

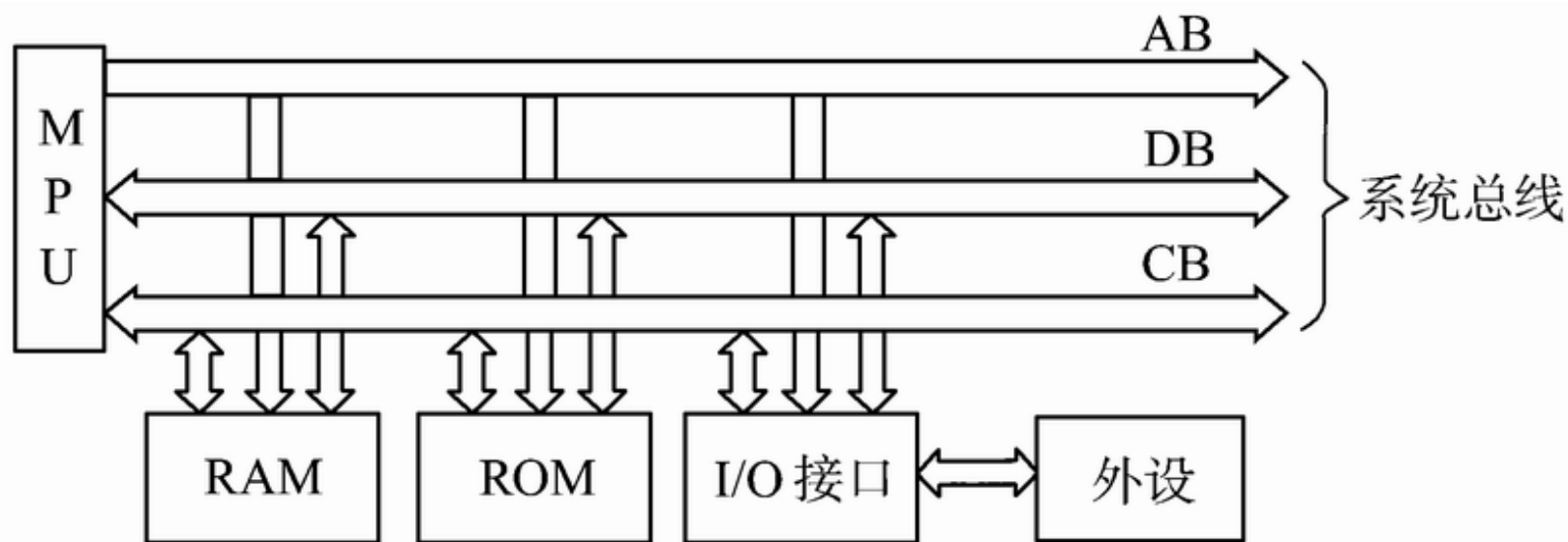
微机原理与接口技术

第八章 接口技术

第八章 接口技术

- * 第一节 I/O接口概述
- * 第二节 CPU与外设传送数据方式
- * 第三节 并行口与串行口
- * 第四节 GPIO 模块
- * 第五节 定时器/计数器
- * 第六节 数模转换器和模数转换器

第一节 I/O接口概述



- * **接口 (interface)**：是介于微处理器和外设之间的一种缓冲电路,是CPU与外部设备交换信息交换的中转站

第一节 I/O接口概述

* I/O接口电路的功能:

- * 数据的寄存和缓冲功能
- * 设备选择功能
- * 在微机与外部设备间传送控制和状态信息
- * 信号转换功能
- * 具备时序控制

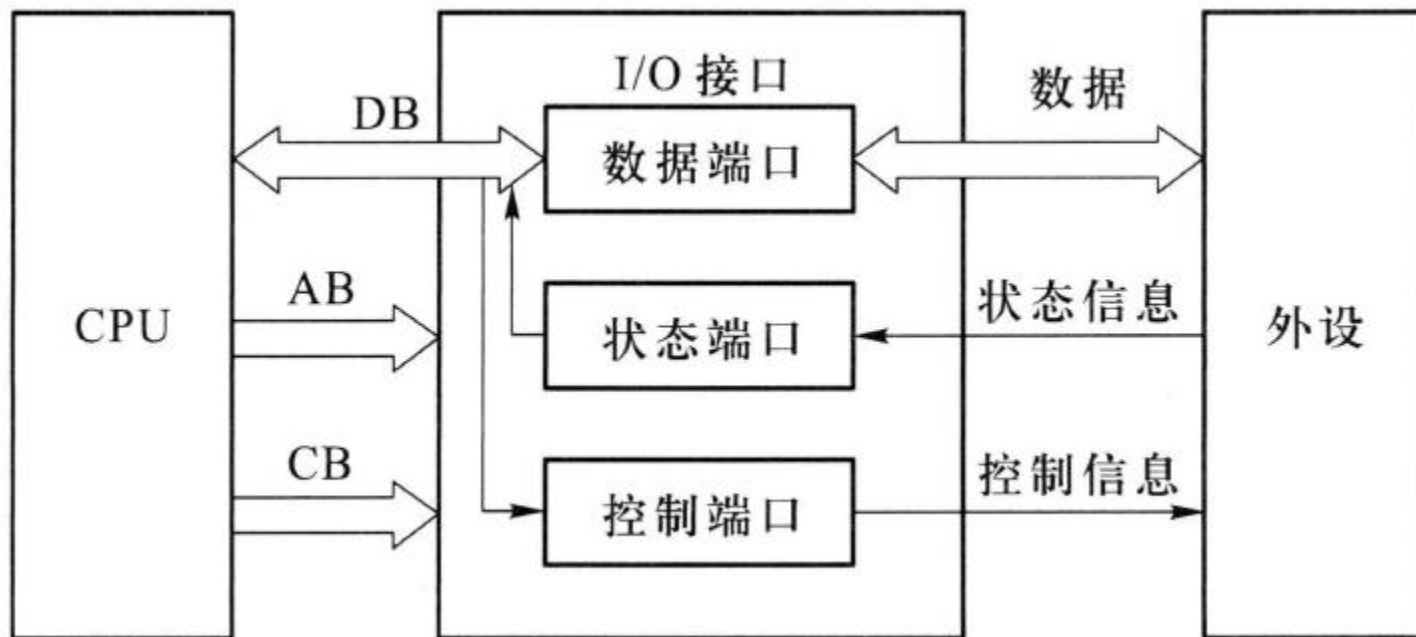
第一节 I/O接口概述

* CPU与I/O设备之间的信息通常包括：

数据信息，状态信息和控制信息

1. **数据信息**：数字量、模拟量、开关量
2. **状态信息**：状态信息一般是外设通过接口送到CPU的；主要是反映外设的工作状态；
3. **控制信息**：控制信息一般是CPU通过接口电路传送给外部设备的，主要用来控制外部设备的动作；

第一节 I/O接口概述



I/O 接口示意图

- * 系统总线包含有三种不同功能的总线，即数据总线DB（Data Bus）、地址总线AB（Address Bus）和控制总线CB（Control Bus）

*

数据总线DB用于传送数据信息。它既可以把CPU的数据传送到存储器或I/O接口等其它部件，也可以将其它部件的数据传送到CPU。需要指出的是，数据的含义是广义的，它可以是真正的数据，也可以指令代码或状态信息，有时可以是一个控制信息，因此，在实际工作中，数据总线上传送的并不一定仅仅是真正意义上的数据。

*

地址总线AB是专门用来传送地址的，地址只能从CPU传向外部存储器或I/O端口。地址总线的位数决定了CPU可直接寻址的内存空间大小，若地址总线为n位，则可寻址空间为 2^n 字节。

*

控制总线CB用来传送控制信号和时序信号。控制信号中，有的是微处理器送往存储器和I/O接口电路的，如读/写信号，片选信号、中断响应信号等；也有是其它部件反馈给CPU的，比如：中断申请信号、复位信号、总线请求信号、限备就绪信号等。因此，控制总线的传送方向由具体控制信号而定，控制总线的位数要根据系统的实际控制需要而定。实际上控制总线的具体情况主要取决于CPU。

第一节 I/O接口概述

- 为了区分以上所述的三种信息，在接口部件中都包含一组寄存器；CPU和外设交换信息时，将根据三种不同的信息，对不同的寄存器进行读写；
- 这些寄存器就称为**端口 (PORT)**；一个外设往往需要几个端口地址；不同的端口中存放不同的信息；
- 端口主要有三类，**状态口，命令口和数据口**
- CPU需要访问外设时，寻址的是端口，而不是笼统的外设；

第一节 I/O接口概述

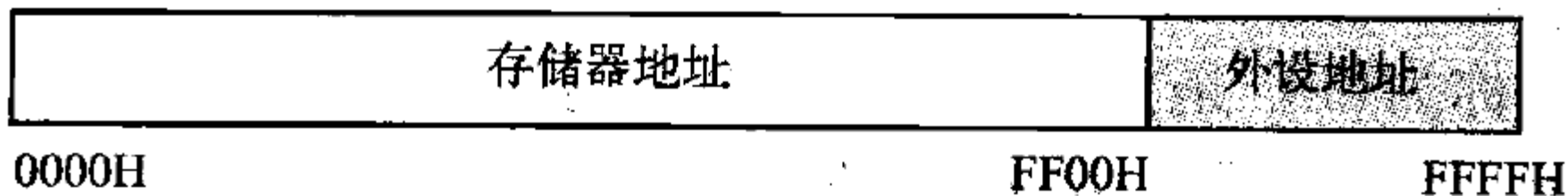
- * I/O接口: 指CPU和外设间的I/O接口芯片和电路
- * I/O端口: 常指在接口电路中用以完成某种信息传送, 并可由编程人员通过端口地址进行读写的寄存器。
- * 一个外设需要一个接口, 但一个接口可以有不止一个端口。
- * 每个端口对应一个地址, CPU通过不同的地址的访问来区别状态信息, 数据信息和命令信息。

第一节 I/O接口概述

* 端口的编址方式

1. 存储器统一编址方式（存储器映像）

- * 在存储空间中划出一部分给外设端口，对端口当作存储器单元一样进行访问，**不设置专门的I/O指令**；
- * 优点：可以直接使用访问存储器的各种指令访问外设端口，使用方便；不需要专门的输入/输出指令；端口地址安排灵活，数量不限。
- * 缺点：使存储器容量变小等；

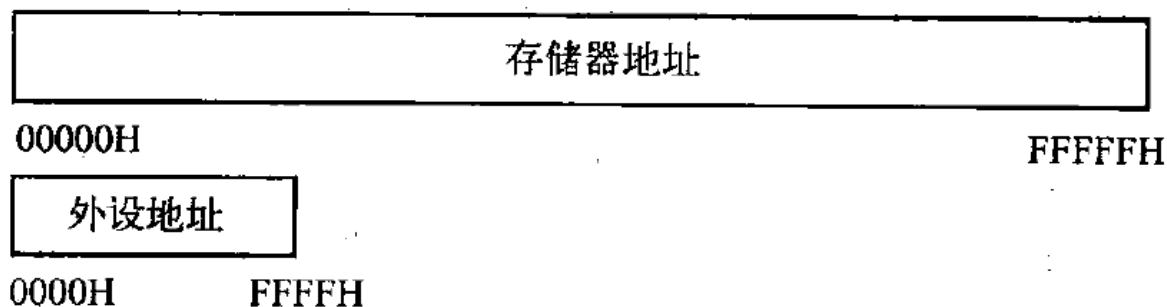


第一节 I/O接口概述

* 端口的编址方式

2. 独立编址方式 (I/O映像)

- * 端口地址单独编址，而不和存储空间合在一起；设置专门的I/O指令来访问端口；
- * 特点：外设地址和内存地址是可以重复的；必须区分发出的地址是存储器地址还是外设地址。需要专用的I/O指令。



第二节 CPU与外设传送数据方式

一、无条件传送方式

- * 在传送信息时，已知外设是“准备好”的；所以CPU在和外设交换数据时，不用查询外设的状态，直接进行输入/输出；
- * 一般用于：
 - * 外设是简单设备
 - * 外设工作速度足够快
 - * 两次数据传送的间隔时间足够长
- * 接口电路中一般有输入缓冲器/输出锁存器（数据端口），由地址译码的输出信号来选通它们。

第二节 CPU与外设传送数据方式

二、查询传送方式

- * 这种方式，CPU要遵循“先查询，后传送”的原则，保证只有在外设已经是在“准备好”状态，才开始传送数据；
- * 接口电路中至少需要两个端口：状态端口和数据端口
- * 查询式传送的一般流程
 - * 先从状态口读入状态字；
 - * 如果状态是“准备好”，开始传送；
 - * 如果状态是“没有准备好”，则继续查询，直到“准备好”，开始传送；

第二节 CPU与外设传送数据方式

二、查询传送方式

- * 在编写查询式传送程序时，要先确定两个问题：
 - * **状态信号的位置**：需要规定状态信号是在寄存器中的第几位；
 - * **状态信号的有效电平**：即是高电平表示准备好，还是低电平表示外设准备好；
- * **优点**：CPU和外设之间可以很好地配合工作；
- * **缺点**：CPU要长期地查询外设的状态，查询实际上就是一种等待；CPU长期的等待会影响CPU的工作效率；

第二节 CPU与外设传送数据方式

三、中断传送方式

- * 当外设要输入/输出时（即外设已把待输入数据存放在数据输入寄存器，或输出时外设已把上一个数据输出，输出寄存器已空），向CPU发中断请求；
- * CPU收到中断请求后，中断当前的工作，为外设服务，服务结束(输入或输出)后，继续原来的工作；
- * 中断方式允许CPU与外设(甚至多个外设)同时工作，克服了查询方式的缺点；

第二节 CPU与外设传送数据方式

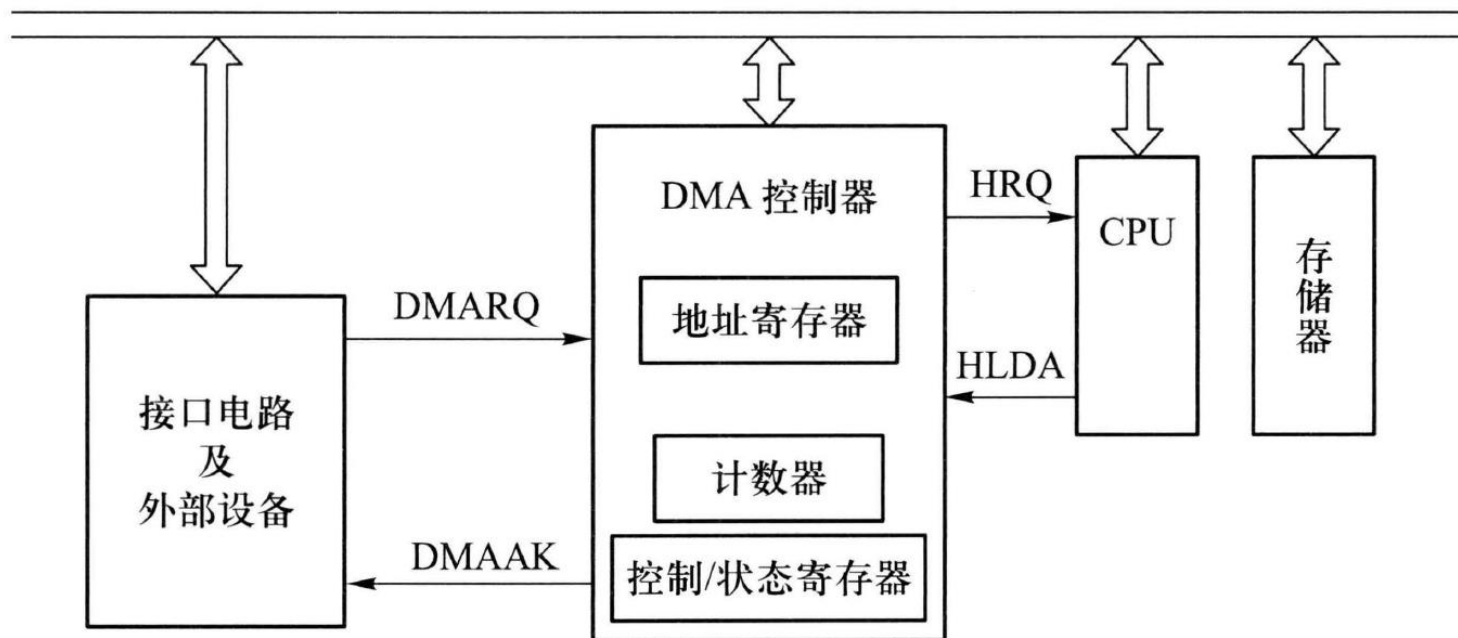
四、直接存储器存取方式 (DMA)

- * 中断方式仍是CPU通过程序来传送，每次传送一字节；为了要传送这一字节需要将保护断点，保护现场等多条指令执行一遍；
- * 对一个高速I/O设备，成组交换数据的情况，中断方式显得不合适；
- * DMA方式是由硬件(DMA控制器)来控制数据从外设到存储器的直接传送，而不通过CPU；
- * DMA方式下，要求CPU让出这些总线，即要求CPU相应的引脚输出为高阻状态，系统总线由DMA控制器接管这些，产生相应的总线信号；

第二节 CPU与外设传送数据方式

四、直接存储器存取方式 (DMA)

* DMA 传送的工作原理:



第二节 CPU与外设传送数据方式

四、直接存储器存取方式 (DMA)

DMA 控制器是能在存储器和外部设备之间实现直接而高速地传送数据的一种专用处理器。它应具有独立访问存储器的能力，能取代CPU，提供存储器地址和必要的读写控制信号，将数据总线上的信息写入存储器或从存储器读出，为此，要求DMA 控制器具有独立控制三总线来访问存储器和I/O 端口的能力。

第三节 并行口与串行口

I/O接口的“本质”是电路，这些电路包括：

- * I/O端口寄存器
- * 地址译码电路
- * 传送方式控制电路
- * 并-串/串-并变换等转换电路

接口部件，使用**本身的寄存器**来实现各种接口，这些寄存器就是I/O端口。

接口部件的寄存器——**CPU当作RAM单元访问**

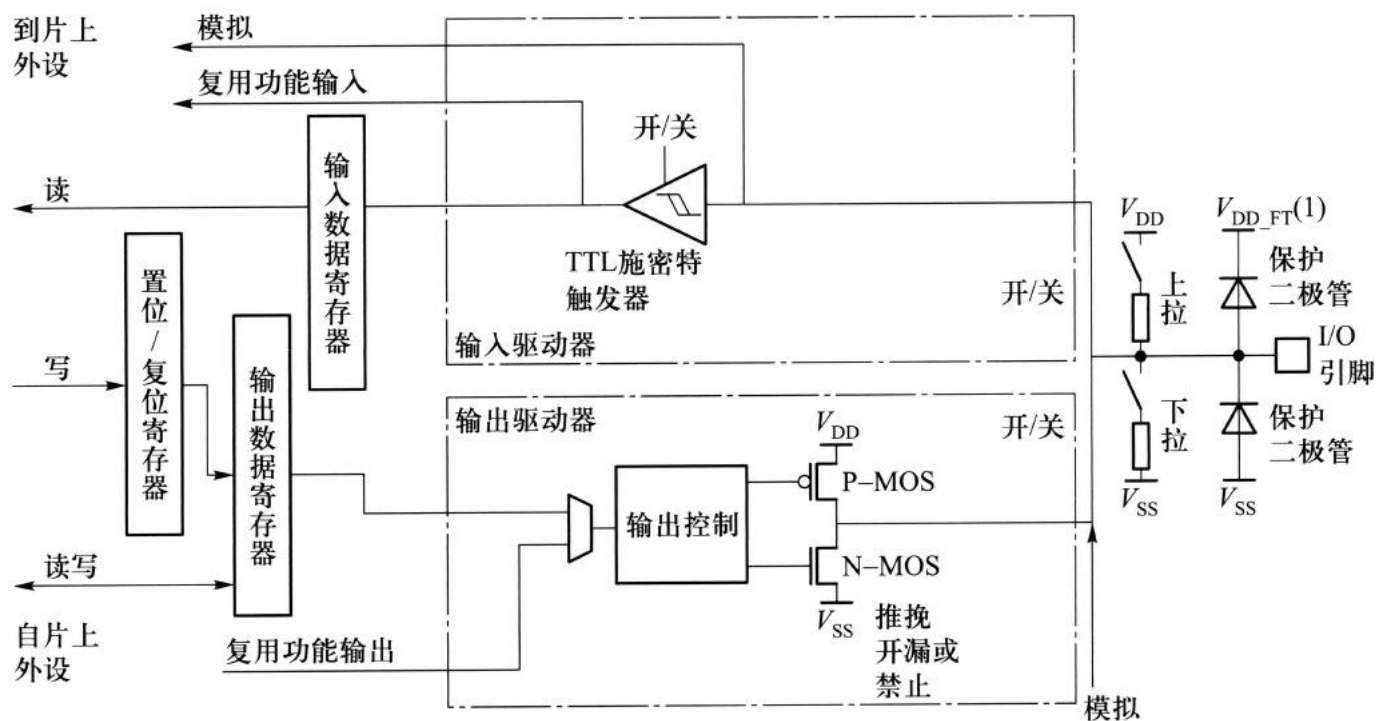
第三节 并行口与串行口

- * 并行I/O接口：以并行方式和CPU传送I/O接口，以并行方式和外设交换数据。
- * 串行口：数据和控制信息是一位接一位串行按顺序的传送的。因为计算机系统内部的数据是并行传送的，所以要串行传送数据的话，需要进行并---串/串---并变换。

第四节 GPIO 模块

GPIO的基本特性

GPIO的基本结构



第四节 GPIO 模块

GPIO 引脚的工作模式：

- (1) 输入模式（上拉／下拉／浮空）**
- (2) 输出模式（推挽／开漏、上拉／下拉／浮空）**
- (3) 复用功能（推挽／开漏、上拉／下拉／浮空）**
- (4) 模拟输入输出模式**

第四节 GPIO 模块

STM32H7xxd的GPIO寄存器

- 4个32-bit 配置寄存器(MODER,OTYPER, OSPEEDR PUPDR),
- 2个32-bit 数据寄存器(IDR,ODR)
- 1个32-bit 位操作寄存器 (BSRR).
- 1个32-bit 锁存寄存器 (LCKR)
- 2个32-bit 复用功能选择 (AFRH , AFRL).

第四节 GPIO 模块

GPIO 寄存器:

* 每个GPIO 端口都有4 个32 位存储器映射的**配置寄存器**，可以用来对GPIO 端口的每个I/O引脚进行设置:

(1) **GPIOx_MODER** 寄存器用于选择I/O方向及工作模式: 输入、输出、AF、模拟;

(2) **GPIOx_OTYPER** 寄存器用于选择输出类型: 推挽或开漏;

(3) **GPIOx_OSPEEDR** 寄存器用于选择输出速度: 2MHz、25 MHz、50MHz、100 MHz, 输出速度是指I/O引脚支持的高、低电平状态的最高切换频率;

(4) **GPIOx_PUPDR** 寄存器用于选择: 上拉/下拉;

第四节 GPIO 模块

GPIO 寄存器:

* 每个GPIO 端口都具有2个16 位**数据寄存器**:

(1) **GPIOx_IDR** 寄存器: 通过I/O引脚输入的数据存储到输入数据寄存器, 它是一个只读寄存器;

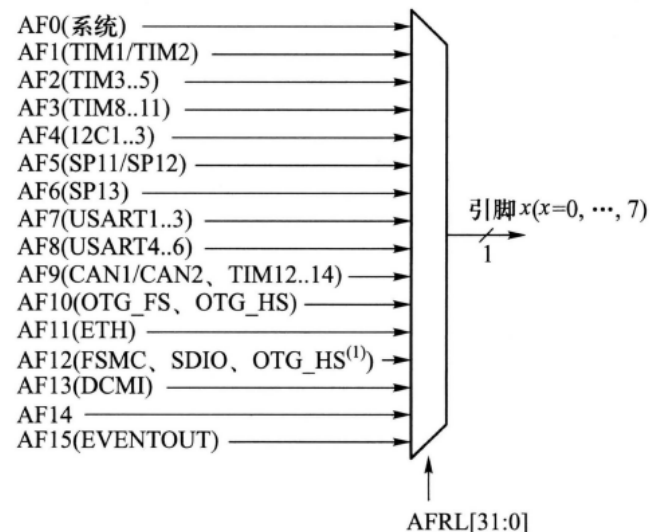
(2) **GPIOx_ODR** 寄存器: 用于存储输出数据, 可对其进行读写访问。
。写入该寄存器的数据直接控制I/O引脚输出高、低电平。

第四节 GPIO 模块

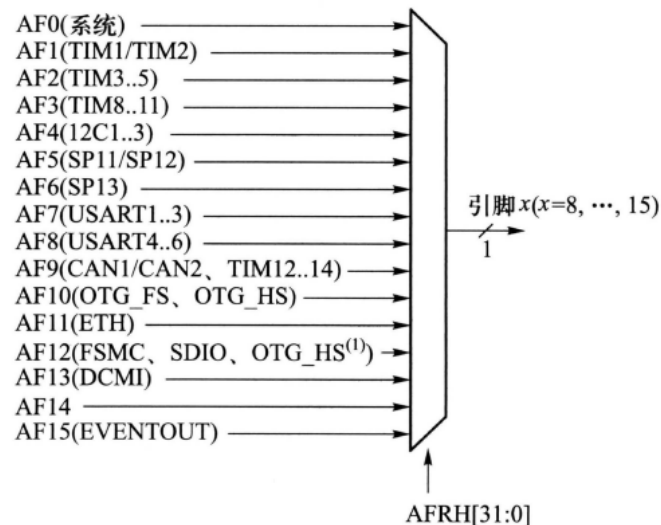
GPIO 寄存器:

- * 每个GPIO 端口还有2 个32 位复用功能寄存器GPIOx_AFRL 和 GPIOx_AFRH，当I/O引脚配置为复用功能工作模式时，通过对这两个寄存器编程，可以使得I/O引脚与可用的16个复用功能引脚中的一个连通。

对于引脚0到引脚7，GPIOx_AFRL[31:0]寄存器会选择专用的复用功能



对于引脚8到引脚15，GPIOx_AFRH[31:0]寄存器会选择专用的复用功能



第四节 GPIO 模块

GPIO 寄存器:

- * **STM32的I/O接口很多，每个I/O接口都对应着一个时钟。为了降低功耗，当芯片上电时这些时钟都是关闭的。当I/O接口工作时，必须将其对应的时钟使能。STM32 所有外I/O接口的时钟都是由复位时钟控制器(RCC) 来管理的。**

位 8:0: GPIOx 时钟使能,由软件置 1 和清零

0: 禁止 GPIOx 时钟

1: 使能 GPIOx 时钟

[illegible]

第四节 异步串行通信接口

* 串行通信的特点:

- * 一根线上既要传数据，又要传控制信息
- * 信息格式有固定的要求
- * 串行通信中对信息的逻辑定义与TTL不兼容，要进行逻辑电平转换

(TTL电平标准:

输出 L: $<0.8V$, H: $>2.4V$; 输入 L: $<1.2V$, H: $>2.0V$

RS232标准:

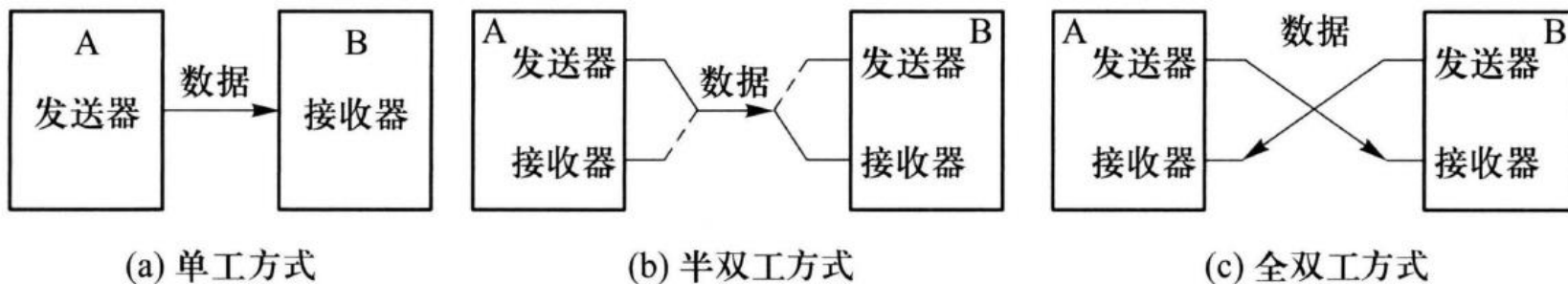
逻辑1的电平为 $-3\sim-15V$ ，逻辑0的电平为 $+3\sim+15V$ ，注意电平的定义反相了)

* 串行通信的分类:

- * 异步，发送和接收双方使用各自的时钟
- * 同步

第四节 异步串行通信接口

* 数据传送方式



第四节 异步串行通信接口

* 起止式异步通信协议

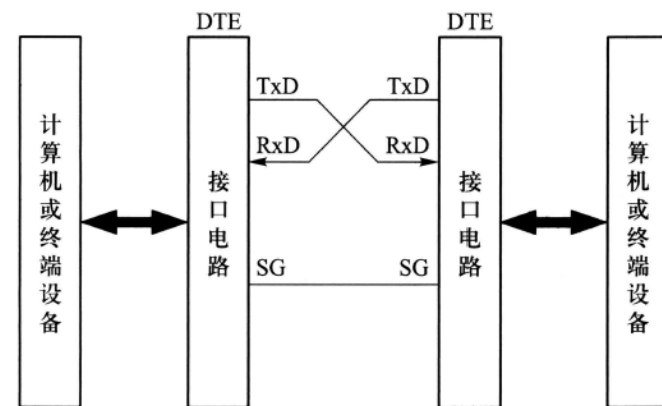
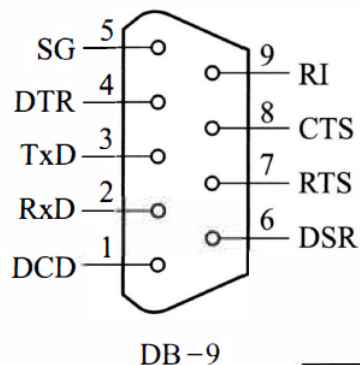
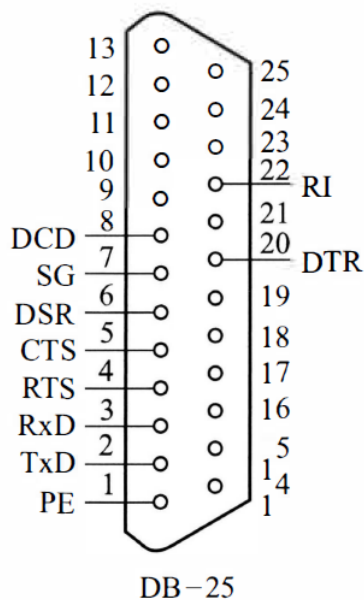
- (1) 1 位起始位，规定为低电平0，是字符开始的标志；
- (2) 5~8 位数据位，规定低位在前，高位在后；
- (3) 0 位或1 位奇偶校验位；
- (4) 1 位、1.5 位或2 位停止位，规定为高电平“1”，是字符结束的标志。

字符与字符之间的空闲位用高电平“1” 填充。



第四节 异步串行通信接口

* RS-232C 接口标准



引脚号 (9 针)	引脚号 (25 针)	信号	方向	功能
3	2	TxD	输出	发送数据
2	3	RxD	输入	接收数据
7	4	RTS	输出	发送请求
8	5	CTS	输入	发送清除
6	6	DSR	输入	数据通信设备 (DCE) 就绪
5	7	SG		信号公共参考地
1	8	DCD	输入	数据载波检测
4	20	DTR	输出	数据终端设备 (DTE) 就绪
9	22	RI	输入	振铃指示

第四节 异步串行通信接口

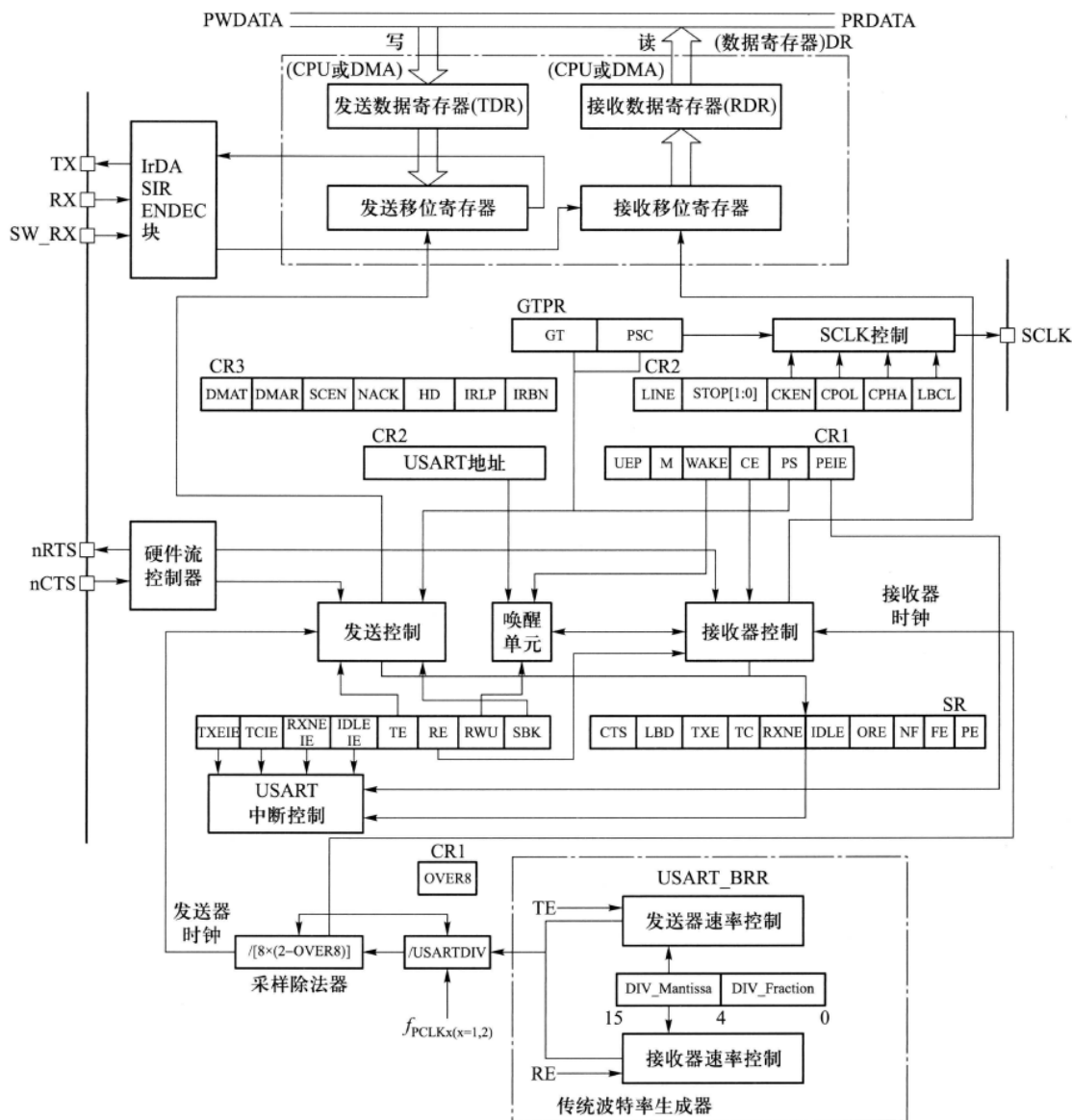
STM32F407的USART模块

TX：发送数据输出引脚。

RX：接收数据输入引脚。

PWDATA：需要发送的并行数据，通过PWDATA 写入发送数据寄存器TDR。

PRDATA：接收到的并行数据保存在接收数据寄存器RDR 中，应用程序通过PRDATA读取接收到的数据。



第四节 异步串行通信接口

STM32F407的USART模块

波特率寄存器(USART_BRR)

位15:4 DIV_Mantissa [11:0]：这12个位用于定义USART除数(USARTDIV)的尾数。

位3:0 DIV_Fraction [3:0]：这4个位用于定义USART除数(USARTDIV)的小数。

f_{ck} 为外设的时钟，分频因子USARTDIV是一个存放在寄存器USART_BRR中的无符号定点数。

例如：假设计算得到的分频因子是27.75，则DIV_Mantissa=0d27=0x1B，小数部分的计算0.75=12/16，所以DIV_Fraction=0xC，于是USART_BRR=0x1BC。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

图 8-3-17 波特率寄存器 (USART_BRR)

串口波特率的计算公式如下：

$$T_x/R_x \text{ 波特率} = \frac{f_{ck}}{8 \times (2-OVER8) \times USARTDIV}$$

第四节 异步串行通信接口

STM32F407的USART模块

例 8-11 异步通信双方约定数据格式采用一位起始位、七位数据位、一位偶校验位和一位停止位；波特率选择 1 200 b/s,波特率因子为 16。试确定收 / 发时钟频率、每秒传送的字符数、传输效率,画出传输字符“E”的波形图。

解:

(1) 收、发时钟频率应为

$$1\,200 \times 16 \text{ Hz} = 19.2 \text{ kHz}$$

(2) 每秒传送的字符数为

$$1\,200 / (1 + 7 + 1 + 1) \text{ 个} = 120 \text{ 个}$$

(3) 传输效率为

$$7 / (1 + 7 + 1 + 1) = 70\%$$

(4) 传输字符“E”的波形图如图 8-3-4 所示,其中 E 的 ASCII 码为 45H,校验位为 1。

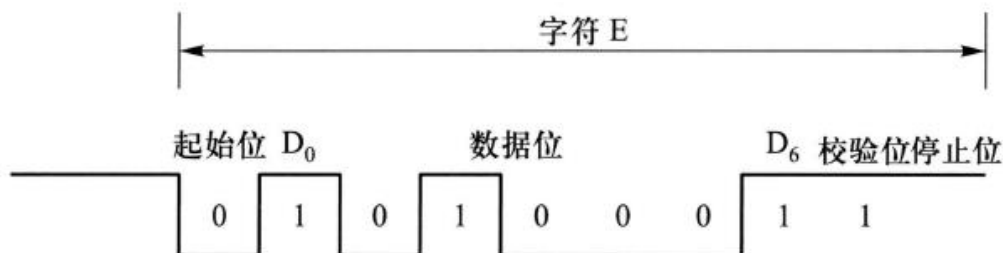


图 8-3-4 传输字符 E 的波形图

第四节 异步串行通信接口

STM32F407的USART模块

例 8-11 异步通信双方约定数据格式采用一位起始位、七位数据位、一位偶校验位和一位停止位；波特率选择 1 200 b/s,波特率因子为 16。试确定收 / 发时钟频率、每秒传送的字符数、传输效率,画出传输字符“E”的波形图。

解:

(1) 收、发时钟频率应为

$$1\,200 \times 16 \text{ Hz} = 19.2 \text{ kHz}$$

(2) 每秒传送的字符数为

$$1\,200 / (1 + 7 + 1 + 1) \text{ 个} = 120 \text{ 个}$$

(3) 传输效率为

$$7 / (1 + 7 + 1 + 1) = 70\%$$

(4) 传输字符“E”的波形图如图 8-3-4 所示,其中 E 的 ASCII 码为 45H,校验位为 1。

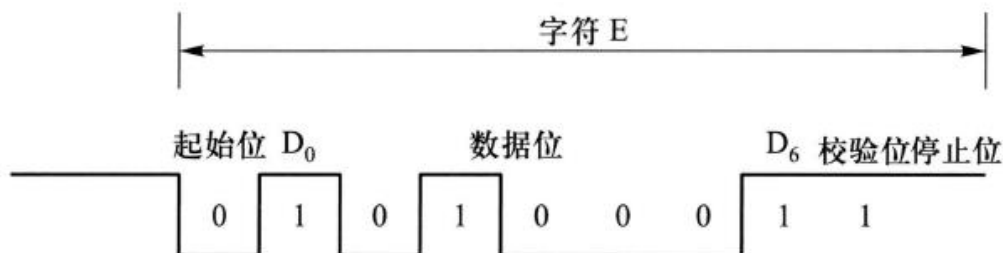


图 8-3-4 传输字符 E 的波形图

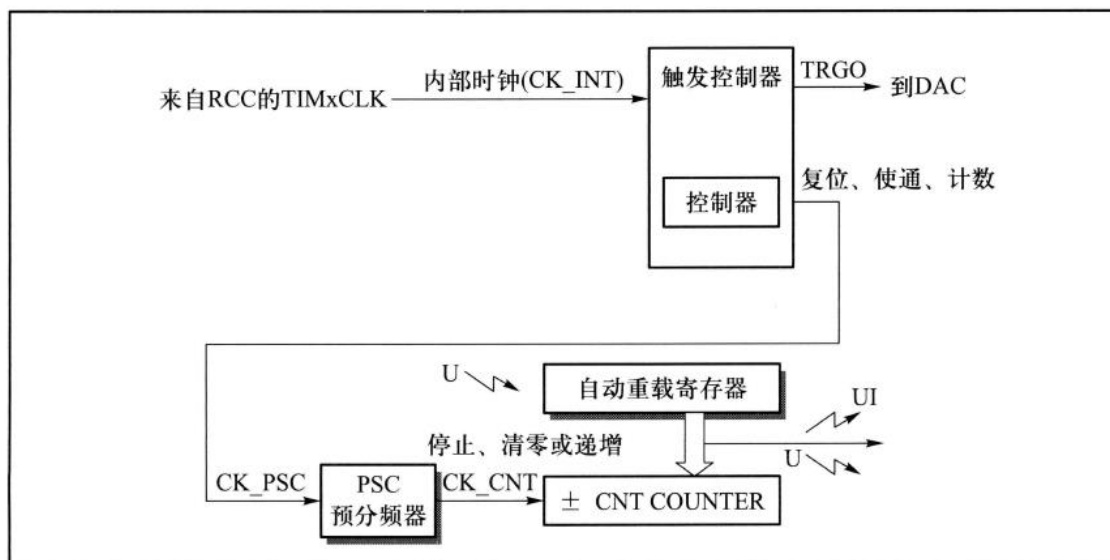
第五节 定时器/计数器

1、基本定时器

通过对一个16位自动重载递增计数器(TIMx_ARR) 和一个16位预分频器(TIMx_PSC) 编程进行配置。

当定时器开始工作时，计数器CNT_COUNTER在计数时钟CK_CNT 的驱动下，计数值从0 开始不断递增（向上计数），当计数值与TIMx_ARR 寄存器的数值相等时，称计数器达到上溢值，定时器自动产生更新事件，完成一次计数过程。

$$CK_CNT = CK_PSC / (TIMx_PSC + 1)$$



标志 发生更新事件时，根据控制位传输到活动寄存器的预装载寄存器

事件

中断和DMA输出

第五节 定时器/计数器

2、通用定时器

每个定时器具有4个独立通道:

TIMx_CH1~TIMx_CH4

支持输入捕获、输出比较、脉冲宽度调制PWM生成以及单脉冲模式输出

