

RM学习

[stm32入门视频](#)

1.控制器：大脑

1.1MCU（微控制器）

STM32单片机

1.2外设：协助MCU对数据和信息进行处理

外设需要根据控制器实现的功能进行选择，搭建不同的电路


1.3MCU的资源


1.3.1GPIO通用输入输出

1.3.2定时器

晶体振荡器（晶振）：与芯片内部的电路相配合，以固定的频率产生信号。

2.嵌入式系统硬件

 + 关注



嵌入式系统硬件


嵌入式处理器


The diagram shows '嵌入式处理器' (Embedded Processor) at the top, branching into four categories: '微处理器 MPU' (Microprocessor MPU), '微控制器 MCU' (Microcontroller MCU), '数字信号处理器 DSP' (Digital Signal Processor DSP), and '片上系统 SOC' (System on Chip SOC). Each category has a corresponding callout box explaining its characteristics and examples.


- 微处理器 MPU**: 由通用CPU演变具备MMU 如386EX和MIPS
- 微控制器 MCU**: 将CPU、存储器、中断系统和外设集成在单个芯片 如8051/MSP430/STM32
- 数字信号处理器 DSP**: 侧重于信号处理 如TI公司的C2000/C5000系列
- 片上系统 SOC**: 可编程逻辑器件 客户定制


01:58 / 08:36


720P 高清 1.5x
















 + 关注




嵌入式系统硬件


微控制器


The diagram shows a microcontroller chip with its internal components listed: '微控制器' (Microcontroller), '中央处理器' (Central Processor), '存储器' (Memory), '中断系统' (Interrupt System), and '片内外设' (On-chip and off-chip peripherals). To the right, a red box labeled '微控制器' (Microcontroller) is connected to a red box labeled '计算机主机' (Computer Host) by a red double-headed arrow.


01:58 / 08:36


720P 高清 1.5x














ARM处理器

传统软件框架

3.步骤

1.目标选择

2.引脚分配

3.外设配置

3.1时钟模式RCC

- Disable:内部时钟
- BYPASS:旁路时钟
- Crystal:晶振/陶瓷振荡器

3.2调试接口配置SYS

- SW:串行调试接口
- JTAG

3.3配置GPIO

HAL库外设模块的设置方法

(定时器为例)

1.简单外设设计

简单外设设计

简单外设的设计方法

简单外设

GPIO外设使用引脚初始化数据类型GPIO_InitTypeDef来描述GPIO引脚的属性：引脚编号、工作模式、输出速度等

复杂外设

定时器外设具有三类功能：

- 定时/计数功能
- 输出比较功能
- 输入捕获功能

每一类功能都需要单独的初始化数据类型

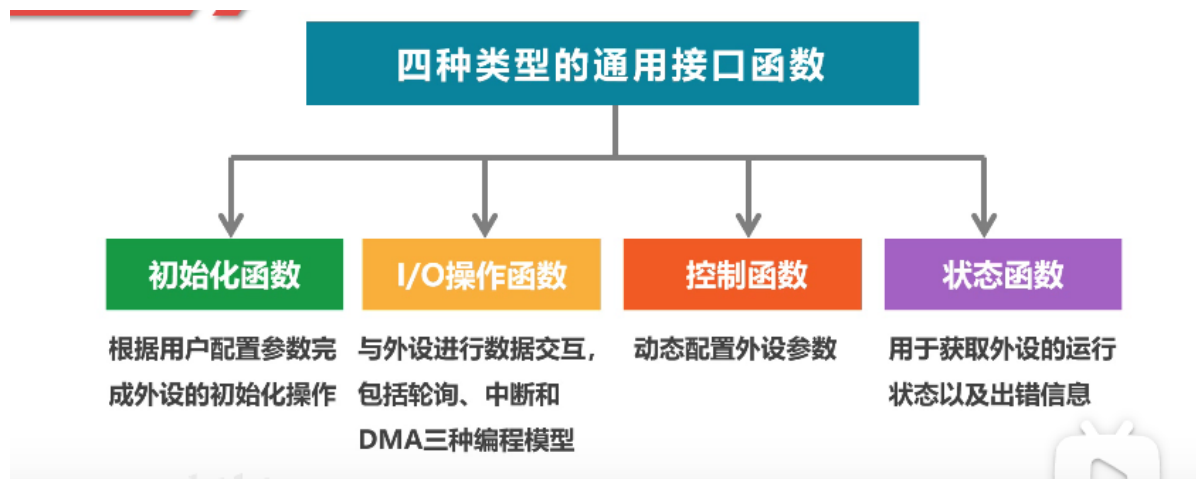
复杂外设设计

2.外设句柄

3.句柄结构组成（定时器句柄结构的组成）

4.三种外设编程模型

5.通用接口函数



6.扩展接口函数

4.时钟设置 (时钟树)

4.1时钟源模块

- (1) 外部低速时钟LSE
- (2) 内部高速时钟LSI
- (3) 内部高速时钟HSL
- (4) 外部高速时钟HSE

4.2配置

修改时钟源频率

选择锁相环输入时钟

选择系统时钟源

设置HCLK时钟频率 (最高100)

5.工程配置 (Project Manager)

5.1project

名称, 路径, 环境

5.2code generator

库设置, 生成文件, hal设置, 模板设置

5.3advanced setting

选择HAL

6.MDK中写程序

添加的代码应位于 `USER CODE BEGIN 3` 后

接口函数：HAL_TogglePin

| | |
|-------|---|
| 函数原型 | void HAL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin) |
| 功能描述 | 翻转引脚电平状态 |
| 入口参数1 | GPIOx: 引脚端口号, 范围是 GPIOA ~ GPIOK |
| 入口参数2 | GPIO_Pin: 引脚号, 范围是 GPIO_PIN0 ~ GPIO_PIN15 |
| 返回值 | 无 |
| 注意事项 | 无 |

指示灯控制引脚PA5对应的端口号为GPIOA, 引脚号为GPIO_PIN5
函数调用形式: HAL_GPIO_TogglePin(GPIOA,GPIO_PIN5)

接口函数：HAL_Delay

| | |
|------|-----------------------------------|
| 函数原型 | void HAL_Delay (uint32_t Delay) |
| 功能描述 | 延时函数, 提供以ms单位的延时 |
| 入口参数 | Delay: 需要的延时时间, 以ms为单位 |
| 返回值 | 无 |
| 注意事项 | 延时功能利用系统节拍定时器SysTick实现, 将占用该定时器资源 |

7.工程设置

仿真器设置(Debug)

reset and run

8.工程文件

8.1生成工程文件的内容

8.2工程框架

MDK-ARM:启动代码文件

User: 用户编辑

HAL_Driver:HAL库驱动文件

CMSIS: 系统初始化文件

4.MDK的调试

1.常规的调试方法

跟踪调试, 单步调试。

2.断点调试

3.观察窗口 （用于查看变量的值

4.内核外设，片内外设查看

5.GPIO

1.端口和引脚

端口PORT：独立的外设子模块，包括多个引脚，通过多个寄存器设置引脚

引脚PIN：对应微控制器的一个管脚，归属于端口，由端口寄存器对应位置控制

2.GPIO模块

电路结构

中断

1.中断相关

(1) 数据传输方式

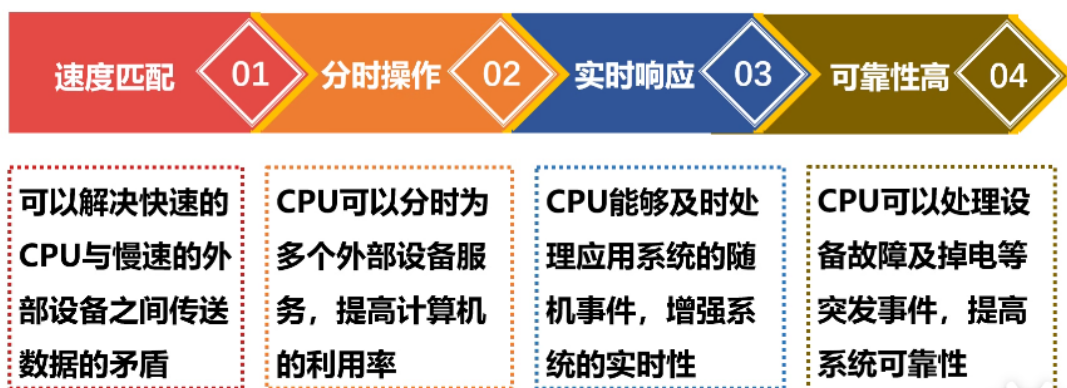
- 无条件传输
- 查询方式
- 中断方式：一方申请中断和另一方传输
- 直接存储器访问

(2) 中断：CPU在处理时间A时发生了时间B而去处理

中断发生，处理，返回

(3) 中断的作用

中断的作用



(4) 中断的优先级（NVIC中有一个八位的中断优先级寄存器）

中断嵌套

高级中端可以打断低级中断；低级不可以打断高级中断

(5) 中断向量

- 中断向量：中断服务程序在内存中的入口
- 中断向量表：把系统中的中断向量集中起来（编号）放到存储器的某一区域叫向量表

(6) NVIC

“内核组件，管理所有中断和异常，为中断源分配中断通道”

中断通道：单个外设具备若干个可以引起中断的中断源，所有中断只能通过指定的中断通道向内核申请中断

HAL_Init将优先级优先分组设置为第四组

2.中断程序

1.编程步骤：

- 设置中断触发条件CubeMx
- 设置中断优先等级CubeMx
- 设能外设中断Cube
- 清楚中断概念Hal库接口函数
- 编写中断程序服务Hal库接口函数

2.HAL封装处理

PPP代表外设名称

一：统一规定处理各个外设的中断服务程序HAL_PPP_IRQHandler

二：在中断服务程序HAL_PPP_IRQHandler完成了中断标志的判断和清除

由外设初始化、中断、处理完成/出错触发的函数

三：将中断中需要执行的操作以回调函数的形式提供给用户

3.接口函数

1 外部中断通用处理函数

| 接口函数：HAL_GPIO_EXTI_IRQHandler | |
|-------------------------------|--|
| 函数原型 | void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin) |
| 功能描述 | 作为所有外部中断发生后的通用处理函数 |
| 入口参数 | GPIO_Pin: 连接到对应外部中断线的引脚，范围是 GPIO_PIN_0 ~ GPIO_PIN_15 |
| 返回值 | 无 |
| 注意事项 | <div>1. 所有外部中断服务程序均调用该函数完成中断处理</div> <div>2. 函数内部根据GPIO_Pin的取值判断中断源，并清除对应外部中断线的中断标志</div> <div>3. 函数内部调用外部中断回调函数HAL_GPIO_EXTI_Callback完成实际的处理任务</div> <div>4. 该函数由CubeMX自动生成</div> |

2 外部中断回调函数

| 接口函数：HAL_GPIO_EXTI_Callback | |
|-----------------------------|--|
| 函数原型 | void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) |
| 功能描述 | 外部中断回调函数，用于处理具体的中断任务 |
| 入口参数 | GPIO_Pin: 连接到对应外部中断线的引脚，范围是 GPIO_PIN_0 ~ GPIO_PIN_15 |
| 返回值 | 无 |
| 注意事项 | <div>1. 该函数由外部中断通用处理函数HAL_GPIO_EXTI_IRQHandler调用，完成所有外部中断的任务处理</div> <div>2. 函数内部根据GPIO_Pin的取值判断中断源，然后执行对应的中断任务</div> <div>3. 该函数由用户根据实际需求编写</div> |

3.计算机通讯

1.通信分为

1.1串行通讯

同步串行和异步串行

异步串行的两个关键点：

1.字符格式(数据的传输形式) 2.波特率：每一位数据的持续时间

1.2并行通信

2.通信速率

单位：波特率 bit/s (bps)

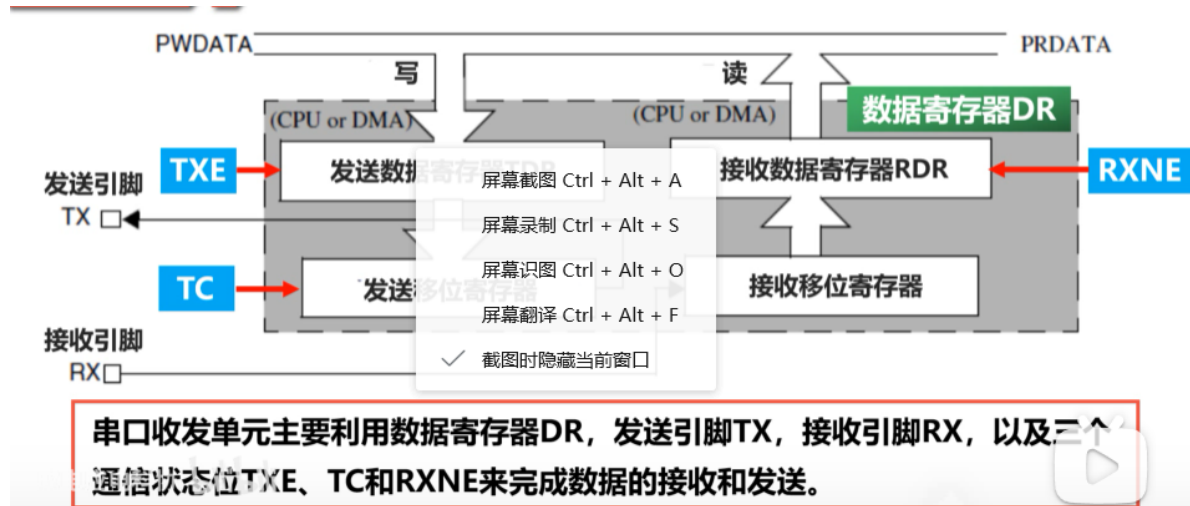
(每秒传送二进制码的位数)

3.串口通讯的数据传输方向

- 单工：一个方向一个线路
- 半双工：两个方向，分时，一条线路
- 全双工：同时双向，双线

4.STM32的串口通信

4.1串口收发单元



4.2串口通信 (F411)

STM32F411芯片的UART引脚

| 串口号 | TX引脚 | RX引脚 |
|-------|--------------|--------------|
| UART1 | PA9/PA15/PB6 | PA10/PB3/PB7 |
| UART2 | PA2 | PA3 |
| UART6 | PA11/PC6 | PA12/PC7 |

4.3串口初始化

5.DMA（直接存储器访问）

1.DMA概述

1.1四个要素

- 传输原：DMA数据传输的来源
- 传输目标：DMA传输数据的目的
- 传输数量：DMA传输数据的数量
- 触发信号：启动一次DMA传输数据的动作

2.

数据流用于连接传输原和传输目标的数据通路

每个数据流可以配置为不同的传输原和传输目标，这些传输原和传输目标成为通道

例如：F411有两个控制器，一个控制器有八个数据流，每个数据流可以映射到8个通道

DMA有16字节的**FIFO**使用其后源数据先传如FIFO，达到触发阈值后在传送到目标地址

3.DMA传输数据的方式

4， 接口函数

| 2 串口DMA方式接收函数：HAL_UART_Receive_DMA | |
|------------------------------------|---|
| 函数原型 | HAL_StatusTypeDef HAL_UART_Receive_DMA (UART_HandleTypeDef *huart,uint8_t *pData, uint16_t Size) |
| 功能描述 | 在DMA方式下接收一定数量的数据 |
| 入口参数1 | huart: 串口句柄的地址 |
| 入口参数2 | pData: 待接收数据的首地址 |
| 入口参数3 | Size: 待接收数据的个数 |
| 返回值 | HAL状态值: HAL_OK表示接收成功; HAL_ERROR表示参数错误; HAL_BUSY表示串口被占用 |
| 注意事项 | 1. 该函数将启动DMA方式的串口数据接收 2. 完成指定数量的数据接收后，可以触发DMA中断，在中断中将调用接收中断回调函数HAL_UART_RxCpltCallback进行后续处理 3. 该函数由用户调用。 |

| UART | |
|-------------------------------------|---|
| 3 获取未传输数据个数函数：__HAL_DMA_GET_COUNTER | |
| 接口函数：__HAL_DMA_GET_COUNTER | |
| 函数原型 | __HAL_DMA_GET_COUNTER(__HANDLE__) |
| 功能描述 | 获取DMA数据流中未传输数据的个数 |
| 参数 | __HANDLE__: 串口句柄的地址 |
| 返回值 | 无 |
| 注意事项 | 1. 该函数是宏函数，进行宏替换，不发生函数调用 2. 函数需要由用户调用，用于获取未传输数据的个数 |

5.空闲中断的特点

- 一帧数据传输结束后，通信线路将会维持高电平，此状态称为空闲状态
- 当CPU检测到通信线路处于空闲状态时，空闲状态标志IDLE将由硬件置一。若串口控制寄存器CR1中IDLEIE位为一，将会触发空闲中断
- 空闲标志实在一帧数据完成后才置位，在有效数据传输过程中不会置位，因此借助空闲中断，可以实现不定长数据的收发

串口通信

1.阻塞时发送

当我们要发送的数据传输完了，函数才会执行结束

2.中断方式

每发送完一个字节就会进入一次中断，来触发下一个字节的发送

中断方式：外设向cpu发出中断请求，cpu响应中断后进行数据传输。但是如果传输较多数据的情况下，cpu得一直花费时间在中断上，也会造成cpu利用率低。

3.DMA

6.定时器

1.概念

- 定时器（计数器的一种特例）：对周期固定的脉冲信号进行技术
- 计数器：对周期不确定的脉冲信号进行技术

2.关注的问题

1.位宽：定时器的计数范围

2.计数值：初值终值的设置

3.处理

时钟频率：定时器模式下，送入定时器的周期性时钟信号频率（一个计数值的时间）

技术周期：1/时钟频率

3.

F411定时器详细特性

| 类型 | 名称 | 位数 | 计数方式 | 预分频系数 | 产生DMA请求 | 捕获/比较通道 | 互补输出 | 时钟频率 | 挂接总线 |
|-------|-----------------|----|------------|---------|---------|---------|------|------|------|
| 高级定时器 | TIM1 | 16 | 递增/递减/中心对齐 | 1~65536 | 是 | 4 | 支持 | 100M | APB2 |
| 通用定时器 | TIM2/ TIM5 | 32 | 递增/递减/中心对齐 | 1~65536 | 是 | 4 | 不支持 | 100M | APB1 |
| | TIM3/ TIM4 | 16 | 递增/递减/中心对齐 | 1~65536 | 是 | 4 | 不支持 | 100M | APB1 |
| | TIM9 | 16 | 递增 | 1~65536 | 否 | 2 | 不支持 | 100M | APB2 |
| | TIM10/ TIM11 | 16 | 递增 | 1~65536 | 否 | 1 | 不支持 | 100M | APB2 |

4. 定时器的功能

1. 定时计数功能

- 定时器模式：计数内部时钟
- 计数器模式：计数外部脉冲

2. 输出比较

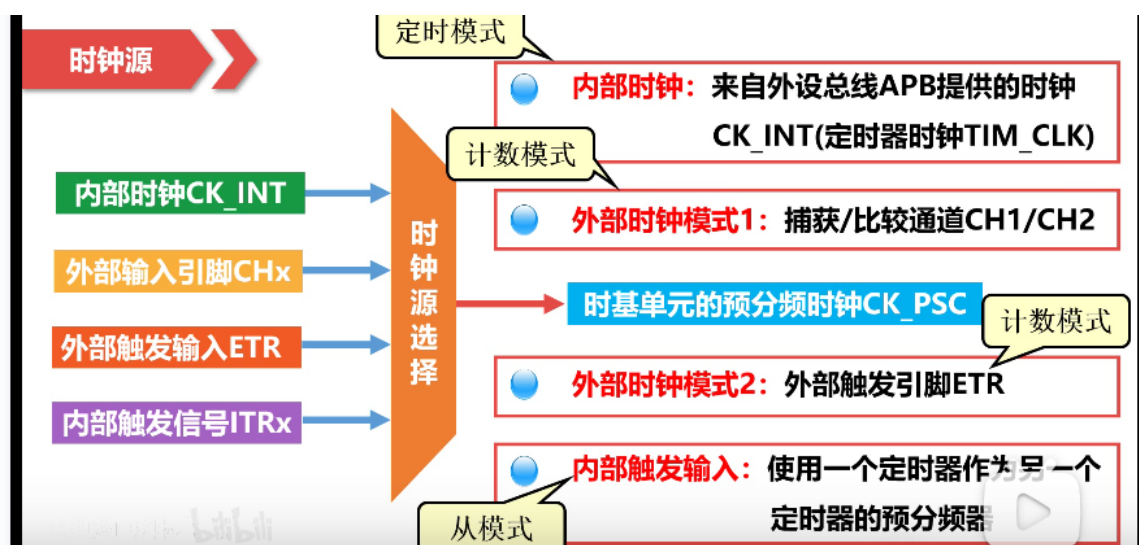
- PWM输出
- 电平翻转
- 单脉冲输出
- 强制输出

3. 输入捕获

- 捕获时保存计时器当前计数值
- 捕获时，可选择触发捕获中断
- 触发捕获的信号边沿类型可选择

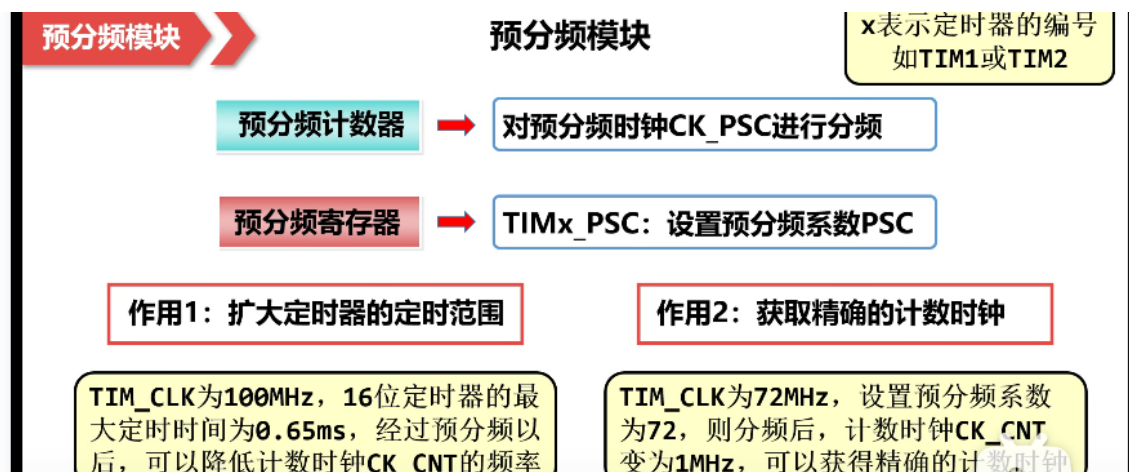
5. 时基单元

1. 送入时基单元的时钟源



2. 时基单元的三个模块

2.1 预分频模块

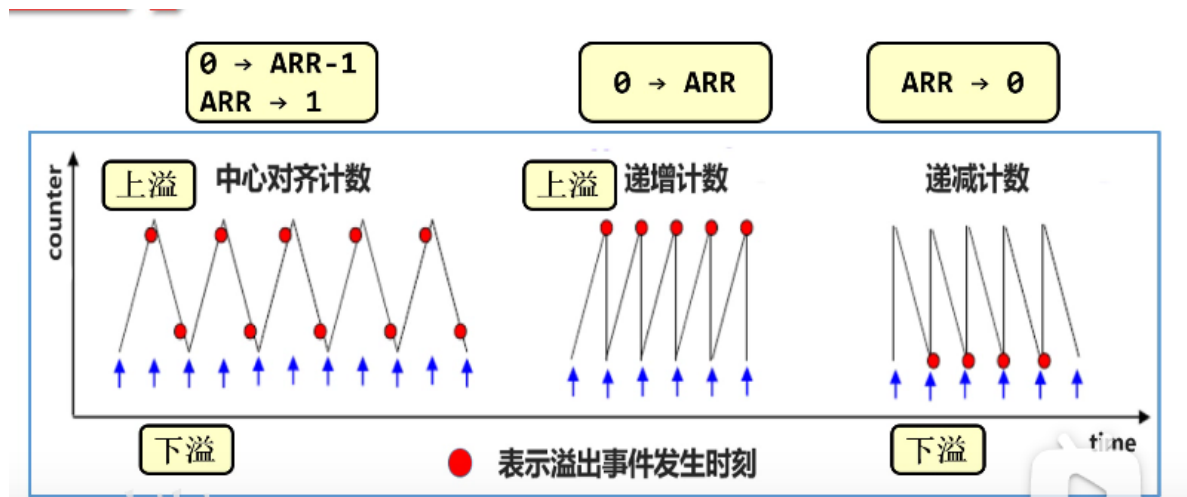


2.2计数模块

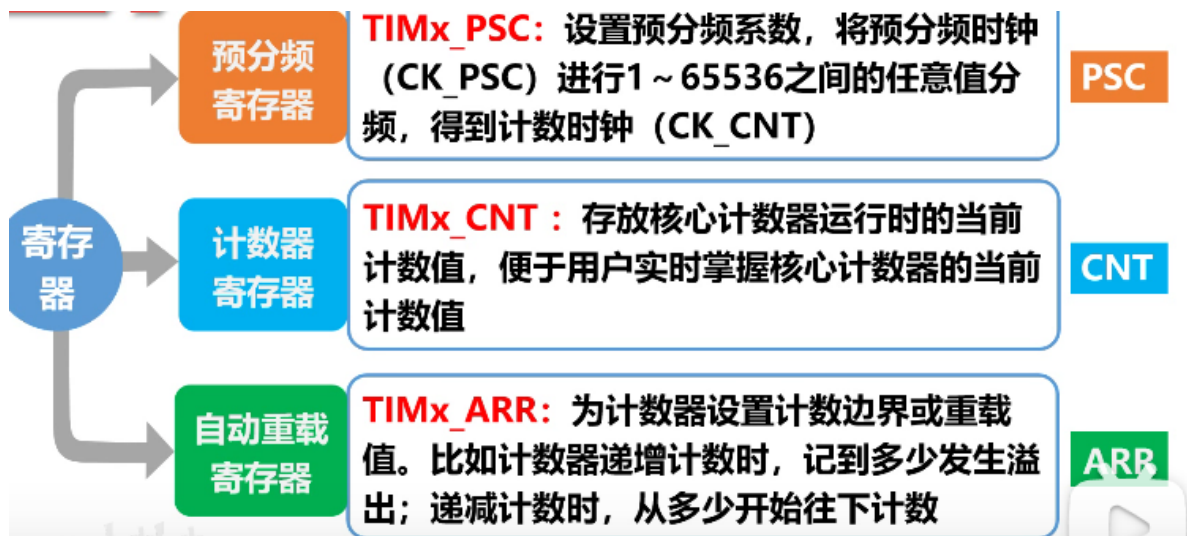
- 核心计数器：对计数时钟CK_CNT进行二次计数
- 计数器寄存器：TIM_CNT存放核心技术器运行时的当前计数值

2.3自动重载模块

6.计数器的三种计数模式



公式



7.外部脉冲计数

选择外部时钟模式

定时器PWM

1.什么是脉冲宽度调制(PWM)?

是一种对模拟信号电平进行数字编码的地方

实质是修改高电平持续时间

2. 定时器在PWM输出模式下是如何产生PWM波的

CNT < CCR时：输出高电平

CNT ≥ CCR时：输出低电平

CNT = ARR时：输出高电平

然后重复此部分

3. 如何产生特定频率，占空比的PWM波？(列出PWM波输出频率和占空比的计算公式)

控制ARR, PSC, TIM_CLK, CRR的值来实现

$$Period(s) = (ARR + 1) * (PSC + 1) / TIM_CLK$$

$$频率 = 1 / Period$$

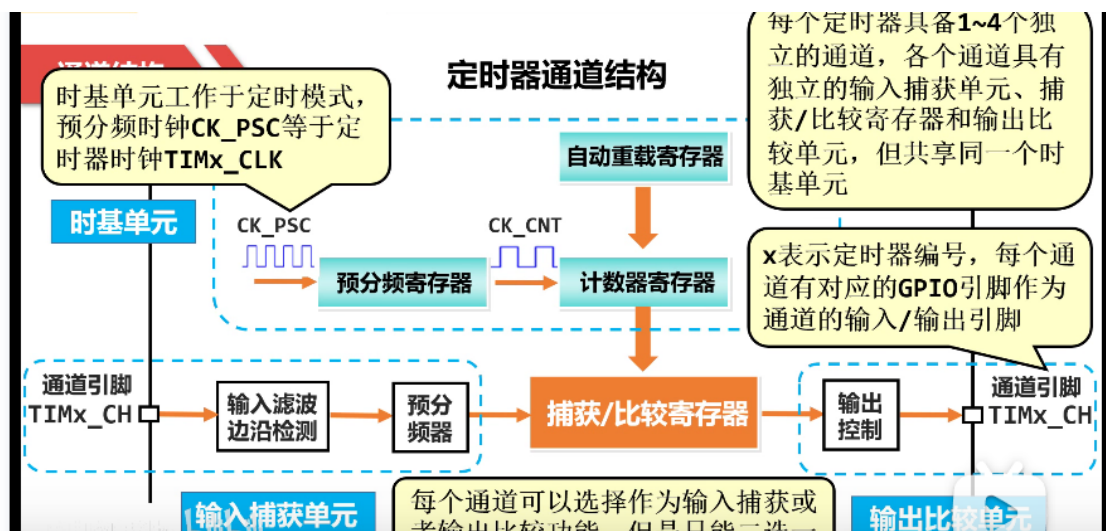
$$Duty = (CRR / (ARR + 1)) * 100$$

4. PWM信号的两个基本参数

- 周期Period：一个完整PWM波形所持续的时间
- 占空比Duty：高电平持续时间(Ton)和周期时间(Period)的比值

$$Duty = (Ton / Period) * 100$$

5. 通道结构



| 定时器2的通道引脚 | 对应GPIO引脚 |
|-----------|--------------|
| TIM2_CH1 | PA0/PA5/PA15 |
| TIM2_CH2 | PA1/PB3 |
| TIM2_CH3 | PA2/PB10 |
| TIM2_CH4 | PA3 |

6.功能单元作用

- 输入捕获单元
- 捕获比较寄存器
- 输出比较单元

7.工作原理

CNT<CCR时：输出高电平

CNT>=CCR时：输出低电平

CNT=ARR时：输出高电平

重复

PWM模式1

递增计数时，当TIMx_CNT（当前计数值）<TIMx_CCR（捕获/比较值）时，通道输出为**有效电平**，否则为无效电平。递减计数模式则刚好相反。

PWM模式2

递增计数时，当TIMx_CNT（当前计数值）<TIMx_CCR（捕获/比较值）时，通道输出为**无效电平**，否则为有效电平。递减计数模式则刚好相反。

8.PWM输出的两种模式

PWM模式1

递增计数时，当TIMx_CNT（当前计数值）<TIMx_CCR（捕获/比较值）时，通道输出为**有效电平**，否则为无效电平。递减计数模式则刚好相反。

PWM模式2

递增计数时，当TIMx_CNT（当前计数值）<TIMx_CCR（捕获/比较值）时，通道输出为**无效电平**，否则为有效电平。递减计数模式则刚好相反。

递增计数高电平有效时：**互补输出**

- PWM1下CCR用于控制高电平持续时间
- PWM2模式下CCR用于控制低电平持续时间

9.相关函数

1.PWM输出启动函数

2.定时器比较/捕获寄存器设置函数