1. pwm初始化 time3定时器 四个通道

void TIM3\_PWM\_Init(u16 arr,u16 psc)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

//TIM3 CH1 1A4 CH2 1B4 （pa6 pa7）

//TIM3 CH3 1A2 CH4 1B2 (pb0,pb1)

//TIM4 CH1 1A3 CH2 1B3 (pb6,pb7)

//TIM4 CH3 1A1 CH4 1B1 (pb8,pb9)

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3, ENABLE); //使能定时器3时钟

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA , ENABLE); //使能GPIO外设和AFIO复用功能模块时钟

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB , ENABLE);

//GPIO\_PinRemapConfig(GPIO\_PartialRemap\_TIM3, ENABLE); //Timer3部分重映射 TIM3\_CH2->PB5

//设置该引脚为复用输出功能,输出TIM3 CH2的PWM脉冲波形 GPIOB.5

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6|GPIO\_Pin\_7; //TIM\_CH2

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);//初始化GPIO

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1; //TIM\_CH2

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);//初始化GPIO

//初始化TIM3

TIM\_TimeBaseStructure.TIM\_Period = arr; //设置在下一个更新事件装入活动的自动重装载寄存器周期的值

TIM\_TimeBaseStructure.TIM\_Prescaler =psc; //设置用来作为TIMx时钟频率除数的预分频值

TIM\_TimeBaseStructure.TIM\_ClockDivision = 0; //设置时钟分割:TDTS = Tck\_tim

TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up; //TIM向上计数模式

TIM\_TimeBaseInit(TIM3, &TIM\_TimeBaseStructure); //根据TIM\_TimeBaseInitStruct中指定的参数初始化TIMx的时间基数单位

//初始化TIM3 Channel2 PWM模式

TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM2; //选择定时器模式:TIM脉冲宽度调制模式2

TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable; //比较输出使能

TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_High; //输出极性:TIM输出比较极性高

TIM\_OC2Init(TIM3, &TIM\_OCInitStructure); //根据T指定的参数初始化外设TIM3 OC2

TIM\_OC1Init(TIM3, &TIM\_OCInitStructure);

TIM\_OC3Init(TIM3, &TIM\_OCInitStructure);

TIM\_OC4Init(TIM3, &TIM\_OCInitStructure);

TIM\_OC2PreloadConfig(TIM3, TIM\_OCPreload\_Enable); //使能TIM3在CCR2上的预装载寄存器

TIM\_OC1PreloadConfig(TIM3, TIM\_OCPreload\_Enable); //使能TIM3在CCR2上的预装载寄存器

TIM\_OC3PreloadConfig(TIM3, TIM\_OCPreload\_Enable); //使能TIM3在CCR2上的预装载寄存器

TIM\_OC4PreloadConfig(TIM3, TIM\_OCPreload\_Enable); //使能TIM3在CCR2上的预装载寄存器

//GPIO\_ResetBits(GPIOA,)

TIM\_Cmd(TIM3, ENABLE); //使能TIM3

}

1. 舵机初始化 使用time4 使用一个通道

void TIM4\_PWM\_Init(u16 arr,u16 psc)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

//TIM3 CH1 1A4 CH2 1B4 （pa6 pa7）

//TIM3 CH3 1A2 CH4 1B2 (pb0,pb1)

//TIM4 CH1 1A3 CH2 1B3 (pb6,pb7)

//TIM4 CH3 1A1 CH4 1B1 (pb8,pb9)

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4, ENABLE); //使能定时器3时钟

//RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_AFIO, ENABLE); //使能GPIO外设和AFIO复用功能模块时钟

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB , ENABLE);

//GPIO\_PinRemapConfig(GPIO\_Remap\_TIM4, ENABLE);

//设置该引脚为复用输出功能,输出TIM3 CH2的PWM脉冲波形 GPIOB.5

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_6|GPIO\_Pin\_7|GPIO\_Pin\_8|GPIO\_Pin\_9; //TIM\_CH2

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOB, &GPIO\_InitStructure);//初始化GPIO

//初始化TIM4

TIM\_TimeBaseStructure.TIM\_Period = arr; //设置在下一个更新事件装入活动的自动重装载寄存器周期的值

TIM\_TimeBaseStructure.TIM\_Prescaler =psc; //设置用来作为TIMx时钟频率除数的预分频值

TIM\_TimeBaseStructure.TIM\_ClockDivision = 0; //设置时钟分割:TDTS = Tck\_tim

TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up; //TIM向上计数模式

TIM\_TimeBaseInit(TIM4, &TIM\_TimeBaseStructure); //根据TIM\_TimeBaseInitStruct中指定的参数初始化TIMx的时间基数单位

//初始化TIM3 Channel2 PWM模式

TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM2; //选择定时器模式:TIM脉冲宽度调制模式2

TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable; //比较输出使能

TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_High; //输出极性:TIM输出比较极性高

TIM\_OC2Init(TIM4, &TIM\_OCInitStructure); //根据T指定的参数初始化外设TIM3 OC2

TIM\_OC1Init(TIM4, &TIM\_OCInitStructure);

TIM\_OC3Init(TIM4, &TIM\_OCInitStructure);

TIM\_OC4Init(TIM4, &TIM\_OCInitStructure);

TIM\_OC2PreloadConfig(TIM4, TIM\_OCPreload\_Enable); //使能TIM3在CCR2上的预装载寄存器

TIM\_OC1PreloadConfig(TIM4, TIM\_OCPreload\_Enable); //使能TIM3在CCR2上的预装载寄存器

TIM\_OC3PreloadConfig(TIM4, TIM\_OCPreload\_Enable); //使能TIM3在CCR2上的预装载寄存器

TIM\_OC4PreloadConfig(TIM4, TIM\_OCPreload\_Enable); //使能TIM3在CCR2上的预装载寄存器

TIM\_Cmd(TIM4, ENABLE); //使能TIM3

}

1. ATGM336初始化 得到经度纬度高度

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

接线说明：

STM32 GPS

VCC ------> VCC

GND ------> GND

RX1 <------ TXD

STM32 USB-TTL模块

GND ------> GND

TX1 ------> RXD

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

#include "stm32f10x.h"

#include "delay.h"

#include "usart.h"

#include "led.h"

//声明

void errorLog(int num);

void parseGpsBuffer(void);

void printGpsBuffer(void);

int main(void)

{

delay\_init();

NVIC\_Configuration(); //设置NVIC中断分组2:2位抢占优先级，2位响应优先级

uart\_init(9600); //串口初始化为9600

Init\_LEDpin();

LED1 = 1;

clrStruct();

printf("Welcome to use!\r\n");

printf("ILoveMcu.taobao.com!\r\n");

while(1)

{

parseGpsBuffer();

printGpsBuffer();

}

}

void errorLog(int num)

{

while (1)

{

printf("ERROR%d\r\n",num);

}

}

void parseGpsBuffer()

{

char \*subString;

char \*subStringNext;

char i = 0;

if (Save\_Data.isGetData)

{

Save\_Data.isGetData = false;

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r\n");

printf(Save\_Data.GPS\_Buffer);

for (i = 0 ; i <= 6 ; i++)

{

if (i == 0)

{

if ((subString = strstr(Save\_Data.GPS\_Buffer, ",")) == NULL)

errorLog(1); //解析错误

}

else

{

subString++;

if ((subStringNext = strstr(subString, ",")) != NULL)

{

char usefullBuffer[2];

switch(i)

{

case 1:memcpy(Save\_Data.UTCTime, subString, subStringNext - subString);break; //获取UTC时间

case 2:memcpy(usefullBuffer, subString, subStringNext - subString);break; //获取UTC时间

case 3:memcpy(Save\_Data.latitude, subString, subStringNext - subString);break; //获取纬度信息

case 4:memcpy(Save\_Data.N\_S, subString, subStringNext - subString);break; //获取N/S

case 5:memcpy(Save\_Data.longitude, subString, subStringNext - subString);break; //获取经度信息

case 6:memcpy(Save\_Data.E\_W, subString, subStringNext - subString);break; //获取E/W

default:break;

}

subString = subStringNext;

Save\_Data.isParseData = true;

if(usefullBuffer[0] == 'A')

Save\_Data.isUsefull = true;

else if(usefullBuffer[0] == 'V')

Save\_Data.isUsefull = false;

}

else

{

errorLog(2); //解析错误

}

}

}

}

}

void printGpsBuffer()

{

if (Save\_Data.isParseData)

{

Save\_Data.isParseData = false;

printf("Save\_Data.UTCTime = ");

printf(Save\_Data.UTCTime);

printf("\r\n");

if(Save\_Data.isUsefull)

{

Save\_Data.isUsefull = false;

printf("Save\_Data.latitude = ");

printf(Save\_Data.latitude);

printf("\r\n");

printf("Save\_Data.N\_S = ");

printf(Save\_Data.N\_S);

printf("\r\n");

printf("Save\_Data.longitude = ");

printf(Save\_Data.longitude);

printf("\r\n");

printf("Save\_Data.E\_W = ");

printf(Save\_Data.E\_W);

printf("\r\n");

}

else

{

printf("GPS DATA is not usefull!\r\n");

}

}

}