须明确要求：**

接收方要求的时钟空闲电平是高电平还是低电平；接收方在时钟的上升沿取数还是下降沿取数。

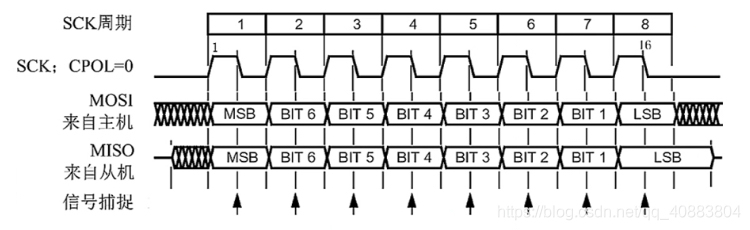
时钟极性由CPOL控制，时钟相位由CPHA来控制。二者结合起来共有四种可能取值情况。

**一、当CPOL=0，CPHA=0时**

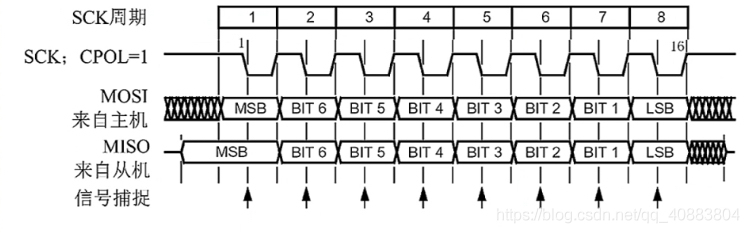
设置CPOL=0，表征空闲电平位低电平，然后设置CPHA=0，表征第一位数据提前半个时钟周期上线，这样设置保证了逐句与从机之间采用同样的时钟极性和时钟相位，才能正常通信。



**二、当CPOL=0，CPHA=1时**



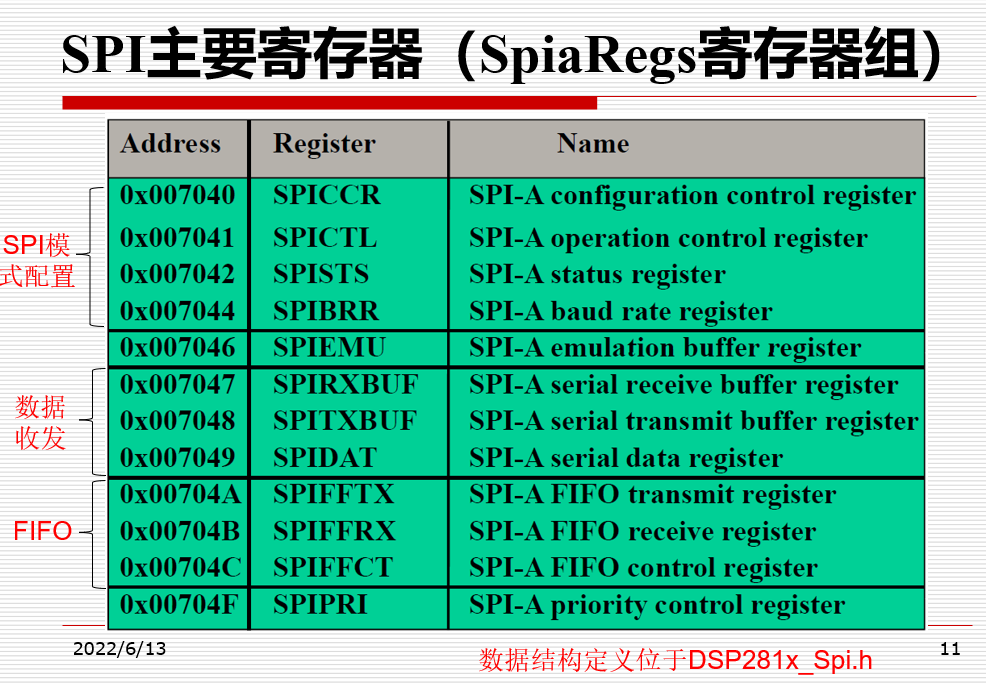
**三、当CPOL=1，CPHA=0时**

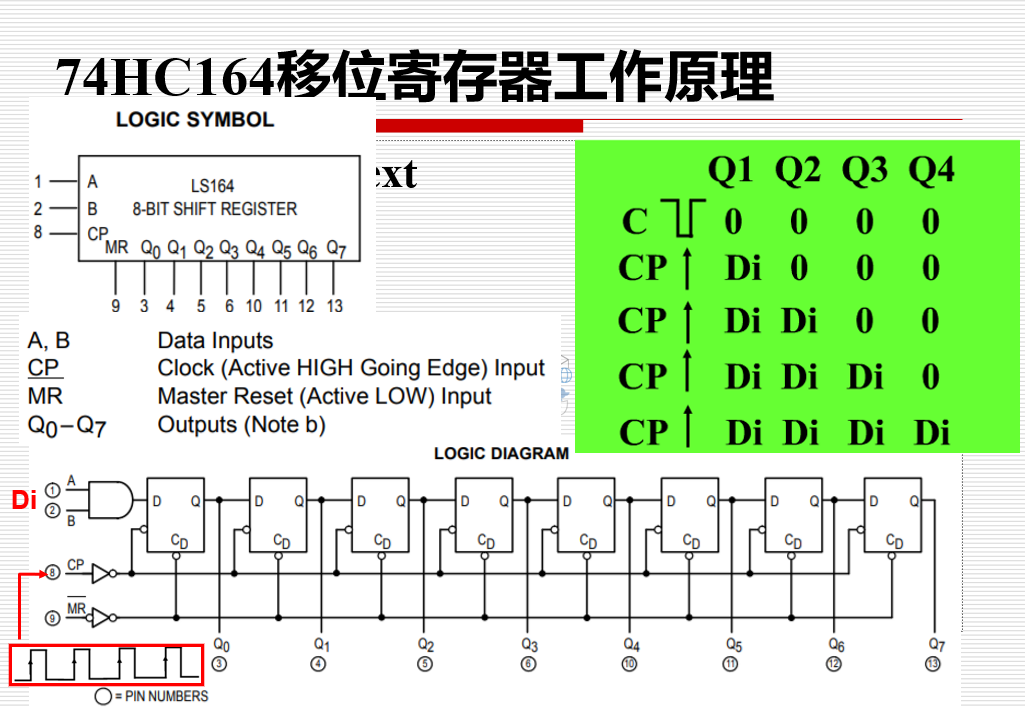


**四、当CPOL=1，CPHA=1时**



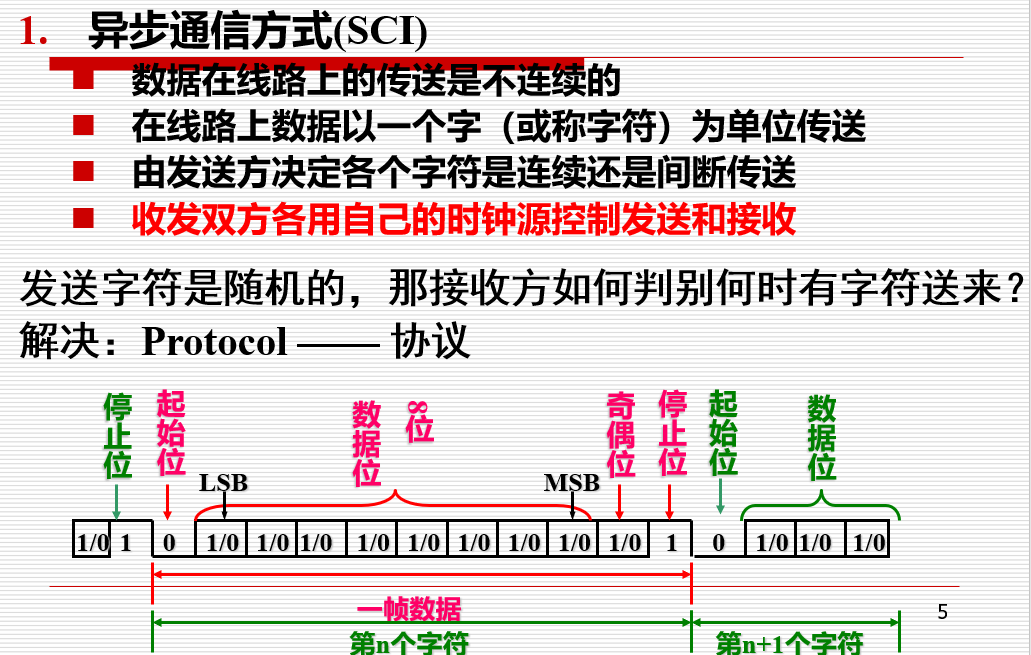
## 寄存器



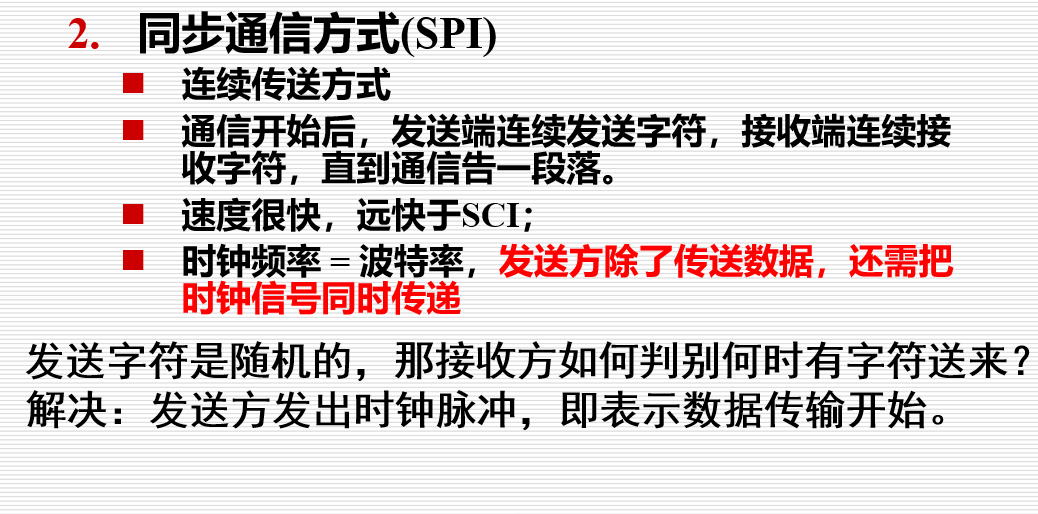


## 通信方式

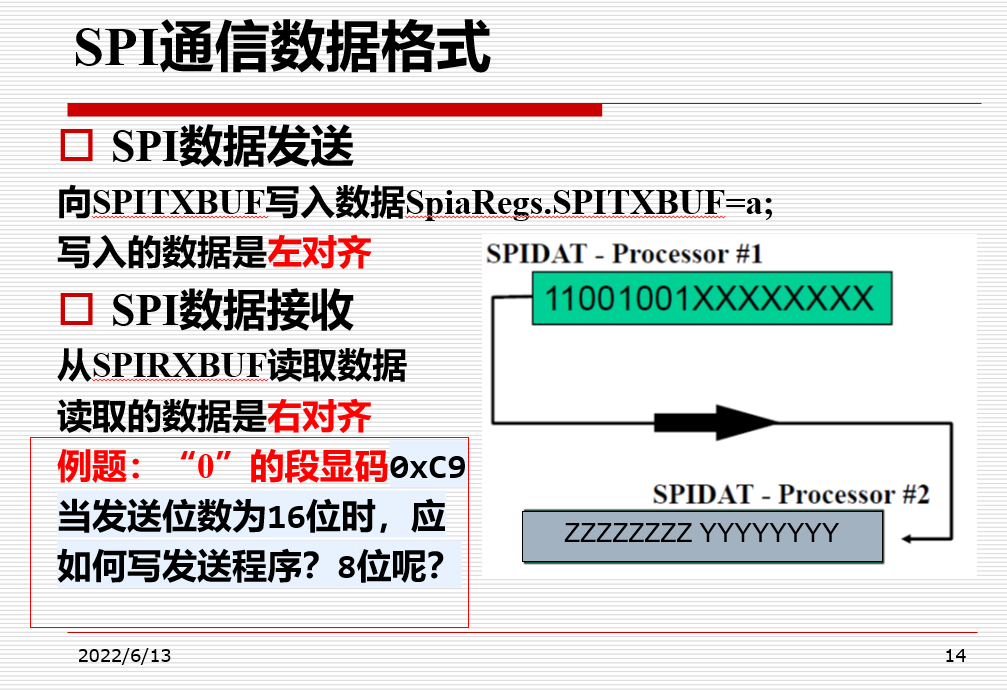
### 异步通信（SCI）



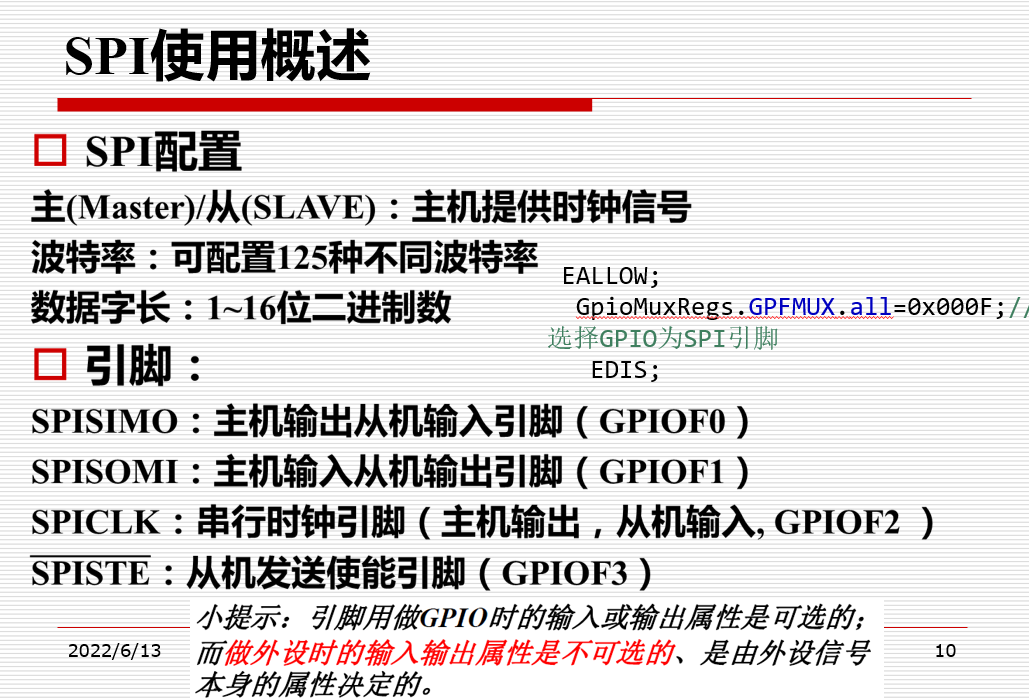
### 同步通信



### 通信格式



## 使用概述



# 实验过程

## 基础

看懂程序，在原程序基础上进行修改，通过8位SPI传输，在数码管0、3位显示“86”，1、2位不亮

### 设定常量

|  |
| --- |
| **代码：** |
| **说明：**  设定段码，扫描量等用以后续使用 |

### 定义函数

|  |
| --- |
| **代码：** |
| **说明：**  初始化数码管函数 |

### 初始化

|  |
| --- |
| **代码：** |
| **说明：**  设置SPI初始化函数 |

|  |
| --- |
| **代码：** |
| **说明：**  为了初始化系统，  调用了DSP281x\_SysCtrl.c文件里的InitSysCtrl()来设置PLL, WatchDog, 使能外设时钟  调用DisableDog()函数来关闭看门狗  调用InitPll(0xA)初始化PLLCR为10  调用InitPeripheralClocks()初始化外设时钟 |

### 清除所有中断,初始化中断向量表

|  |
| --- |
| **代码：** |
| **说明：**  调用DINT来禁止CPU全局中断；  调用InitPieCtrl()来初始化PIE控制寄存器到他们的默认状态，其中包括禁止PIE，清除所有PIEIER寄存器；  调用IER = 0x0000、IFR = 0x0000来禁止CPU中断和清除所有CPU中断标志；  初始化PIE中断向量表，并使其指向中断服务子程序（ISR）； |

### 初始化片内外设

|  |
| --- |
| **代码：** |
| **说明：**  调用spi\_fifo\_init()和spi\_info()函数来初始化 |

### 代码执行

|  |
| --- |
| **代码：** |
| **说明：**  已定义段码和spi，按要求选取86段码及对应GPIO口，执行函数show86()即可 |

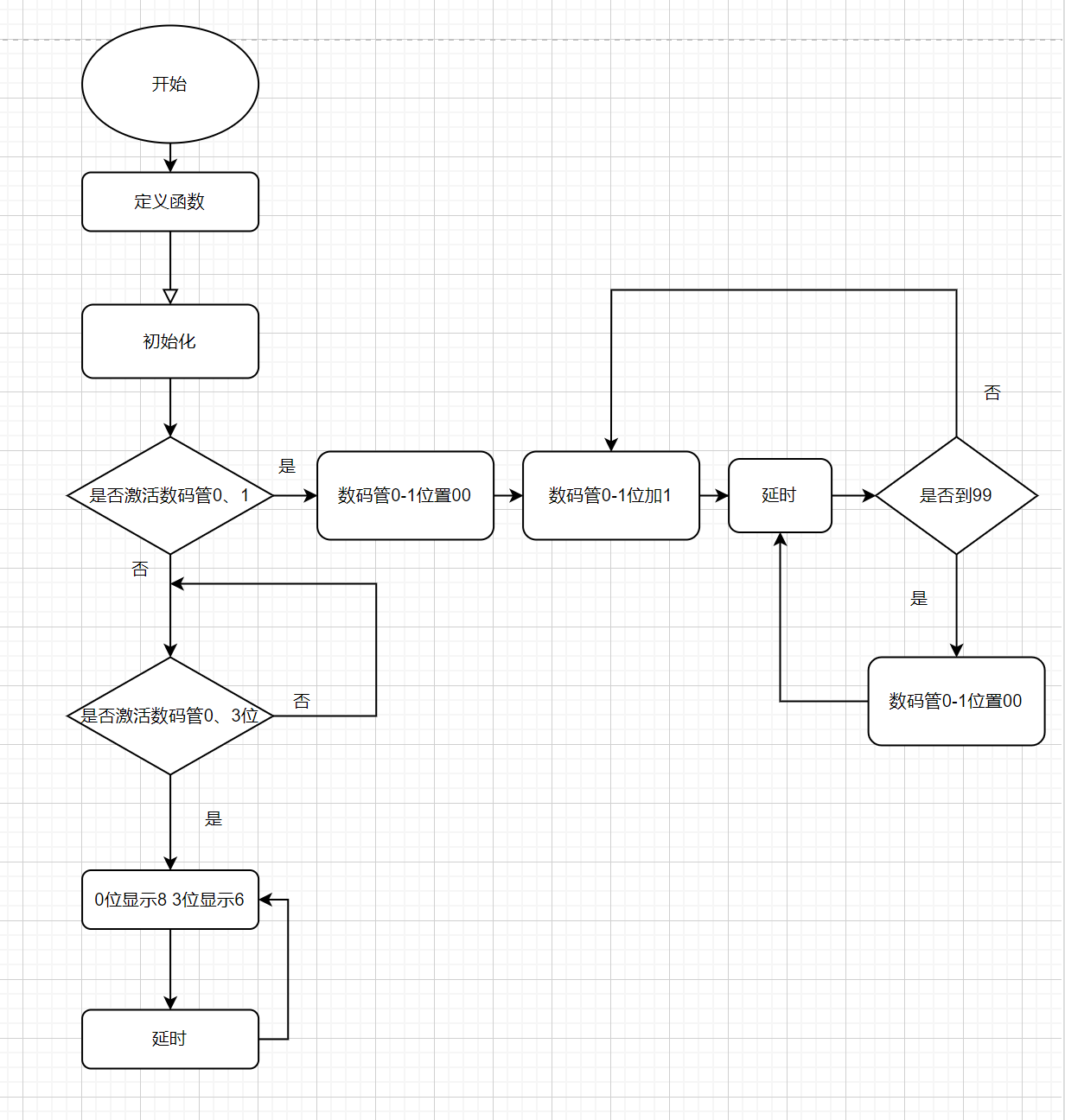
## 提高

实现二位数码管从00-99递增循环

### 代码执行

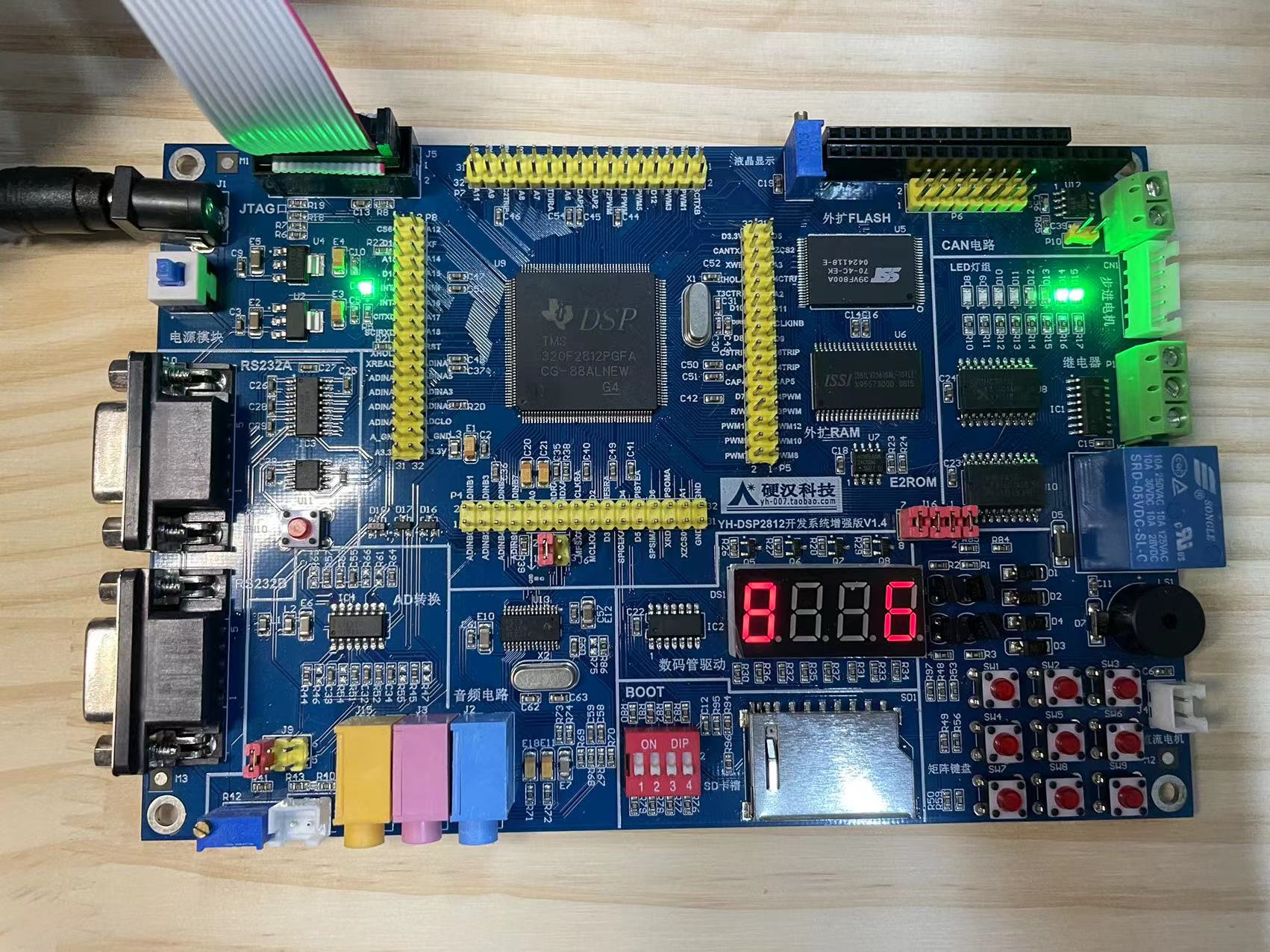
|  |
| --- |
| **代码：** |
| **说明：**  选取需要的晶体管位置和GPIO口，配合延迟函数delay()函数，既可实现要求 |

## 流程图



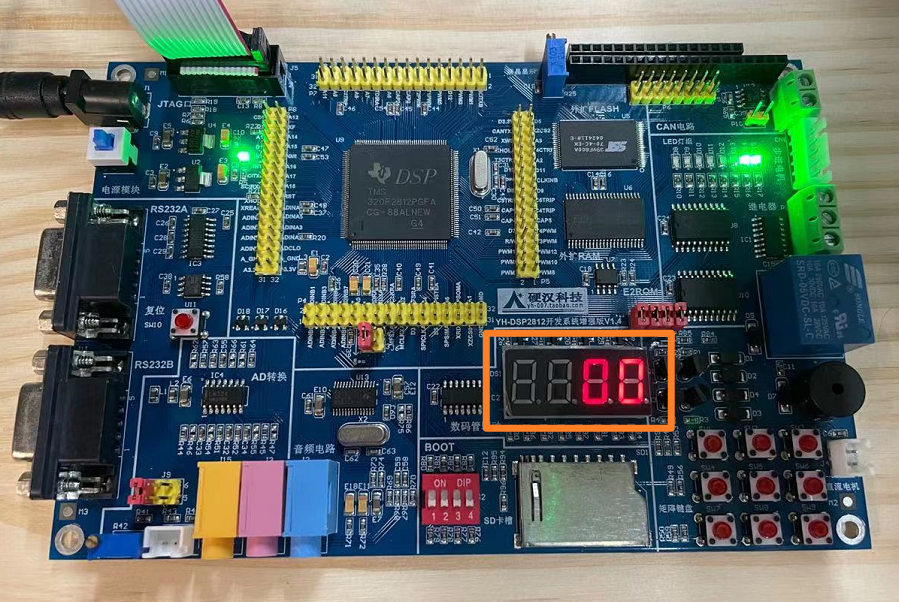
# 实验结果

1. 程序运行，数码管的第0位和第3位显示数字“8”和 “6” 。

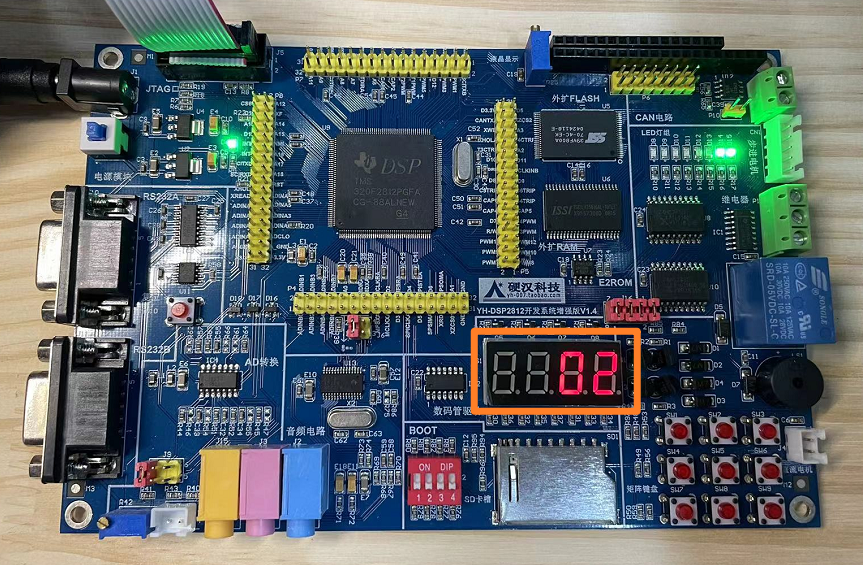


1. 程序运行，观察到数码管显示两位数字从00-99

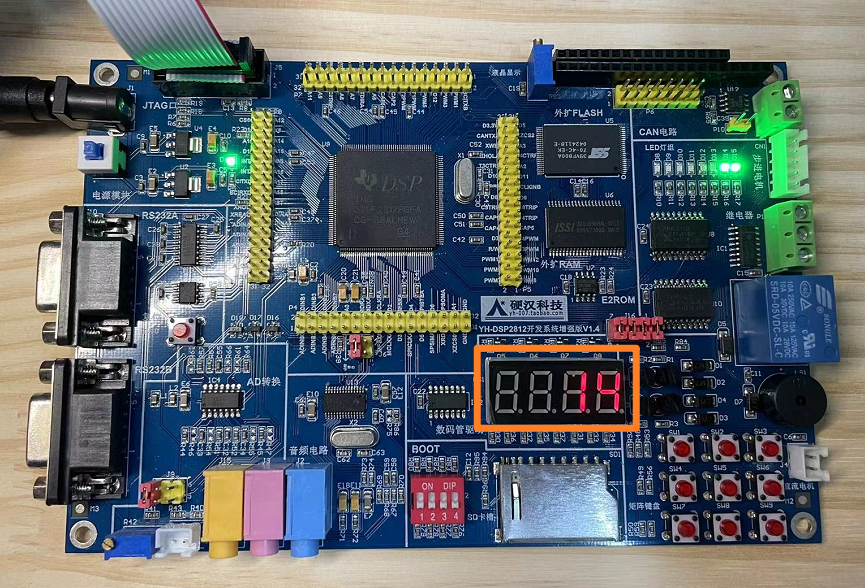
显示数字 ”00”



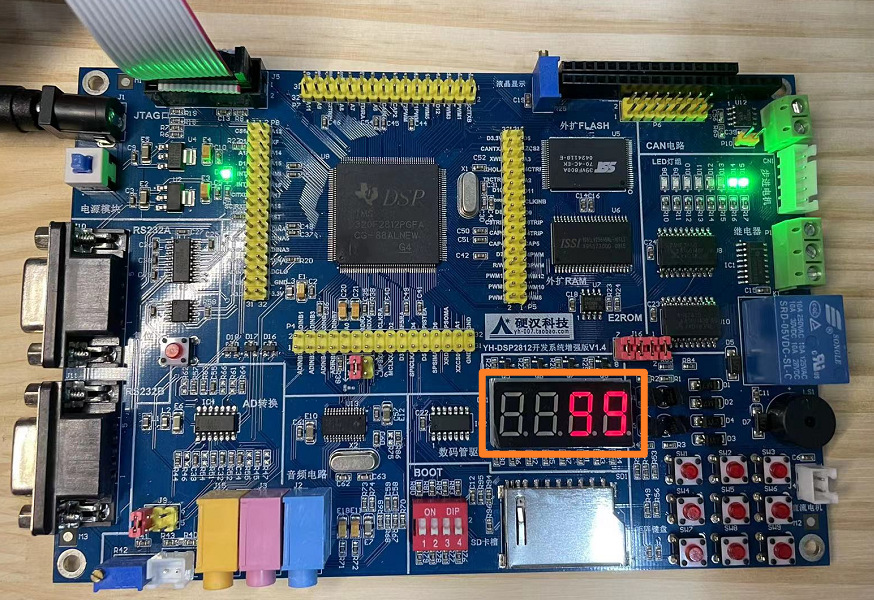
显示数字递增



…



…



# 心得体会

本次实验通过学习并配置串口SPI，掌握了SPI的驱动方式，事件的工作原理和初始化方法，并结合了数码管，较好地完成了实验任务

过程难点:

1. 在对实验进行预习和理解的过程中，需要对 串口SPI和 数码管 有一定的认知和理解，需要花费一定的时间和精力去读懂其原理。
2. 事件管理器是在学习DSP开发的难点与重点，初学者需要理解其开发原理，需要对相关寄存器的配置有一定的掌握。

# 代码附录

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

\* 文件名： 数码管显示程序

\* 描述: 执行该程序，数码管显示数字8\*\*0,

\* 一段时间后，开始显示 00->99 递增

\* Author: team0ne

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

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

程序说明：

更换控制平台，仅需更改IO口初始化、宏定义 以及spi初始化/

实验结果：

4位数码管显示8 6；

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

#include "demo3.h"

#include "demo1.h"

void show86**(**void**);**

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

Uint16 DisData **=** 8006**;** //显示的数字

Uint16 Loop **=** 0**;** //循环扫描变量

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

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

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）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

**{**

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

**{**

**case** 0**:**

RST\_BIT1**;**

SET\_BIT1**;**

RST\_BIT2**;**

RST\_BIT3**;**

RST\_BIT4**;** //关断数码管第四位

//选通数码管第一位

**break;**

**case** 1**:**

RST\_BIT1**;**

RST\_BIT2**;**

SET\_BIT2**;** //选通数码管第二位

RST\_BIT3**;**

RST\_BIT4**;** //关断数码管第一位

**break;**

**case** 2**:**

RST\_BIT1**;**

RST\_BIT2**;**

RST\_BIT3**;**

SET\_BIT3**;** //选通数码管第三位

RST\_BIT4**;** //关断数码管第二位

**break;**

**case** 3**:**

RST\_BIT1**;**

RST\_BIT2**;**

RST\_BIT3**;**

RST\_BIT4**;** //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【】\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

void delay1**(**unsigned long int t**)**

**{**

unsigned long int i **=** 0**;**

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

**}**

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

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

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

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

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

**{**

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

**}**

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

static void show86**(**void**){**

DisData **=** 8006**;**

int counter **=** 0**;**

**for(;;)**

**{**

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

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

**{**

**if(**Loop **==** 1**)** Loop **=** 3**;**

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

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

delay1**(**2500**);** //延时配合人眼反应时间 250000 -> 250

**}**

counter**++;**

**if(**counter **>** 500**)** **return;**

**}**

**}**

static void show\_counter**(**void**){**

DisData **=** 0**;**

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

**if(**DisData **==** 100**)** DisData **=** 0**;**

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

Loop **=** **(**DisData **%**2**);**

**for(**Loop **=** 0**;**Loop **<** 100**;**Loop**++){**

Sellect\_Bit**(**Loop**%**2**);** //选择要扫描的数码管位

spi\_xmit**(**msg**[**DisData\_Bit**[**Loop**%**2**]]);** //串行输出要显示的数字

delay1**(**2500**);**

**}**

delay1**(**25000**);**

DisData**++;**

**}**

**}**

void demo3**(**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

//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**();**

// 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:

spi\_fifo\_init**();** // 初始化Spi FIFO

spi\_init**();** // 初始化 SPI

// 执行显示

show86**();**

show\_counter**();**

**}**

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

// 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。