嵌入式期末考试复习资料

原创：Linda Silk & Laurel Shay

👉阅读原文👈

2019/12/21

21.ST-Link未装驱动或坏了会出现什么情况

---

1. 选择与填空

1. PID控制器分别是哪三个含义（单词）的缩写：比例、积分、微分

   PID：Proportional-Integral-Derivative control

2. 嵌入式系统的设计三个阶段：分析、设计与实现
3. 嵌入式开发模式：本机开发、交叉开发、模拟开发
4. 嵌入式系统一般采用交叉开发模式

   宿主机上编译、链接；目标机上运行、调试

5. 根据是**否具有实时性能**可以分为：实时系统、非实时系统
6. 根据实时**响应速度**可以分为：硬实时系统、软实时系统

   硬实时系统：关键任务在确定的时间得到响应，不能失败

   软实时系统：偶尔任务的响应时间超出限制也不会有严重后果

7. 音频信号采样：DAC；数字信号转音频信号输出：ADC
8. STM定时器主要参数：计数器值、预分频值
9. FreeRTOS对系统任务的数量没有限制，既支持**优先级调度算法**，又支持**轮转调度算法**
10. FreeRTOS采用双向链表来进行调用
11. 与μC/OS-II相对应的嵌入式操作系统是FreeRTOS
12. μC/OS-II最多支持**64**个任务同时运行。任务编号与优先级范围均为0~63。数字越小优先级越高
13. μC/OS-II内核是针对**实时系统**的要求来设计实现的，为可剥夺(抢占式)实时多任务内核，但是**没有网络功能和文件系统**
14. μC/OS-II任务间通信方法有4种：全局变量、信号量、邮箱、消息队列
15. ESP8266是一个低功耗的WIFI芯片
16. 嵌入式系统的加载启动任务由BootLoader来完成
17. 机器人一般由执行机构、驱动装置、检测装置、控制系统、复杂机械组成
18. 机器人一般由机械本体、控制系统、传感器、驱动器和输入/输出系统接口等五部分组成。
19. 通用定时器TIM三种工作方式：向上计数模式、向下计数模式、中央对齐模式
20. STM32引脚未接外界电路，呈现高阻态模式（伏空输入模式）
21. 减速器由多级齿轮组合而成，级数越高，转速越低，扭矩越大，载荷能力越高
22. STM32芯片内核: ARM Cortex-M3，**最高频率 ？**
23. 接受外部模拟型号，引脚配置：模拟输入

    模拟输出/推挽输出?

24. 初始化PWM，初始引脚配置：复用推挽输出

25. 管脚输出电压3.3V， 配置PWM占空比30%，则有效输出电压为：3.3*0.3=0.99V

---

2. 判断🤔

1. STM32配置串口时，需要将Rx引脚设置为推挽输出模式

   错。R为Recieve, 故应设置为输入

2. 脉冲宽度调制（PWM）改变LED灯亮度通过调节电压高低来实现

   错。是通过调节亮灭时间比来实现的

---

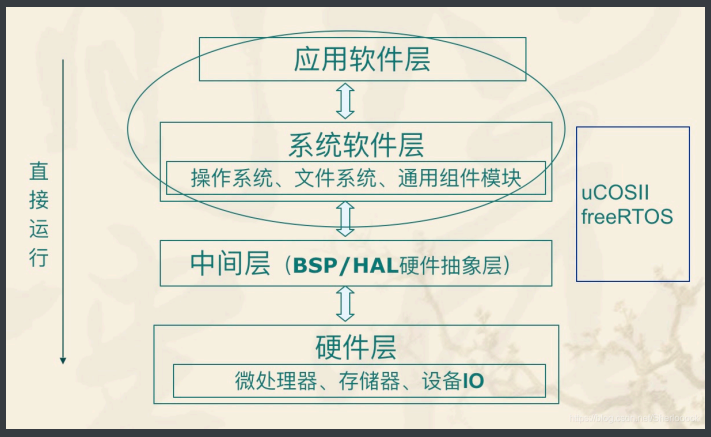
3. 简述

1. 简单的嵌入式应用开发过程
   1. 在个人电脑上配置交叉开发环境、安装仿真器，编写程序
   2. 交叉编译，在仿真器上运行测试并修改直到软件满足需求
   3. 利用面包板开发和调试软、硬件
   4. 利用编程器将软件烧到开发好的系统中进行现场运行测试
2. FreeRTOS特点
   1. 用户可配置内核功能
   2. 多平台的支持（几乎可以在所有ARM体系芯片中运行）
   3. 提供一个高层次的信任代码的完整性
   4. 目标代码小，简单易用
   5. 强大的执行跟踪功能
   6. 堆栈溢出检测
   7. 没有限制的任务数量
   8. 没有限制的任务优先级
   9. 多个任务可以分配相同的优先权
   10. 队列，二进制信号量，计数信号灯和递归通信和同步的任务
   11. 优先级继承
   12. 免费开源的源代码

---

4. 软硬件四层结构（复习幻灯片）

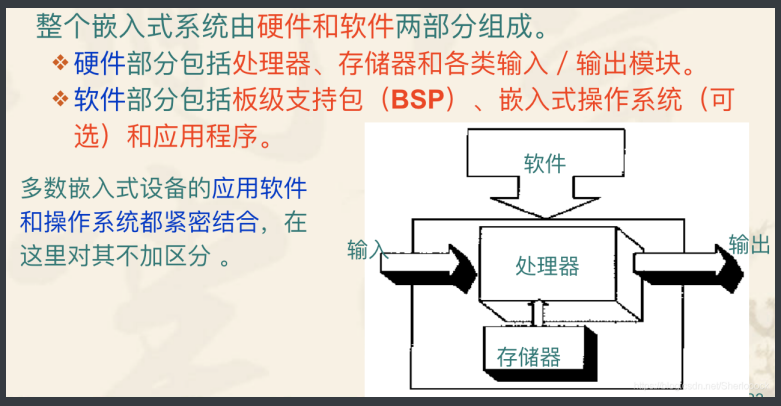
硬件层、中间层、系统软件层、应用软件层



---

5. 嵌入式系统体系结构

硬件组成部分：处理器、存储器、输入输出系统



---

6. 嵌入式处理器四种类型

嵌入式处理器是嵌入式系统**硬件**中的最核心的部分

- MPU （Microprocessor Unit) 微处理器
- MCU （Microprogrammed Control Unit) 微程序控制器
- DSP (Digital Signal Processor) 数字信号处理器
- SoC （System on a Chip) 片上系统

7. 机器人🤖的主要传感器及其特性

1. 光线传感器：感受周围光线明暗程度并转化成电信号
2. 人体红外传感器：探测静止人体的红外热释
3. 红外测距传感器：用红外线为介质测量
4. **红外灰度传感器**（循迹传感器组成成分）：根据接收器收到的红外光多少来判断障碍物的黑白颜色情况（黑色物体吸收光多）
   - **蓝色**的是红外发射管;**黑色**的是红外接收管。二者合称为**红外对管**
   - 三个接线口：
     - GND：接地
     - VCC (Voltage Common Collector)：直流5V电压输入
     - OUT：信号输出（黑色输出低电平，白色输出高电平）
   - 多个红外灰度传感器组成循迹传感器（即巡线传感器）
5. 温度传感器：感受温度并转化成可输出信号

8. RAM/ROM/Cache/寄存器

8.1 RAM ROM及其特性

RAM(Random-Access Memory 内存)：特性断电数据**丢失**，访问速度快，存取短期数据
ROM（Read-Only Memory）：特性断电**不丢失**数据，访问速度较慢，存储长期不能丢失的数据

8.2 寄存器、Cache、Flash速度比较

寄存器 > Cache > Flash（ROM）> 外部存储器

9. 嵌入式Linux（1选择题或填空题）

1. 特点
   - 开源、免费、高效、内核小、功能强、API丰富、系统健壮
   - 支持多种CPU芯片，包括x86 CPU
2. 种类
   - uCLinux
   - ARMLinux

- RT-Linux/RTAI
- Embedix
- Blue Cat Linux
- Hard Hat Linux　*区别于：Red Hat*
- Ubuntu　*区别于：Debian*

---

10. STM32芯片编号的含义

STM32	STM32代表ARM Cortex-M内核的32位微控制器。
F	F代表芯片子系列。
103	103代表增强型系列。
R	R这一项代表引脚数，其中T代表36脚，C代表48脚，R代表64脚，V代表100脚，Z代表144脚，I代表176脚。
C	C这一项代表内嵌Flash容量，其中B代表128K字节Flash，C代表256K字节Flash，D代表384K字节Flash等等。
T	T这一项代表封装，其中H代表BGA封装，T代表LQFP封装，U代表VFQFPN封装。
6	6这一项代表工作温度范围，其中6代表-40——85℃，7代表-40——105℃。

---

11. 引脚🦶

11.1 STM32引脚工作模式

清楚每种工作模式的名字+含义

1. **ADC** ：　Analog to Digital Converter　模数转换器（Imp）
2. **DAC** ：　Digital to Analog Converter　数模转换器（Imp）
3. I2C：　Inter－Integrated Circuit　集成电路总线
4. USART：　Universal Synchronous/Asynchronous Receiver/Transmitter　通用同步/异步串行接受/发送器
5. SPI ：　Serial Peripheral Interface　串行外设接口
6. FSMC：　Flexible Static Memory Controller　可变静态存储控制器
7. SDIO ：　Secure Digital Input and Output Card　安全数字输入输出卡
8. CAN ：　Controller Area Network　控制器域网

11.2 解释USART\ADC\DAC\PWM

1. ADC （Analog to Digital Converter）模数转换器

   将连续的模拟信号转换为离散数学信号的器件

2. DAC （Digital to Analog Converter）数模转换器

   将离散的数字信号转换为连续模拟信号的其器件

3. USART （Universal Synchronous/Asynchronous Receiver/Transmitter） 通用同步/异步串行接收/发送器

   USART是一个全双工通用同步/异步串行收发模块，该接口是一个高度灵活的串行通信设备

4. PWM （Pulse Width Modulation）脉冲宽度调制

   脉冲宽度调制是一种模拟控制方式，其根据相应载荷的变化来调制晶体管基极或MOS管栅极的偏置，来实现晶体管或MOS管导通时间的改变，从而实现开关稳压电源输出的改变

11.3 STM32芯片引脚输入输出模式（了解）

参考

英文名	解释	中文名
GPIO_Mode_AIN	Analog in	模拟输入
GPIO_Mode_IN_FLOATING	input floating	浮空输入
GPIO_Mode_IPD	input pulled down	下拉输入
GPIO_Mode_IPU	input pulled up	上拉输入
GPIO_Mode_Out_OD	output open drain	开漏输出
GPIO_Mode_Out_PP	output push-pull	推挽输出
GPIO_Mode_AF_OD	alternate function open drain	复用开漏输出
GPIO_Mode_AF_PP	alternate function push pull	复用推挽输出

- 模拟输入：接收外接电路的模拟信号
- 浮空输入：接收外接电路的数字信号，未接外接电路时呈现高阻态
- 下拉输入：把电位拉低到GND
- 上拉输入：把电位拉低到VCC
- 开漏输出：IO输出0，为GND电平；IO输出1，为悬空状态，需外接上拉电阻，才能实现输出高电平
- 推挽输出：IO输出0，为GND电平；IO输出1，为VCC电平，读输入值是未知的
- 复用开漏输出：高级功能下的开漏输出模式，如串口输出
- 复用推挽输出：高级功能下的推挽输出模式，如I2C的SDA和SCL输出（I2C：同步串行总线、SDA：串行数据线、SCL：串行时钟线）

11.4 GPIO管脚编程初始化过程

```
void Tracker_Init(void)
{
#if TRACKERSENSOR_ENABLE
    GPIO_InitTypeDef GPIO_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOD, ENABLE);   //使能GPIOD

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5 | GPIO_Pin_6 | GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;   //浮空输入
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOD, &GPIO_InitStructure);
#endif
}
```


1. STM32驱动LED发光二极管
2. 如何设置管脚为推挽输出（记住代码） `GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP`
3. 清楚GPIO_Speed含义：引脚速度。定义输入/输出的最高频率

12. 串口

12.1 STM32中串口编程

串口编程初始化步骤：

1. 设置波特率（波特率115200）

   问：通过串口连接的两台计算机一秒传输的字节数为多少?

   115200/8 byte　115200bit
2. 设置字长
3. 设置停止位（记住代码）
4. 设置奇偶校验位（记住代码）
5. 设置

```
//USART初始化设置

USART_InitStructure.USART_BaudRate = bound;                    //波特率为9600
USART_InitStructure.USART_WordLength =USART_WordLength_8b;     //字长为8位数据
USART_InitStructure.USART_StopBits =USART_StopBits_1;          //1个停止位
USART_InitStructure.USART_Parity =USART_Parity_No;             //无奇偶校验位
USART_InitStructure.USART_HardwareFlowControl =USART_HardwareFlowControl_None;//无硬
件数据流控制
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;//收发模式

USART_Init(USART1, &USART_InitStructure);                      //串口初始化
USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);                 //中断开启
USART_Cmd(USART1, ENABLE);                                     //串口使能
```

12.2 串口通讯中收/发/收发双工模式代码

```
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx; //收发模式
```

---

13. LED灯💡

13.1 设置LED灯亮（高电平）/灭（低电平）/延时

代码

```
GPIO_SetBits(GPIOB, GPIO_Pin_5);      //点亮LED0
GPIO_ResetBits(GPIOE, GPIO_Pin_5);    //关闭LED1
delay_ms(500);                        //延时500ms
```

---

14. 定时器⏲️

14.1 定时器分类及名称

1. 分辨率16位
   - 高级定时器（2个）：TIM1、TIM8，互补输出功能
   - 普通定时器（4个）：TIM2 ~ TIM5，无互补输出功能
   - 基本定时器（2个）：TIM6、TIM7，无捕获/比较通道，无互补输出功能
2. 分辨率24位
   - 系统滴答(systick)定时器（1个）：倒计时定时器。在睡眠模式下也能工作
3. 其它
   - 看门狗定时器（2个）

14.2 初始化定时器的代码

```
//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_Period = arr;
//设置用来作为TIMx时钟频率除数的预分频值  不分频
TIM_TimeBaseStructure.TIM_Prescaler = psc;
//设置时钟分割:TDTS = Tck_tim
TIM_TimeBaseStructure.TIM_ClockDivision = 0;
//TIM向上计数模式
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
//根据TIM_TimeBaseInitStruct中指定的参数初始化TIMx的时间基数单位
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
```


1. 自动重装载寄存器周期值：TIM_Period

   该值用于定时器计数，在根据该值从而产生定时器中断

2. 设置计时器的预分频值：TIM_Prescaler

   该值用于定时器频率分割

问题：定时器编码中如果设置***定时器***经过计数N=10000后进入中断处理函数，在初始化时应设置什么变量的值=9999?

参考：

设置定时器预分频值，用作定时器时钟频率分割。

比如：STM32时钟为72MHz，预分频值为719，分频之后其工作频率为：72 000 000/(719+1)=100 000Hz，即100KHz。

注意：预分频值计算时，需要加1。

频率： $F = \frac{72M}{(arr+1)*(psc+1)}$　　**占空比：** $Dc = \frac{Tim->CCR1}{psc+1}$

72M：STM32时钟频率　　*arr*：自动重装载值

psc：预分频值　　*TIM->CCR1*：动态设置的比较值

14.3 TIM_setcompile含义（第五章定时器编程）

如何用其设置指定占空比的皮达姆波(PWM)

思考：

- 栈空比50% 如何设置?
- 频率50k时，预分频值为多少?

```
void right_motor_forward(u16 pwm){
  GPIO_ResetBits(GPIOB, GPIO_Pin_1);
  TIM_SetCompare3(TIM3, pwm)
}
```

---

15. 电机

15.1 掌握步进电机引脚初始化的代码（第五章PPT）

& 掌握直流电机引脚初始化的代码

1. 指定GPIO口
2. 指定GPIO口工作模式（out pp）
3. 指定GPIO工作频率
4. 初始化GPIO口

```c
void Motor_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure ;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|RCC_APB2Periph_GPIOE, ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;            //设为推挽输出模式
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_14;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;            //设为推挽输出模式
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOE, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;            //设为推挽输出模式
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOE, &GPIO_InitStructure);

    GPIO_ResetBits(GPIOE,GPIO_Pin_14);
    GPIO_ResetBits(GPIOB,GPIO_Pin_1);
    GPIO_SetBits(GPIOE,GPIO_Pin_11);
}
```


---

16. STM32中中断函数初始化的代码

理解此代码中的抢占优先级，响应优先级的设置方法

```c
//中断优先级NVIC设置
NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;  //TIM3中断
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;  //抢占优先级0级
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;  //响应优先级3级
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE; //IRQ通道被使能
NVIC_Init(&NVIC_InitStructure);  //初始化NVIC寄存器
```


> **抢占优先级**和**响应优先级**，其实是一个中断所包含的两个优先级，其中前者是对抢占优先级的级别划分，后者是相同抢占优先级的优先级别的划分。

---

17. STM32中PWM设置输出通道1、2比较值的函数

---

18. STM32中若串口输出数据应如何操作

使用专门的串口输出函数不能用printf

---

19. μC/OS-II中信号量的两个操作

1. OSQ pend （tend?）

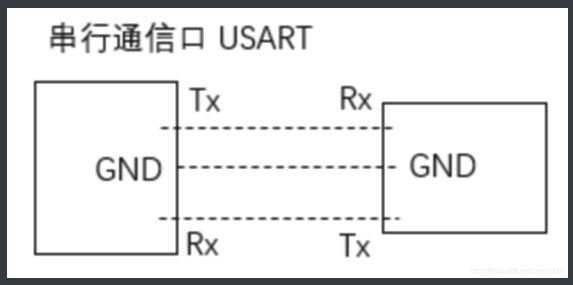
   操作消息的函数

2. OSQ post
   操作消息队列的函数

---

20. 串行通信口(USART)连接方式

单片机和串行通信口之间的通信，设备的Tx接单片机的Rx，设备的Rx接单片机的Tx



21.ST-Link未装驱动或坏了会出现什么情况

无法烧录

复习结束

那么...

与君歌一曲，请君为我倾耳听~

我的其它相关文章

- Python期末考试总复习资料
- Oracle期末考试总复习资料
- 《软件测试与质量保证》期末复习重点
- 离散数学期末复习笔记【精华版】