第十讲 嵌入式系统组成和设计方法

§1 教学安排

§2 定义和分类

§3 系统组成

§4 设计方法

§2.1 嵌入式系统的定义

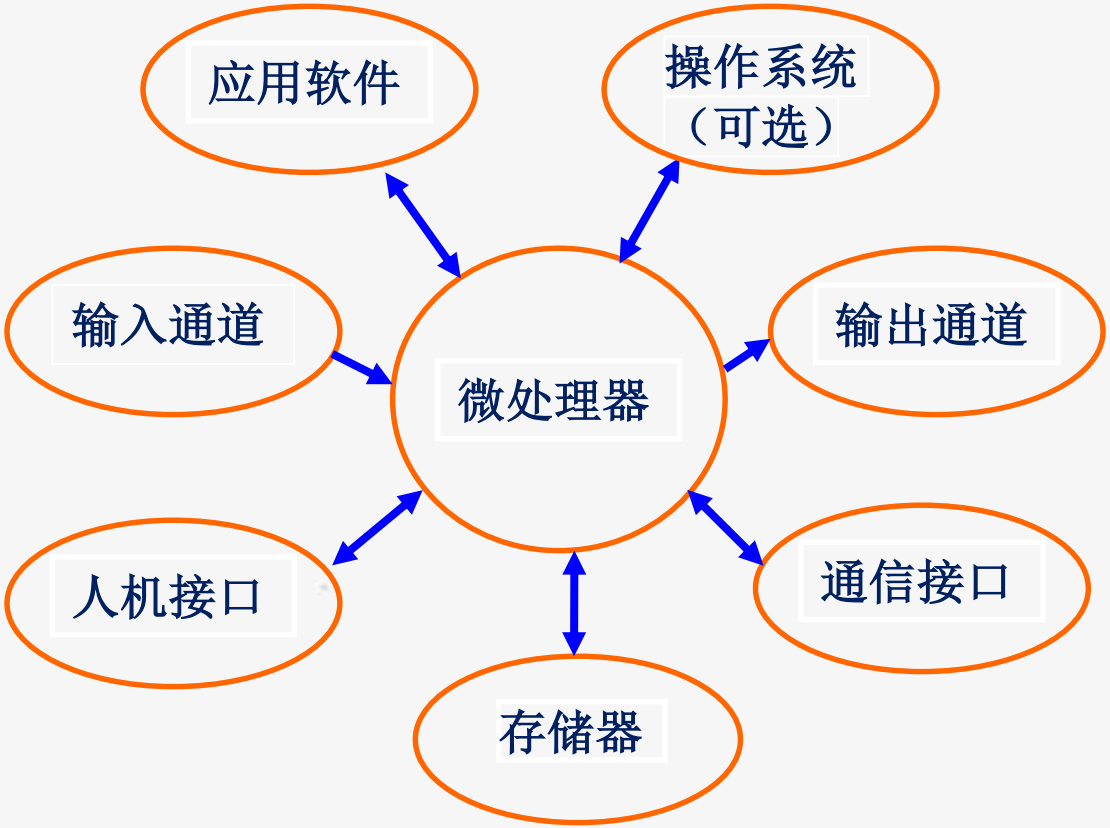
- IEEE(Institute of Electrical and Electronics Engineering国际电气和电子工程师协会)的定义
  device used to control, monitor, or assist the operation of equipment, machinery or plants

  从应用上定义,包括硬件、软件,甚至是机械装置.
  但这种定义没有反映嵌入式系统的基本特征。

- 国内普遍认同的定义

  以应用为中心、以计算机技术为基础，软硬件可裁减，适用于应用系统的、对功能、可靠性、成本、体积、功耗有严格要求的专用计算机系统。

§3 嵌入式系统的组成模型



硬件：微处理器、存储器、过程接口、人机接口、通信接口

软件：应用程序、实时操作系统（可选）

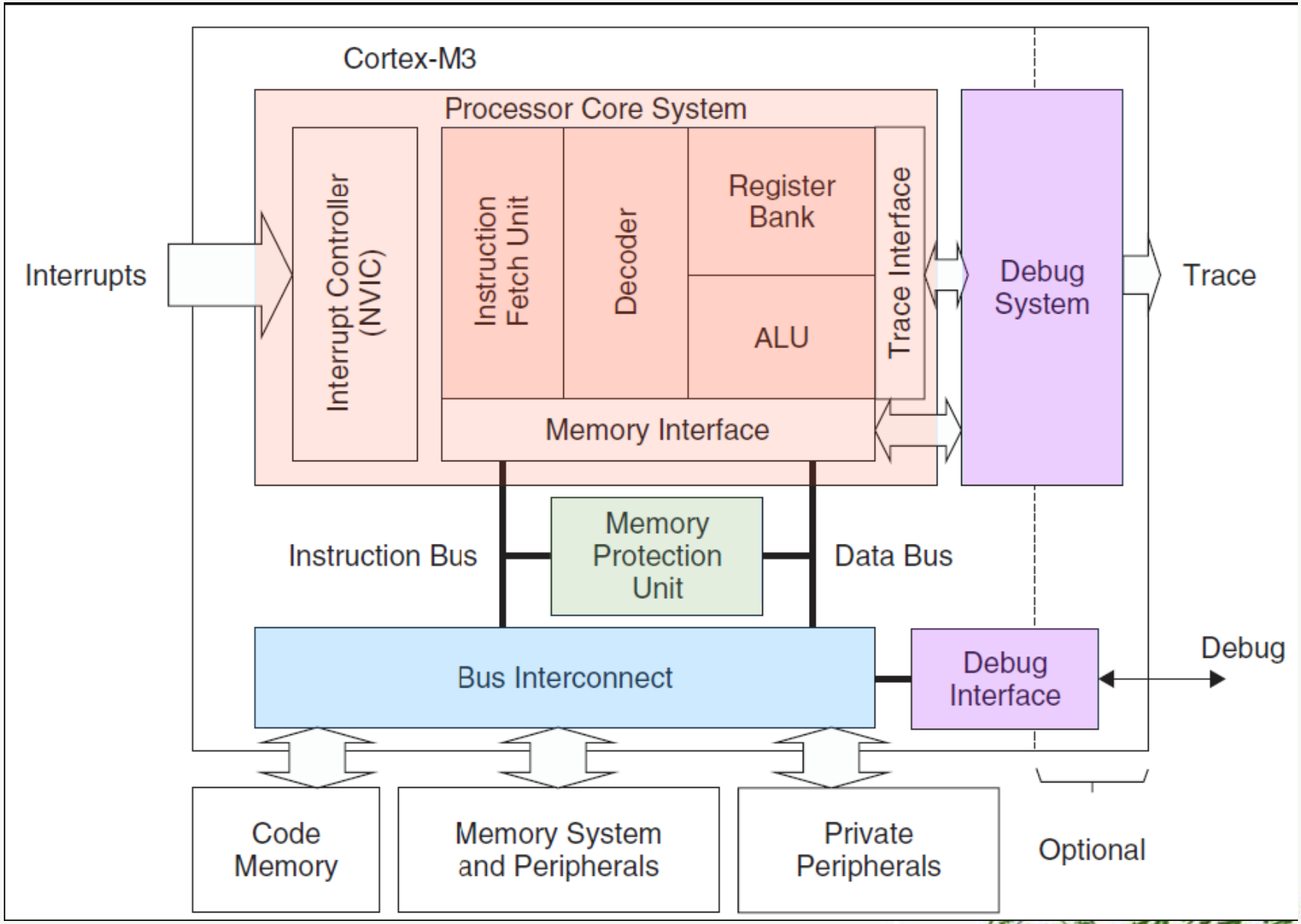
第十讲 Cortex-M3架构和指令系统

§1 ARM公司简介

§2 Cortex-M3内核

§3 指令系统

§4 嵌入式编程

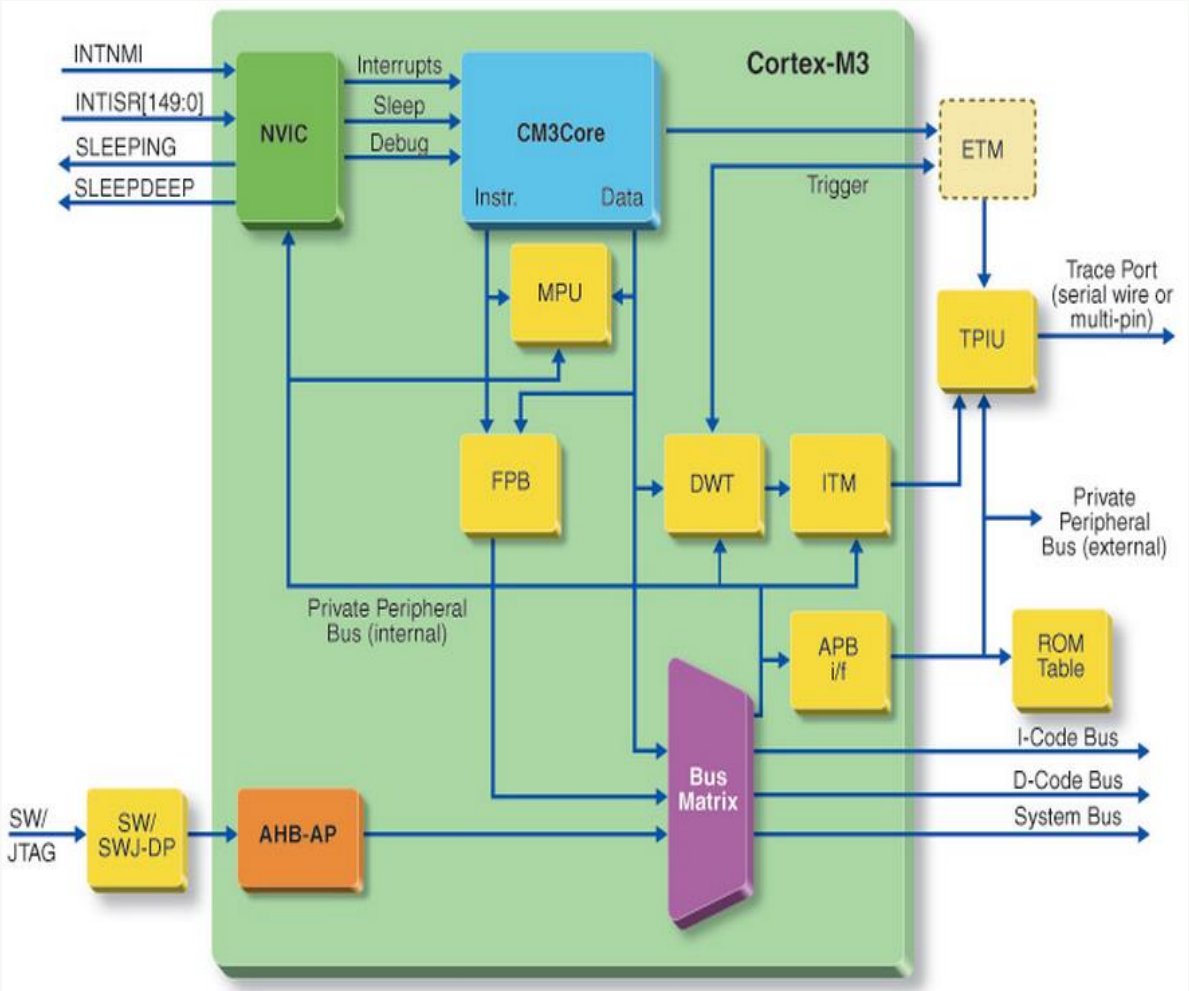


Cortex - M3 的一个简化视图　　来源： STM32权威指南.pdf

Cortex－M3内核方框图

●Cortex-M3处理器主要包括：

1. 处理器内核
2. 与处理器内核紧密结合的嵌套向量中断控制器（NVIC）以实现低延迟的中断处理
3. 存储器保护单元（MPU），可选部件MPU实现存储器保护
4. 总线接口
5. 调试接口



§2.2 三级流水线

CM3处理器使用流水线+分支预测的操作模式，使几个操作同时进行，并使处理和存储器系统连续操作，当出现转移时流水线无需刷新，几乎无损失，增加了处理器指令流的速度。

CM3的流水线分3级，分别为：

取指→译码→执行

来源：zlgmcu ppt

§2.2 三级流水线

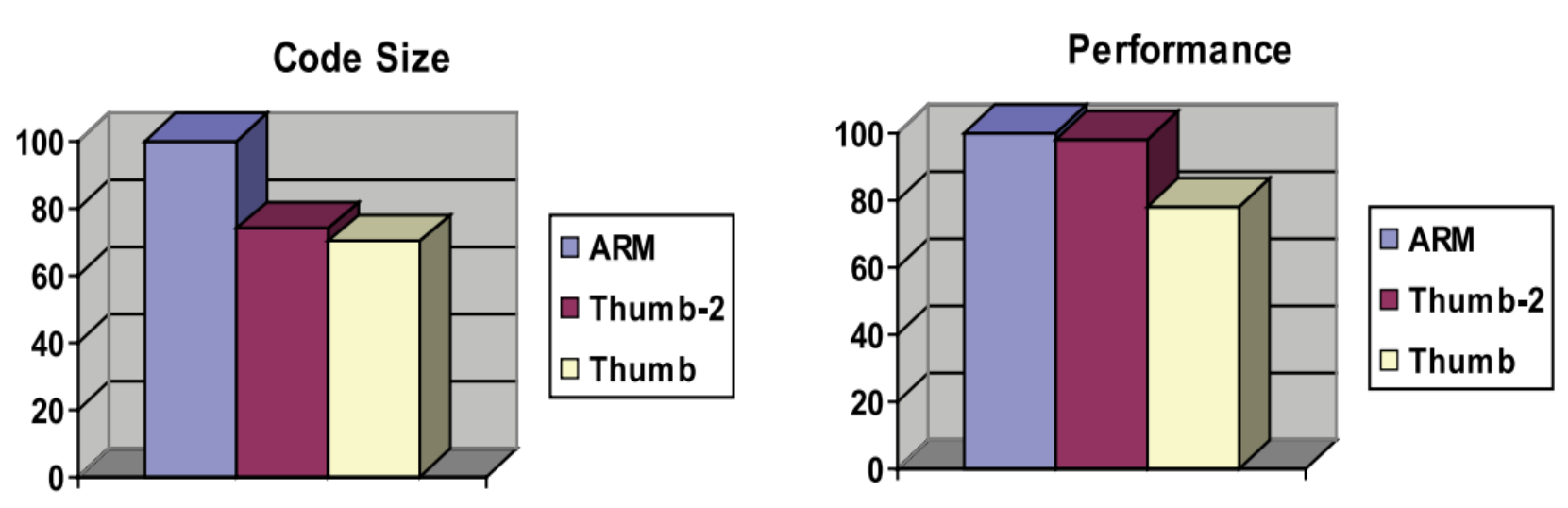
正常操作过程中，在执行一条指令的同时对下一条指令进行译码，并将第三条指令从存储器中取出。这三条指令之间的位置关系如下表所示：

流水线上各指令的地址		流水线工位	描述
ARM指令集	Thumb指令集		
PC	PC	取指	指令从存储器中取出
PC-4	PC-2	译码	对指令使用的寄存器进行译码
PC-8	PC-4	执行	从寄存器组中读出寄存器，执行移位和ALU操作，寄存器被写回到寄存器组中

来源：zlgmcu ppt

Cortex-M3处理器指令集Thumb-2

- ARM Thumb-2技术的代码密度和代码性能



- 引自：ARM公司Richard Phelan撰写的《如何使用Thumb-2 改善代码密度和性能》

Cortex-M3处理器指令集Thumb-2

- Thumb-2技术可以带来很多好处：
  - 可以实现ARM指令的所有功能。
  - 增加了12条新指令，可以改进代码性能和代码密度之间的平衡。
  - 代码性能达到了纯ARM代码性能的98%。
  - 相对ARM代码，Thumb-2代码的大小仅有其74%
  - 代码密度比现有的Thumb指令集更高。
    - 代码大小平均降低5%。
    - 代码速度平均提高2-3%。

注：Cortex-M3不支持ARM指令集。

§2.4 两种工作模式

1）线程模式和处理模式

目的： 引入处理模式（handle mode）和线程模式(thead mode) 的本意，是用于区别普通应用程序（线程模式）和中断服务程序（处理模式）；

效果： 有效的简化了 ARM7体系结构支持7种处理器模式(用户模式、快中断模式、中断模式、管理模式、中止模式、未定义模式和系统模式)。

§2.4 两种工作模式

2）特权级和用户级

目的：用于存储器访问的保护机制。使得普通的用户程序代码不能意外地，甚至是恶意地执行涉及到要害的操作；

特权级：该级别的程序可以访问所有范围的存储器（如果有MPU，还要在MPU规定的禁地之外），并且可以执行所有指令；

用户级：用户级程序不能直接改写CONTROL寄存器，需执行一条系统调用指令(SVC)，由异常服务例程修改CONTROL寄存器，才能在用户级的线程模式下重新进入特权级。

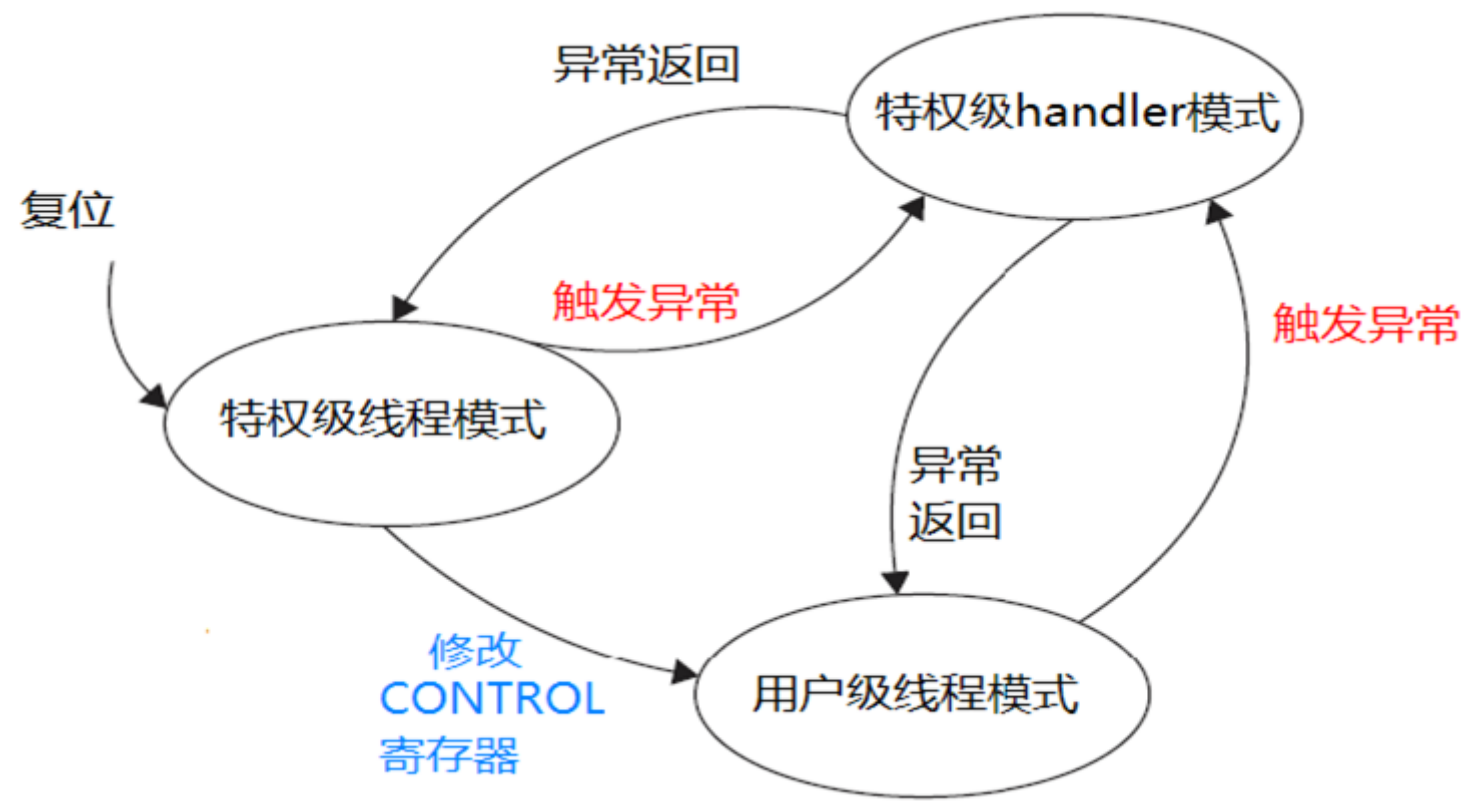
§2.4 两种工作模式

3）默认状态

- 在CM3 运行主程序（后台程序）时是线程模式，既可以使用特权级，也可以使用用户级；
- 中断(异常)服务程序必须在处理模式（始终特权级）下执行；
- 复位后，mcu默认进入线程模式，特权极访问。

§2.4 两种工作模式

4）模式转换

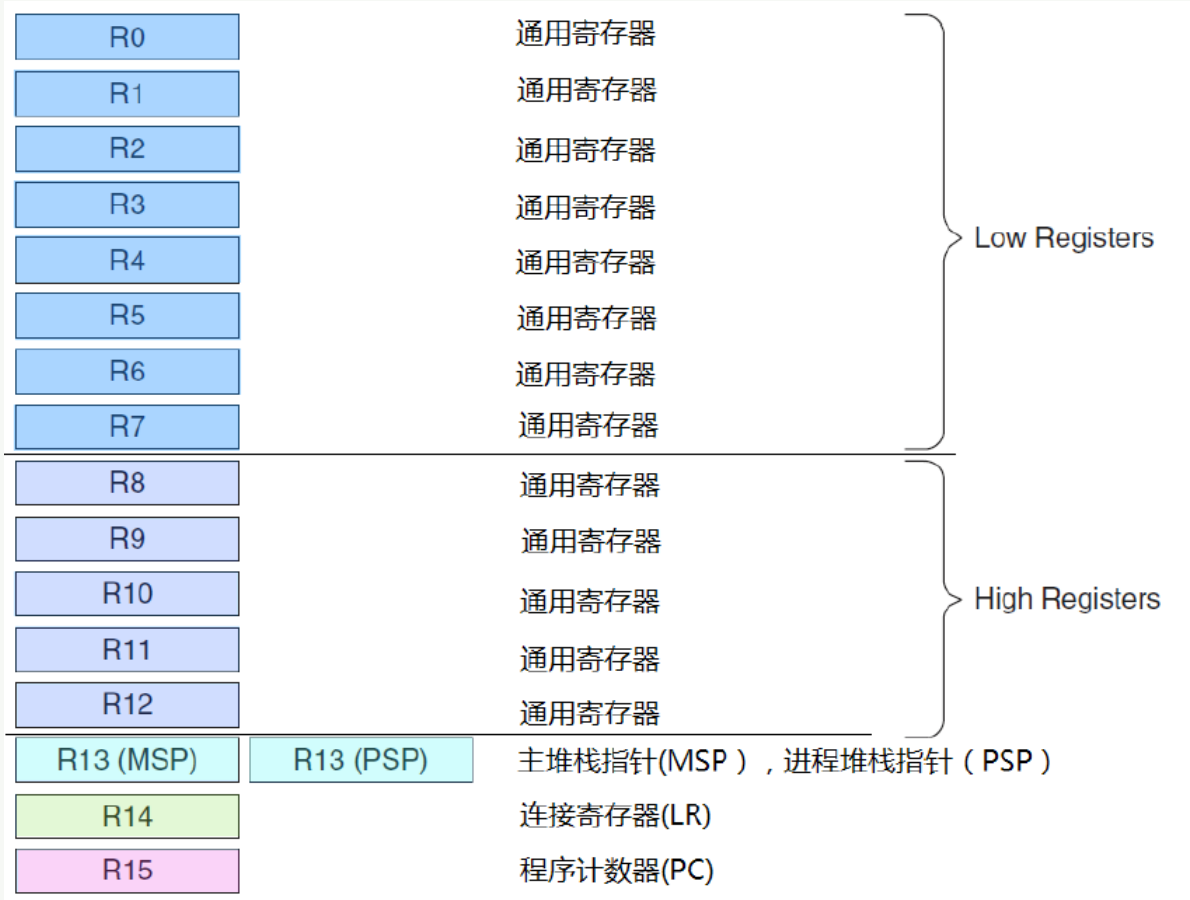


存储器保护单元MPU

- MPU是保护内存的一个组件。 支持标准的 ARMv7(PMSA) “Protected Memory System Architecture ” 模型。
- MPU为以下操作提供完整的支持：
  - 保护区域
  - 重叠保护区区域
  - 访问权限
  - 将存储器属性输出到系统
- MPU可以用于:
  - 强制执行特权原则
  - 分离程序
  - 强制执行访问原则

§2.5 内部寄存器

1）组成：CM3 处理器拥有R0 - R15 的寄存器组。其中R13 作为堆栈指针SP，SP 有两个，但在同一时刻只能有一个可以使用。



§2.6 中断机制

1）重要性

嵌入式系统的实时性通过MCU的中断功能来时实现，因此中断机制和中断部件使用非常重要

2）中断流程

当预定义的事件发生时，正常程序被暂停，处理器进入异常模式。自动将处理器状态保存到堆栈中，执行中断服务程序（ISR），结束时自动从堆栈中恢复，继续执行被暂停的正常程序

3）高效率

嵌套向量中断控制器（NVIC）支持末尾连锁（tail-chaining）中断技术，执行背对背中断（back-to-back interrupt），能够省略连续中断之间的状态保存和恢复指令

§2.6 中断机制

4）其它特点

- 中断优先级可动态重新设置
- 中断数目可配置为1～240
- 中断优先级的数目可配置为1～8位（1～256级）
- 处理模式和线程模式具有独立的堆栈和特权等级
- 使用C/C++标准的调用规范：*ARM架构的过程调用标准（PCSAA）执行ISR控制传输*。

§2.6 中断机制

● 优先级

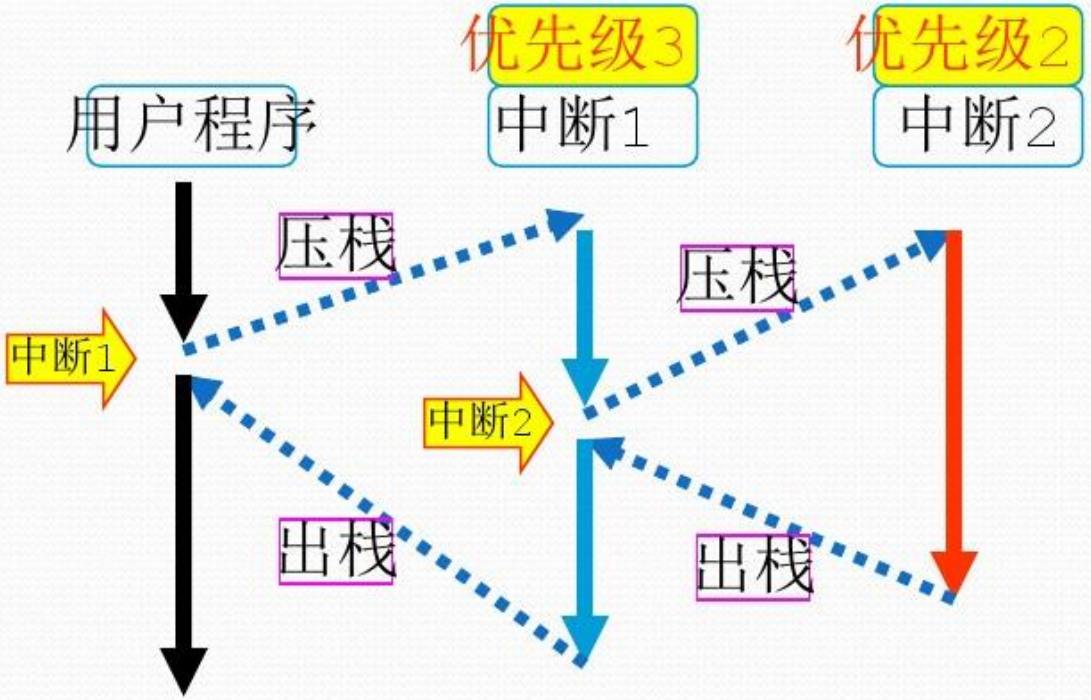
通过对中断优先级寄存器的8 位PRI_N 区执行写操作，将中断的优先级指定为0～255，（0 优先级最高，255 优先级最低）

软件优先级的设置对复位，NMI，和硬故障无效。它们的优先级始终比外部中断要高。

如果两个或更多的中断指定了相同的优先级，则由它们的硬件优先级来决定处理器对它们进行处理时的顺序。

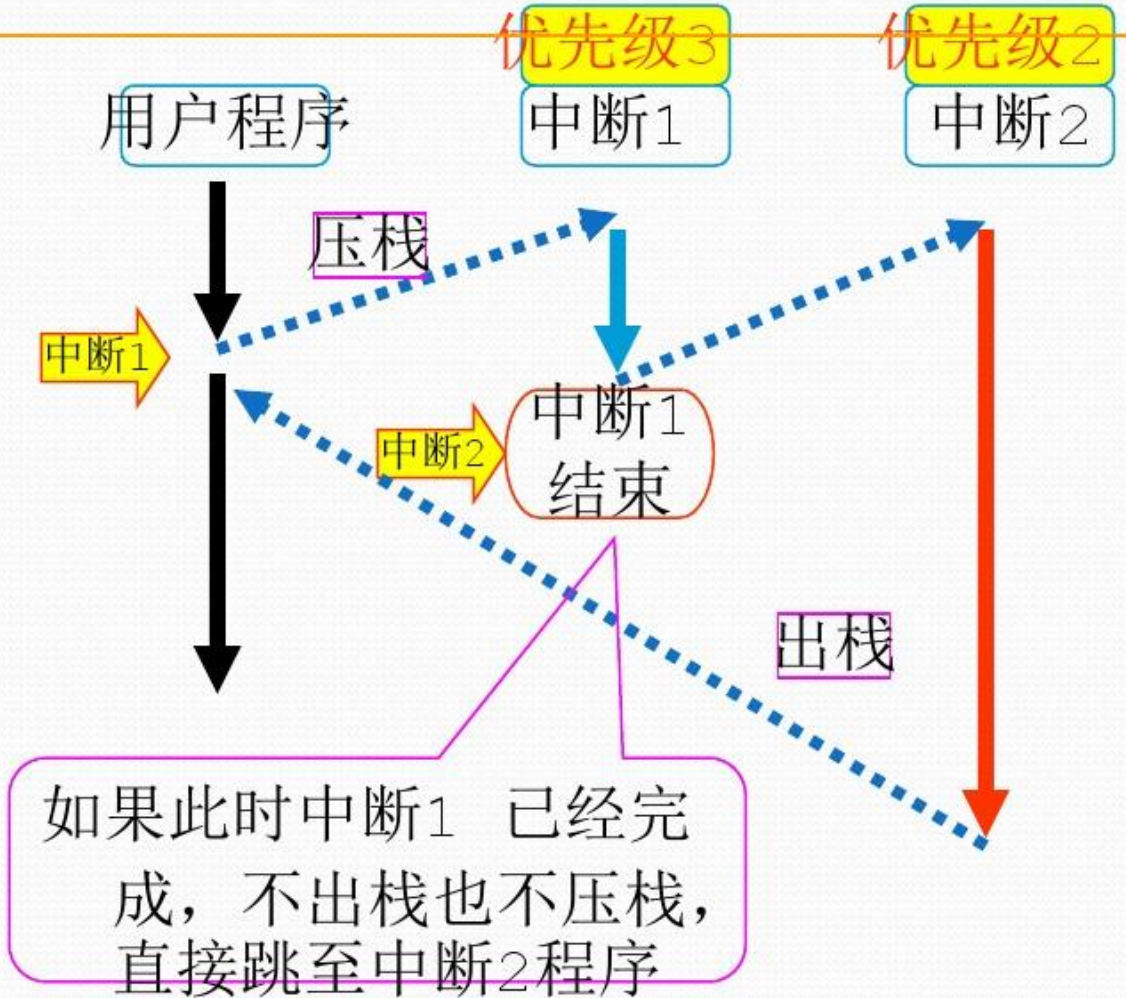
4 占先

在异常处理程序中，一个新的异常比当前的异常优先级更高，处理器打断当前的流程，响应优先级更高的异常，此时产生中断嵌套。



5 末尾连锁

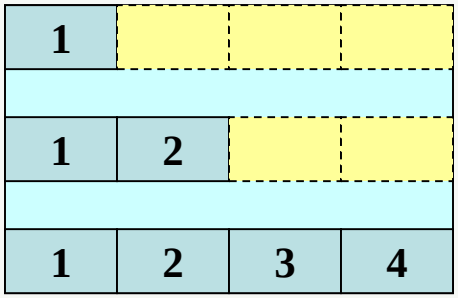
末尾连锁是处理器用来加速中断响应的一种机制。在结束ISR时，如果存在一个挂起中断，其优先级高于正在返回的ISR或线程，那么就会跳过出栈操作，转而将控制权让给新的ISR。



§2.8 存储器及其映射

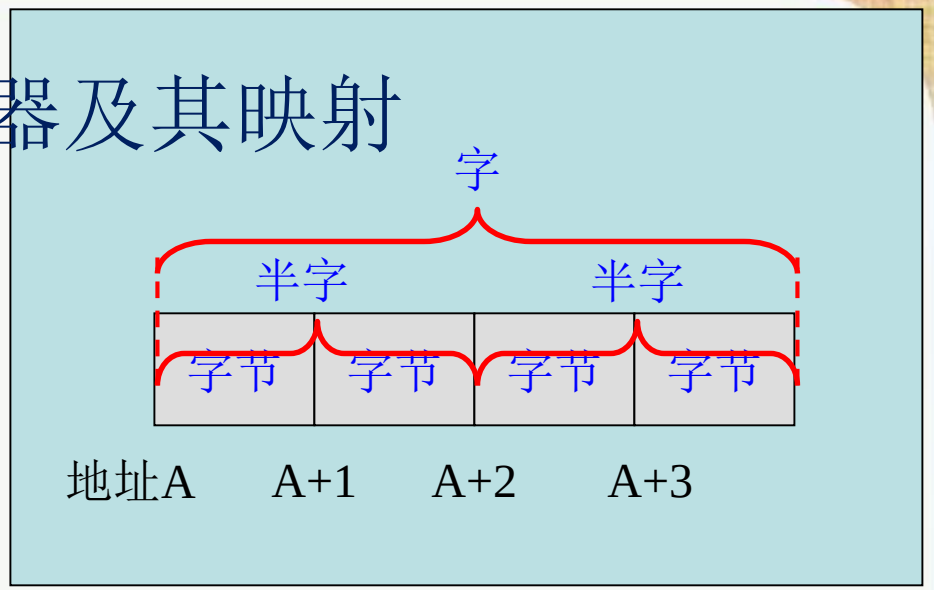
1）硬件直接支持的数据类型

- 字节：8位
- 半字：16位（必须分配为占用两个字节）
- 字：32位（必须分配为占用4各字节）



来源：zlgmcu ppt

§2.8 存储器及其映射



2）存储器格式

- 地址空间的规则：
- 位于地址A的字包含的字节位于地址A,A+1,A+2和A+3；
- 位于地址A的半字包含的字节位于地址A和A+1；
- 位于地址A+2的半字包含的字节位于地址A+2和A+3；
- 位于地址A的字包含的半字位于地址A和A+2；

来源：zlgmcu ppt

§2.9 调试接口

CM3处理器的调试接口有2种：

1) JTAG: 6线制接口

2) SWD: 2线制接口

§4 STM32F103的功能部件

§4.1 GPIO

§4.2 Timer

§4.3 UART

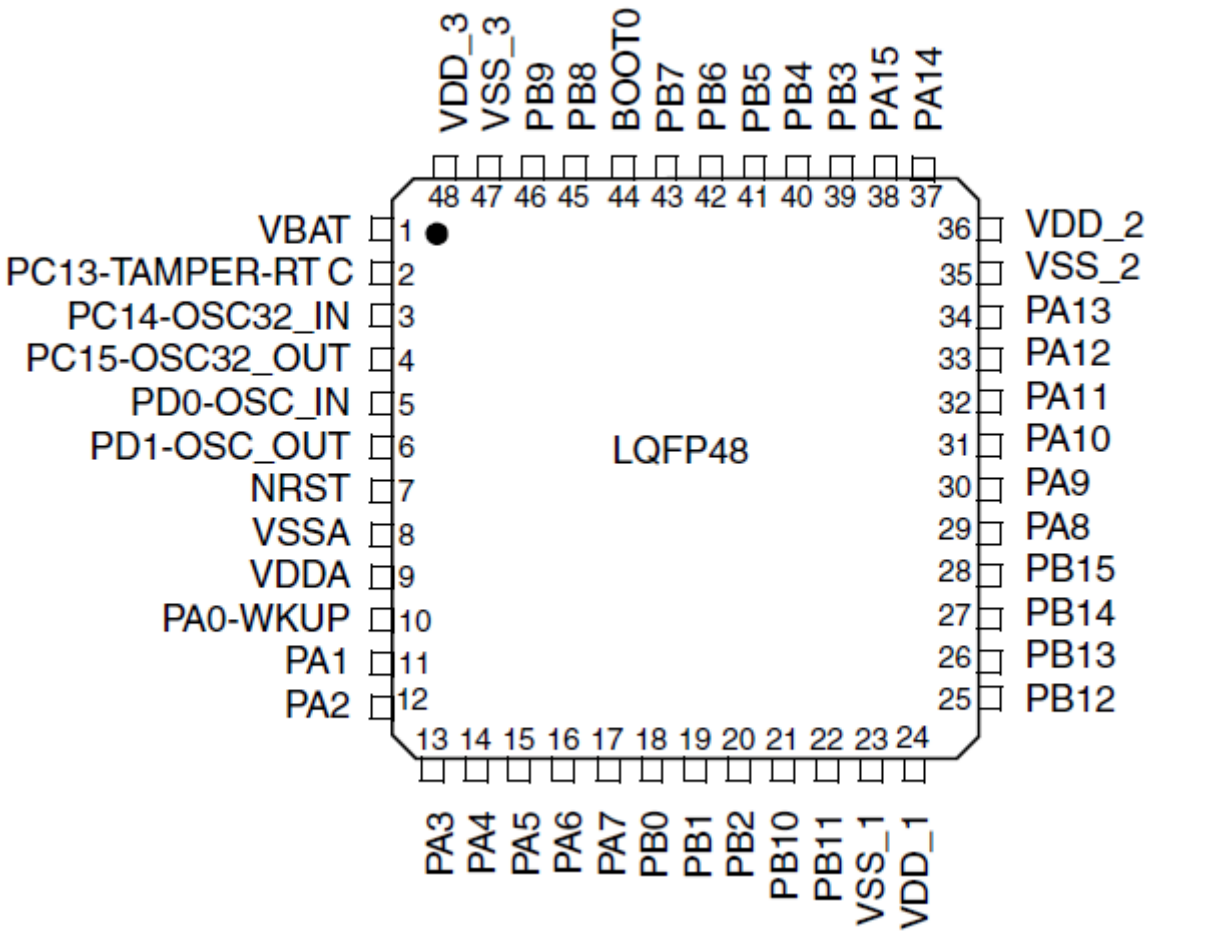
§4.4 A/D

§4.5 D/A

§4.1 GPIO

一、引脚分布

STM32F103C8: 48引脚，PA0~15,PB0~15等



§4.1 GPIO

二、主要功能

① DI 数字量输入：高电平1，低电平0，输入中断、唤醒等；

② DO 数字量输出：高电平1，低电平0；

③ AF 其它功能：引脚复用，预定义的功能模块引脚；

三、工作模式

通过设置GPIO的寄存器来选择：

① Input floating、 Input pull-up、Input-pull-down

② Output open-drain、 Output push-pull

③ Analog

④ Alternate function push-pull、 open-drain

§4.1 GPIO

七、应用例

```
void LED_Init(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure; //描述GPIO寄存器的结构
    GPIO_InitStructure.GPIO_Pin = PIN_LED; //选择LED控制的引脚_2
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //速度_3
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;//模式_4
    RCC_APB2PeriphClockCmd(RCC_LED, ENABLE);  //给引脚加时钟_1
    GPIO_Init(GPIO_LED, &GPIO_InitStructure);  //初始化设置
}
```

.\8--ARM 例程\1 ARM例程\stm32例程\1，LED\流水灯\Users\Src\led.c

§ 4.1 GPIO

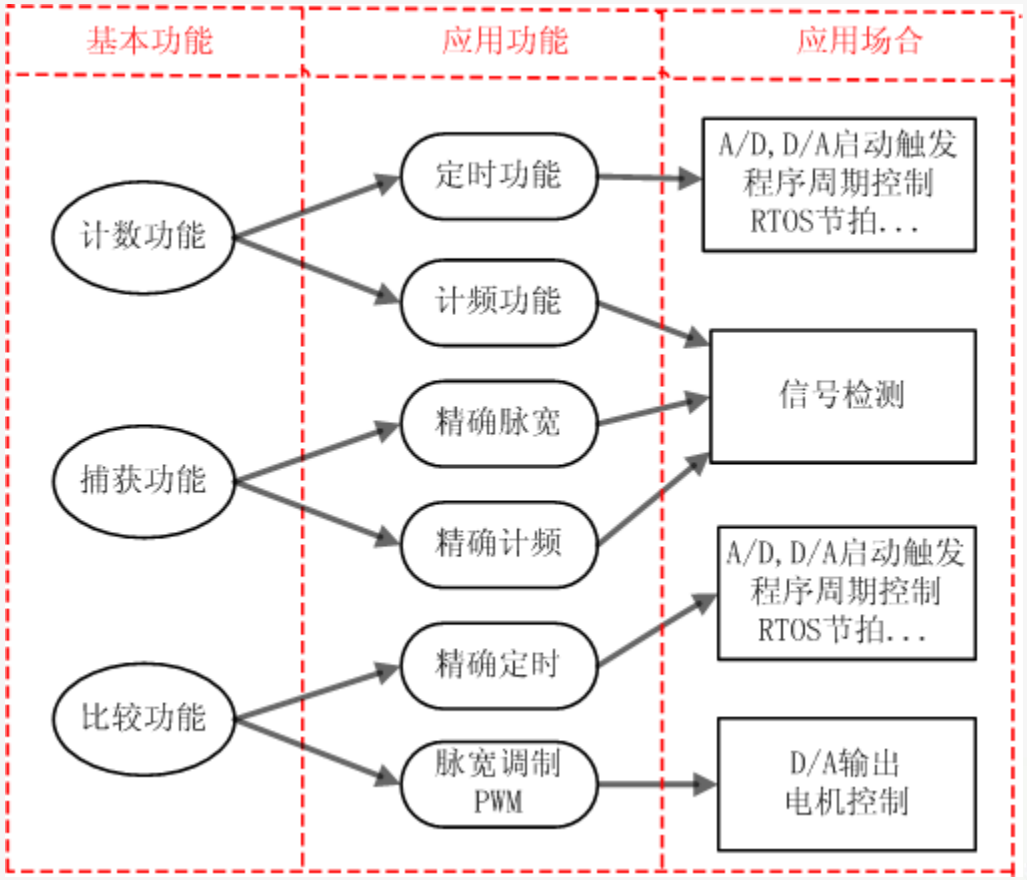
七、应用例

```
void LED_Sets(uint8_t data)
{
        uint16_t  setValue;

        setValue = GPIO_ReadOutputData(GPIO_LED); //调用API
         //((uint16_t)GPIOx->ODR)-->setValue

        setValue &= 0x00ff;
         setValue |= (uint16_t)data << 8;
        //data-->setValue

       GPIO_Write(GPIO_LED, setValue);              //调用API
       //setValue-->((uint16_t)GPIOx->ODR)          //等效的寄存器操作
}
```

.\8--ARM 例程\1 ARM例程\stm32例程\1，LED\流水灯\Users\Src\led.c

§4.2 Timer

一、定时器的功能和应用



二、STM32F103C8的定时器

1）4个通用16位定时器

- 具有4个比较、扑获通道；
- 计数模式：上升、下降、上升和下降；（对比51）
- T1为增强型，带互补输出，紧急停止等功能，可用于电机控制
- 中断产生条件：定时器溢出、比较值相等、扑获引脚有指定的跳变

Table 4. Timer feature comparison

Timer	Counter resolution	Counter type	Prescaler factor	DMA request generation	Capture/compare channels	Complementary outputs
TIM1	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4	Yes
TIM2, TIM3, TIM4	16-bit	Up, down, up/down	Any integer between 1 and 65536	Yes	4	No

四、库函数应用代码例

```
/**-------------------------------------------------------
  * @函数名 TIM5_IRQHandler
  * @功能   TIM5中断处理函数，每秒中断一次
  * @参数   无
  * @返回值 无
***-------------------------------------------------------*/
void TIM5_IRQHandler(void)
{
    /* www.armjishu.com ARM技术论坛 */
    static u32 counter = 0;

    if (TIM_GetITStatus(TIM5, TIM_IT_Update) != RESET)
    {
        TIM_ClearITPendingBit(TIM5, TIM_IT_Update);
        /* LED1指示灯状态取反 */
        SZ_STM32_LEDToggle(LED1);

        /* armjishu.com提心您：不建议在中断中使用Printf，此示例只是演示。 */
        printf("\n\rarmjishu.com提示您：不建议在中断中使用Printf，此示例只是演示。\n\r");
        printf("ARMJISHU.COM-->TIM5:%d\n\r", counter++);
    }
}
```

§4 STM32F103的功能部件

§4.1 GPIO

§4.2 Timer

§4.3 UART

§4.4 A/D

§4.5 D/A

§4.3 USART

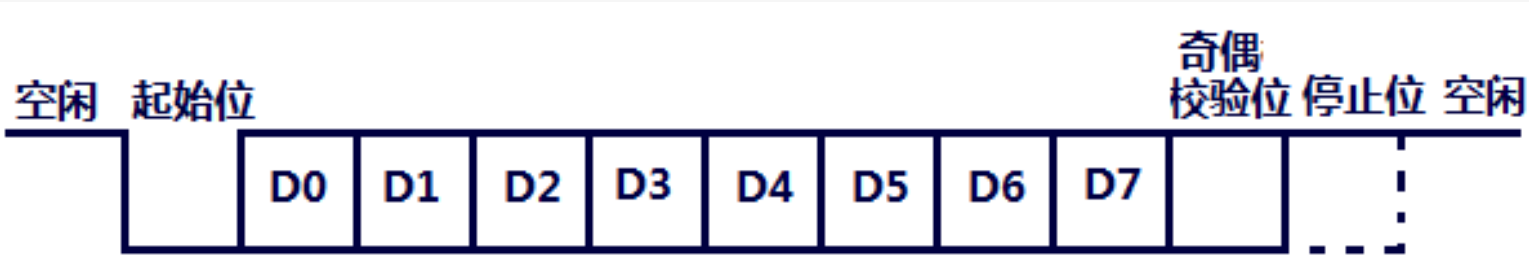
一、USART概述

- USART：Universal synchronous/asynchronous receiver transmitter，通用同步/异步收发器
- UART：通用异步收发器

1）异步串行的字节帧

UART通信是以字节帧为单位的，常用的字节帧：

1个起始位＋8个数据位＋1个校验位＋1个停止位



§4.3 UART

2）通信参数

① 波特率：pbs，每秒多少位，即每一位的时间宽度；

② 数据区长度：8位（常用），或7位

③ 数据区顺序：低位在先（常用），或高位在先

④ 奇偶校验位：无、奇校验、偶校验

⑤ 停止位：1位、1.5位、2位

UART部件能按上述参数，自动完成一个字节帧的接收或发送；在每位宽度的中间连续采样3次，确定该位的数值；异步通信体现在通信速度是事先约定的。

§4.3 UART

三、UART的工作流程

① 接收过程：UART监听总线，有下跳变时，启动数据采样，一个字节的数据收到后，如果奇偶校验正确，则把数据存到接收寄存器中，置状接收态标志位，并向CPU申请中断，让CPU及时读取；

② 发送过程：UART的发送寄存器接收到CPU写入的数据，立刻启动，按设定的参数逐位发送，发送完毕，置状发送态标志位，并向CPU申请中断，告诉CPU可以发送下一个字节。

§4.3 UART

六、程序例

功能：中断接收1个字节，马上发送该字节；

参见：STM32 实验18，串口通信；

```
void USART1_IRQHandler(void)
{
  u8 k;
  if(USART_GetITStatus(USART1,USART_IT_RXNE)!=RESET)//检查指定的USART中断发生与否
  {
    k=USART_ReceiveData(USART1);
    k++;
    USART_SendData(USART1,k);//通过外设USARTx发送单个数据
    //USART_ReceiveData(USART1)返回USARTx最近接收到的数据
    while(USART_GetFlagStatus(USART1,USART_FLAG_TXE)==RESET);
  }
}
```

§4.3 UART

六、程序例

```
void USARTINIT()        //通讯串口的配置
{
  USART_InitTypeDef  USART_InitStructure;

  USART_InitStructure.USART_BaudRate=9600;    //波特率设置为9600
  USART_InitStructure.USART_WordLength=USART_WordLength_8b;
  USART_InitStructure.USART_StopBits=USART_StopBits_1;
  USART_InitStructure.USART_Parity=USART_Parity_No;
  USART_InitStructure.USART_HardwareFlowControl=USART_HardwareFlowControl_None;
  USART_InitStructure.USART_Mode=USART_Mode_Rx|USART_Mode_Tx;

  USART_Init(USART1,&USART_InitStructure);

  USART_Cmd(USART1, ENABLE);
  USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);//使能或者失能指定的USART中断 接收中断
  USART_ClearFlag(USART1,USART_FLAG_TC);//清除USARTx的待处理标志位
}
```

```
int main()
{
  RCCINIT();       //系统时钟的初始化
  GPIOINIT();      // 端口的初始化
  USARTINIT();     // 串口的配置及其初始化
  NVICINIT();     // 中断模式的初始化
  while(1);
}
```

§4 STM32F103的功能部件

§4.1 GPIO

§4.2 Timer

§4.3 UART

§4.4 A/D

§4.5 D/A

§4.4 A/D转换器

一、A/D的概述

- STM32F103xx包含2个12位的逐次比较型ADC；
- 每个ADC有多达16个外部通道；
- ADC时钟是PCLK2经过预分频器得到；
- A/D转换时间 1~1.5us；
- 带有自标定功能，可以减小电路漂移的影响；
- 附加的A/D值比较功能，可实现模拟量超限的自动报警；
- 有多种A/D转换的触发源；
- ADC 工作电压: 2.4 V to 3.6 V；
- 输入信号电压范围: VREF- ≤ VIN ≤ VREF+

§4.4 A/D转换器

二、工作方式

1）单次转换模式：在每个通道上，只执行一次转换；

2）连续转换模式：在每个通道上，执行连续转换；

3）扫描转换模式：在一组选定的模拟输入通道上自动转换；

4）启动A/D转换

软件命令、定时器(TIM1)产生的事件、外部触发和DMA触发；其中外部触发和DMA触发，允许应用程序同步AD转换和时钟的操作；

5）A/D转换结束后自动产生中断；

6）CPU通过查询状态位、中断响应、DMA方式获取A/D值。

§4.4 A/D转换器

六、代码例

```
int main()
{
  u32 ad=0;
  u8 i;
  RCCINIT_PRINTF();     //初始化printf的系统时钟
  RCCINIT_ADC();        //初始化ADC的系统时钟
  GPIOINIT_ADC();       //初始化ADC的端口配置
  GPIOINIT_PRINTF();
  USARTINIT_PRINTF();   //printf串口的初始化配置
  NVICINIT_PRINTF();     //printf中断模式的初始化配置
  ADCINIT_ADC();
  while(1)
  {
    ad=0;
    for(i=0;i<50;i++)//读取50次的AD数值取其平均数较为准确
    {
      ADC_SoftwareStartConvCmd(ADC1, ENABLE);
      while(!ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC));//转换结束标志位
      ad=ad+ADC_GetConversionValue(ADC1);//返回最近一次ADCx规则组的转换结果
    }
    ad=ad/50;
    printf("ad=%f\n",ad*3.3/4096);
    delay_ms(1000);
  }
}
```

§4 STM32F103的功能部件

§4.1 GPIO

§4.2 Timer

§4.3 UART

§4.4 A/D

§4.5 D/A

§4.5 D/A转换器

一、D/A的特点

- STM32F103C8 中无DAC；
- 2个独立的12位D/A，每个D/A有一个电压信号输出端；
- 能产生 三角波、随机噪声波；
- 每个DAC 具有DMA传送能力；
- 带外部触发信号，启动D/A转换；
- 外接参考电压VREF+（与ADC共用）输入，可提高DAC分辨率；
- 输出电压计算： Vout = DA /4096* (VREF+ — VREF-)

§4.5 D/A转换器

二、工作模式

- 单通道模式和双通道模式；
- 双通道模式中，2个DAC可以独立运行，或同时转换；
- 每个DAC可选择8位或12位转换模式；
- 在12位转换模式中，数据可选择左对齐，或右对齐。

§4.5 D/A转换器

六、程序例

```
int main()
{
  u8 i;
  float da;
  RCCINIT();
  GPIOINIT();
  NVICINIT();
  USARTINIT();
  DACINIT();
  while(1)
  {
    da=0;
    for(i=0;i<=10;i++)
    {
      da=i*400;
      //12位 右对齐 PA4 端口输出
      DAC_SetChannel1Data(DAC_Align_12b_R,da);
      printf("da=%fv\n",3.3*da/4096);
      delayms(1000);
      delayms(1000);
      delayms(1000);
      delayms(1000);
      delayms(1000);//间隔5秒输出一个电压
    }
  }
}
```