驱动程序设计

该小系统主要实现温度采集与电机速度控制，这是一个典型的控制系统实例。

1 数码管显示控制

由电路图可知，PC12 控制 LED，输出低时 LED 亮，输出高时 LED 灭。共阳七段数码管分别连接 PC3、PC4、PC5、PC8、PC9、PC10、PC11。某管脚为低电平时，相应的 LED 亮，为高电平时灭。数字 0～9 显示与七个管脚的电平对应如表 1 所示。

表 1　共阳数码管显示控制电平

	g(PC11)	f(PC10)	e(PC9)	d(PC8)	c(PC5)	b(PC4)	a(PC3)	输出值
0	1	0	0	0	0	0	0	0x800
1	1	1	1	1	0	0	1	0xF08
2	0	1	0	0	1	0	0	0x420
3	0	1	1	0	0	0	0	0x600
4	0	0	1	1	0	0	1	0x308
5	0	0	1	0	0	1	0	0x210
6	0	0	0	0	0	1	0	0x010
7	1	1	1	1	0	0	0	0xF00
8	0	0	0	0	0	0	0	0x000
9	0	0	1	0	0	0	0	0x200

定义函数 void LedInit()来配置 PC12 为推挽低速无上下拉输出管脚。

定义函数 void LedSw()来进行亮灭切换，即输出电平翻转。

定义函数 void SegInit()来初始化七段数码管的驱动电路，主要完成使能 GPIOC 时钟、配置 PC3、PC4、PC5、PC8、PC9、PC10、PC11 为低速开漏无上下拉输出并对各管脚置位。

定义函数 void SegDisp(char num)用来显示数值。对于有效数值，根据该数值所对应的电平组合分别对 GPIOC 中的相应位进行置位或清零操作；对于无效数值，将所有相应位进行置位操作，使数码管无显示。

==========segdisplay.h==========

```
#ifndef _SEGDISPLAY_H
#define _SEGDISPLAY_H
void LedInit();
void LedSw();
void SegInit();
```

```
void SegDisp( unsigned char num);
#endif
==========segdisplay.c===========
#include "stm32f4xx.h"
#include "segdisplay.h"
void LedInit()
{//配置 PC12 为推挽低速无上下拉输出
    RCC->AHB1ENR |= 0x4;//使能 GPIOC 时钟
    GPIOC->MODER = (GPIOC->MODER & ~0x3000000)|0x1000000;//输出管脚
    GPIOC->OTYPER &= ~0x1000; //推挽输出
    GPIOC->OSPEEDR &= ~ 0x3000000;//低速
    GPIOC->PUPDR &= ~ 0x3000000;//无上拉无下拉
    GPIOC->ODR &= ~ 0x1000; //指示灯亮
}
void LedSw()
{
    GPIOC->ODR ^= 0x1000;//PC12 取反
}
void SegInit()
{//配置 PC3、PC4、PC5、PC8、PC9、PC10、PC11 为低速开漏无上下拉输出
    RCC->AHB1ENR |= 0x4;//使能 GPIOC 时钟
    GPIOC->MODER = (GPIOC->MODER & ~0xFF0FC0)|0x550540;//输出管脚
    GPIOC->OTYPER |= 0xF38; //开漏输出
    GPIOC->OSPEEDR &= ~ 0xFF0FC0;//低速
    GPIOC->PUPDR &= ~ 0xFF0FC0;//无上拉无下拉
    GPIOC->ODR |= 0xF38; //置高（不亮）
}
static int disp_bits[]={0x800, 0xF08, 0x420, 0x600, 0x308, 0x210, 0x010,
0xF00, 0x000, 0x200};
void SegDisp( unsigned char num )
{
    if (num<=9)
        GPIOC->ODR = (GPIOC->ODR & ~0xF38)|disp_bits[num];
    else
        GPIOC->ODR |= 0xF38;
}
```

2 温度测量控制

由电路可知，PC2 作为 ADC 的输入 ADC1_IN12。

定义函数 void TmepMeasInit()配置 ADC1 参数，即常规转换为非连续模式、精度为 12bit、非连续通道数为 1、单次模式、右对齐、非外部触发，最后使能 ADC。

定义函数 void TempMeasRun()启动 ADC。

定义函数 int TempMeasGet(unsigned char *ptemp)返回获取是否成功及获取采样值高 8 位更新温度测量值。

```
==========tempmeasure.h==========
#ifndef _TEMPMEASURE_H
#define _TEMPMEASURE_H
void TempMeasInit();
void TempMeasRun();
int TempMeasGet(unsigned char *ptemp);
#endif
========== tempmeasure.c==========
#include "stm32f4xx.h"
#include " tempmeasure.h"
void TempMeasInit()
{
    RCC->AHB1ENR |= 0x1<<2;
    GPIOC->MODER |= 0x3<<4; //配置 ADC1_IN12 的输入管脚（PC2）
    RCC->APB2ENR |= 0x1<<8; //使能 ADC1 的时钟
    ADC->CCR |= 3;
    ADC1->SMPR1 |= 0x7<<6;
    ADC1->SQR1 = 0;
    ADC1->SQR3 = 12;
    ADC1->CR1 = 1<<11;
    ADC1->CR2 = 1;
}
void TempMeasRun()
{
    ADC1->SR &= ~0x2;
    ADC1->CR2 |= 1<<30;
}
int TempMeasGet(unsigned char *ptemp)
{
    if(!(ADC1->SR & 2))
        return 0;
    *ptemp = (unsigned char)(ADC1->DR>>4);
    return 1;
}
```

3 按钮控制

定义函数 void BtnInit()初始化按钮输入端，使能 GPIOC 时钟和配置 PC13 为无上拉下拉输入。

定义函数 void BtnIntEn(void (*isr)())来配置 PC13 的输入电平下降沿产生中断及中断处理所要调用的外部程序。

定义函数 void EXTI15_10_IRQHandler()作为 PC13 所产生的中断的处理函数。

```
==========button.h==========
#ifndef _BUTTON_H
#define _BUTTON_H
```

```
void BtnInit();
void BtnIntEn(void (*isr)());
void EXTI15_10_IRQHandler(void);
#endif
==========button.c============
#include <stm32f4xx.h>
#include "button.h"
static void (*btn_isr)() = 0;
void BtnInit()
{
    RCC->AHB1ENR |= 1<<2; //使能 GPIOC 时钟

    GPIOC->MODER &= ~(0x3<<26);
    GPIOC->PUPDR &= ~(0x3<<26);
    btn_isr = 0;
}
void BtnIntEn(void (*isr)())
{
    btn_isr = isr;
    RCC->APB2ENR |= 1<<14; //使能 SYSCFG 时钟
    SYSCFG->EXTICR[3] = (SYSCFG->EXTICR[3] & ~(0xF<<4))| (2<<4); //EXTI13 信
号源为 PC13
    EXTI->IMR |= 1<<13; //取消对 EXTI13 信号线的屏蔽
    EXTI->FTSR |= 1<<13; //设定 EXTI13 中断触发信号为下降沿
    NVIC->ISER[1] |= 1<<8; //在 NVIC 中设置 EXTI15_10 中断使能
}
void EXTI15_10_IRQHandler(void)
{
    EXTI->PR |= 1<<13; //清除当前已经产生的 EXTI13 中断
    if(btn_isr)
        btn_isr();
}
```

4 异步串口收发控制

异步串口采用 USART6，其接收采用中断方式，而发送采用单字符方式。

定义函数 void UartInit()初始化 USART6 并使能接收中断。

定义函数 int UartTx(char txd)实现对字符的发送。若可以发送，则发送且返回成功，否则返回失败。

定义函数 int UartRx(char *prxd)实现对字符的接收。若收到数据，则读取且返回成功，否则返回失败。

定义函数 void UartRxIntEn(void (*isr)(char))使能接收中断及配置接收中断服务程序所要调用的外部处理程序。

定义函数 void USART6_IRQHandler()实现接收中断处理，读取每次接收到的字符，并调用处理函数。

==========uart.h==========

```
#ifndef _UART_H
#define _UART_H
void UartInit();
int UartTx(char txd);
int UartRx(char *prxd);
void UartRxIntEn(void (*isr)(char));
void USART6_IRQHandler();
#endif
```

==========uart.c==========

```
#include <stm32f4xx.h>
#include "uart.h"
static void (*rx_isr)() = 0;
void UartInit()
{
    RCC->AHB1ENR |= 1; //使能 GPIOA 时钟
    RCC->APB2ENR |= 1<<5; //使能 USART6 时钟
    //配置 PA11、PA12 为复用、推挽输出、高速
    GPIOA->MODER = (GPIOA->MODER & ~(0xF<<22))| (0xA<<22);
    GPIOA->OTYPER &= ~(0x3<<11);
    GPIOA->OSPEEDR = (GPIOA->OSPEEDR & ~(0xF<<22)) | (0xA<<22);
    //设置 AFRH 寄存器，PA11 和 PA12 复用模式为 AF7，分别为 U6TX 和 U6RX
    GPIOA->AFR[1]= (GPIOA->AFR[1] & ~(0xFF<<12)) | (0x88<<12);
    USART6->BRR = 1667;
    USART6->CR1 = (1<<13)|(1<<3)|(1<<2); //使能接收和发送功能
    rx_isr = 0;
}
int UartTx(char txd)
{
    if(!(USART6->SR & (0x1<<7)))
        return 0;
    USART6->DR = txd;
    return 1;
}
int UartRx(char *prxd)
{
    if(!(USART6->SR & (0x1<<5)))
        return 0;
    *prxd = USART6->DR;
    return 1;
}
void UartRxIntEn(void (*isr)(char))
{
    USART6->CR1 |= 1<<5;
```

```
    NVIC->ISER[2] |= 1<<7;
    rx_isr = isr;
}
void USART6_IRQHandler(void)
{
    char rxd;
    if(UartRx(&rxd))
    {
        if(rx_isr)
            rx_isr(rxd);
    }
}
```

5 电机驱动控制

采用定时器 3 的通道 1 产生 PWM。

定义函数 void MotorInit()使能定时器 3 时钟、配置 PC6 为推挽高速无上拉下拉输出、初始化 TIM3 周期和 CH1 的比较参数。

定义函数 void MotorSpeedSet(unsigned char ratio)来设置 PWM 的高电平时长以控制电机转速。

```
==========motorctrl.h==========
#ifndef _MOTORCTRL_H
#define _MOTORCTRL_H
void MotorInit(void);
void MotorSpeedSet( unsigned char ratio );
#endif
==========motorctrl.c===========
#include <stm32f4xx.h>
#include "motorctrl.h"
void MotorInit(void)
{
    RCC->AHB1ENR  |= 1<<2; //使能 GPIOC 时钟
    GPIOC->MODER = (GPIOC->MODER & ~(3<<12)) | (2<<12); //设置 PC6 为 AF
    GPIOC->OTYPER &= ~(1<<6);//设置 PC6 为推挽
    GPIOC->OSPEEDR = (GPIOC->OSPEEDR & ~(3<<12)) | (2<<12);//设置 PC6 为高速
    GPIOC->AFR[0] = (GPIOC->AFR[0] & ~(0xf<<24))|(2<<24); //将 PC6 复用为 TIM3
的 CH1 输出
    RCC->APB1ENR |= 1<<1; //使能 TIM3 时钟

    TIM3->CR1 = 0x80; //设定为边沿对齐，向上计数工作模式
    TIM3->ARR = 159; //设定计数器分频数
    TIM3->PSC = 999; //设定预分频数
    TIM3->CCR1 = 0; //配置 CH1 为 PWM1 输出模式
    TIM3->CCER |= 1<<0; //使能比较通道 1 作为输出
    TIM3->CCMR1 = (TIM3->CCMR1 & ~(0xFF<<0)) | (0x68<<0);
```

```
    TIM3->CR1 |= 0x1; //开启 TIM2
}
void MotorSpeedSet( unsigned char ratio )
{
    if( ratio >=10 )
        TIM3->CCR1 = TIM3->ARR+1;
    else
        TIM3->CCR1 = (TIM3->ARR+1) * ratio/10;
}
```

6 时序控制

数码管的点作为工作指示灯每 0.5 秒闪亮切换一次，温度测量每 0.2 秒进行一次。用 TIM4 产生 0.5 秒的定时来控制工作指示灯，用定时器 TIM2 产生 0.2 秒的定时来测量温度。

```
==========timer.h=========
#ifndef _TIMER_H
#define _TIMER_H
void Tim2Init();
void Tim2IntEn(void (*isr)());
void TIM2_IRQHandler();
void SysTickInit();
void SysTickIntEn(void (*isr)());
void SysTick_Handler();
#endif
============timer.c===========
#include <stm32f4xx.h>
#include "timer.h"
static void (*tim2_isr)()=0;
void Tim2Init()
{
    RCC->APB1ENR |=1<<0; //使能 TIM2
    TIM2->CR1 = 1<<7; //设定为边沿对齐，向上计数工作模式
    TIM2->ARR = 3199; //设定计数器分频数
    TIM2->PSC = 999; //设定预分频数
    TIM2->CR1 |= 1<<0; //开启 TIM2
    tim2_isr = 0;
}
void Tim2IntEn(void (*isr)())
{
    tim2_isr = isr;
    TIM2->DIER |= 1<<0; //设置中断更新使能
    NVIC->ISER[0] |= 1<<28; //在 NVIC 中设置 TIM2 中断使能
}
void TIM2_IRQHandler(void)
{
```

```
    TIM2->SR &= ~(1<<0); //清除当前中断事件;
    if(tim2_isr)
        tim2_isr();
}
static void (*tim4_isr)()=0;
void Tim4Init()
{
    RCC->APB1ENR |=1<<2; //使能 TIM2
    TIM4->CR1 = 1<<7; //设定为边沿对齐，向上计数工作模式
    TIM4->ARR = 7999; //设定计数器分频数
    TIM4->PSC = 999; //设定预分频数
    TIM4->CR1 |= 1<<0; //开启 TIM2
    tim4_isr = 0;
}
void Tim4IntEn(void (*isr)())
{
    tim4_isr = isr;
    TIM4->DIER |= 1<<0; //设置中断更新使能
    NVIC->ISER[0] |= 1<<30; //在 NVIC 中设置 TIM4 中断使能
}
void TIM4_IRQHandler(void)
{
    TIM4->SR &= ~(1<<0); //清除当前中断事件;
    if(tim4_isr)
        tim4_isr();
}
```