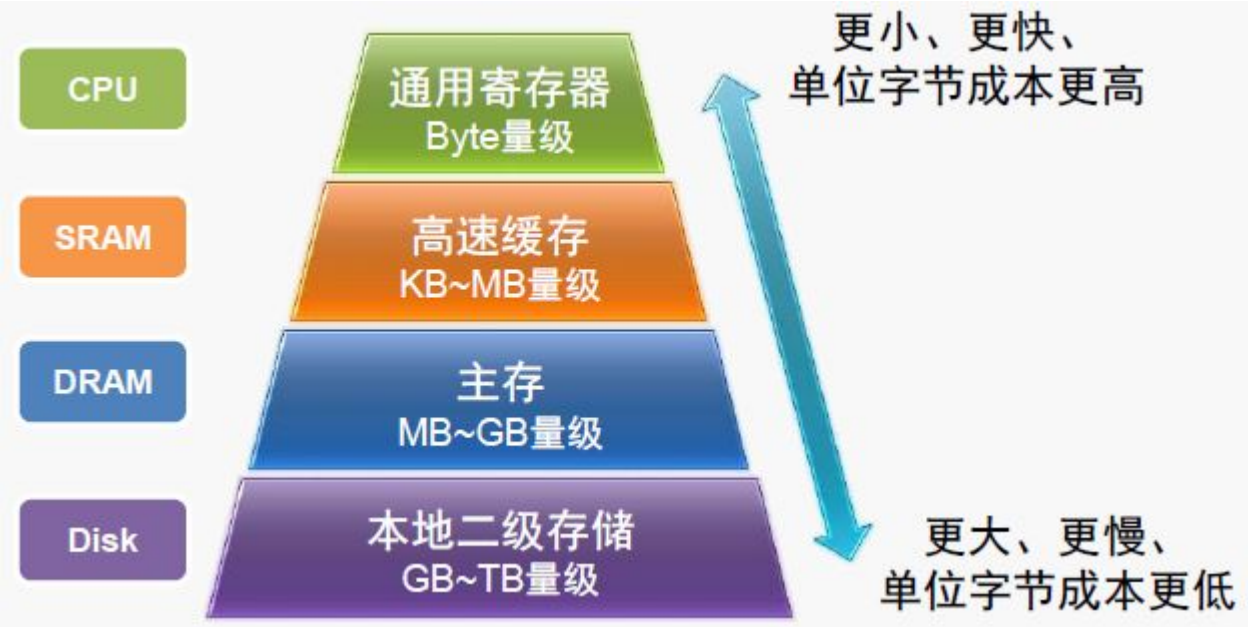
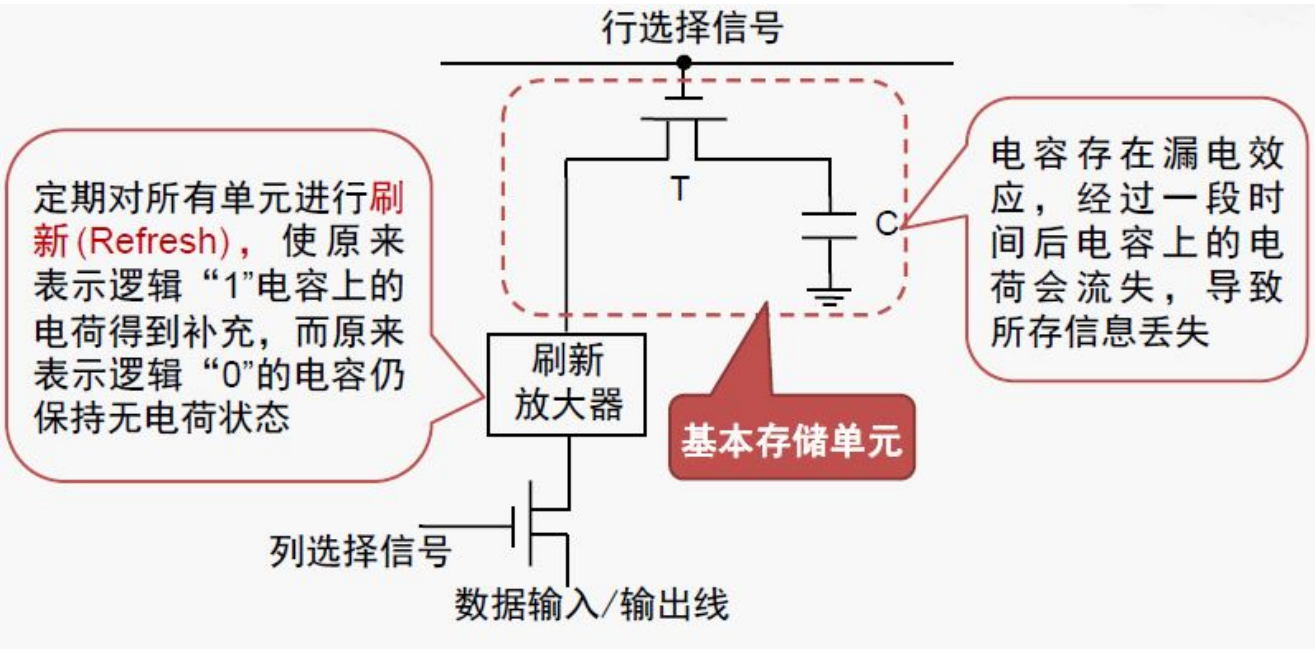
计算机组成 Final Exam

1. 存储层次结构

存储器的特性——非易失性，可读可写，随机访问，访问时间，容量，价格，功耗...



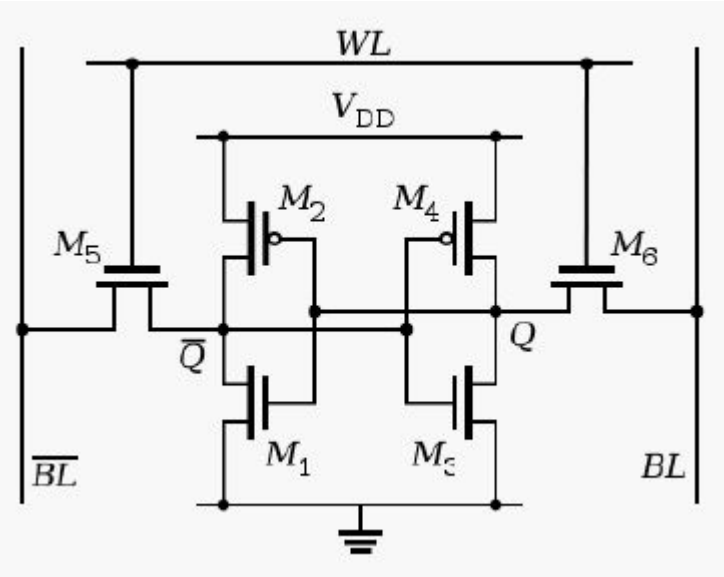
DRAM为电容阵列，通过行选择信号和列选择信号来选择到某个基本存储单元，充电为1，无电荷为0。由于电容会存在漏电效应，因此需要定期对所有单元进行刷新，补充为1的电容上的电荷。



优点：集成度高，功耗低，价格低；缺点：速度慢，需要定时刷新。

现代PC机大多采用DRAM作为主存，SDRAM，DDR3 SDRAM等。

SRAM使用MOS实现。存储单元内部是一个双稳态触发器。写入数据时在BL输入数据，并将WL置1；读取数据时将WL置1，从BL读取数据。



优点：速度快；缺点：集成度低，功耗高，价格高

现在CPU中的Cache常用SRAM实现。

DRAM vs. SRAM

	DRAM	**SRAM**
存储单元	电容	双稳态触发器
集成度	高:)	低
功耗	低:)	高
价格	低:)	高
速度	慢	快:)
刷新	定期	无需:)

SDRAM访存——总线请求；（预充电和）行选择；列选择；总线传输。

总线请求：CPU通过系统总线向内存控制器发出访问请求

行选择：内存控制器通过行解码器选到所需的行。ACT (Activate) or RAS (Row Access Strobe)

列选择：内存控制器通过列解码器选到所需的列，此时定位到需要读出的存储单元。 READ or CAS (Column Access Strobe)

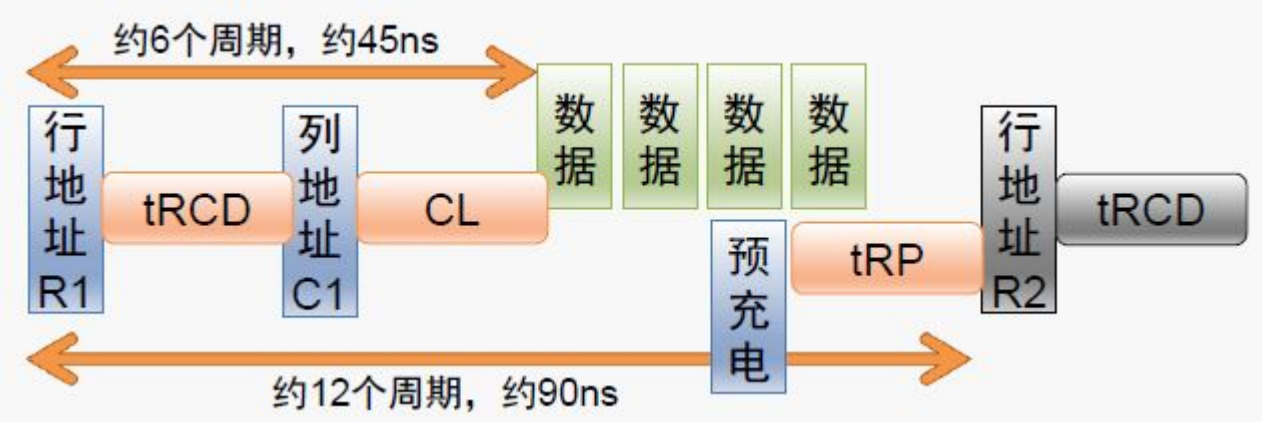
总线传输：读出所需的存储单元，通过系统总线送回CPU。

SDRAM的关键性能参数

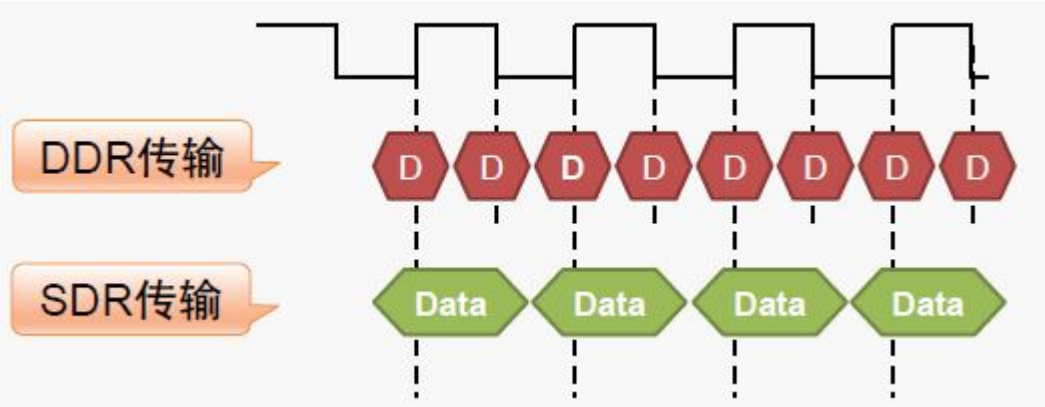
tRCD——time Row to Column Delay，从行选到列选的延迟

CL——CAS Latency，从列选到数据输出的延迟

tRP——time RAS Precharge，行预充电（关闭行）的延迟

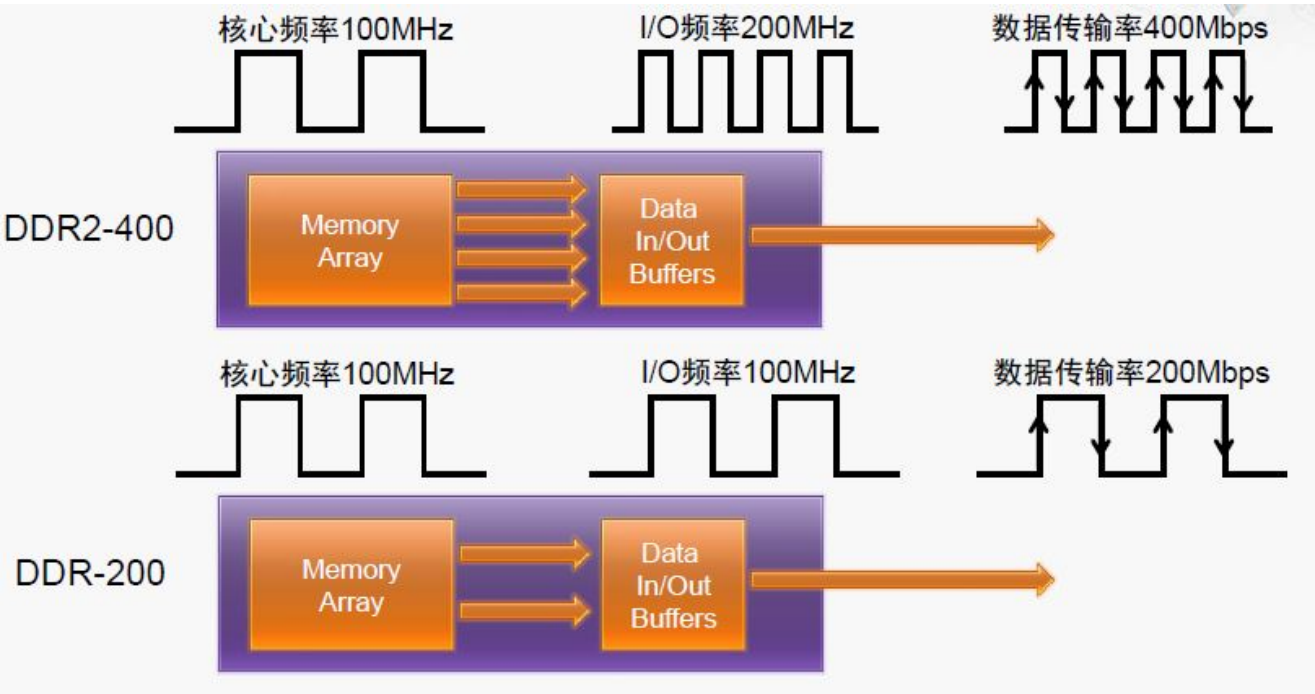


DDR——Double Data Rate，不止在上升沿传输数据，也在下降沿传输数据。



核心频率不变的情况下，采用DDR传输，相当于等效频率提高一倍。

DDR2——在DDR的基础之上，提升I/O频率，将I/O频率提升至核心频率的两倍，使得数据传输率相较于SDR提升至四倍。



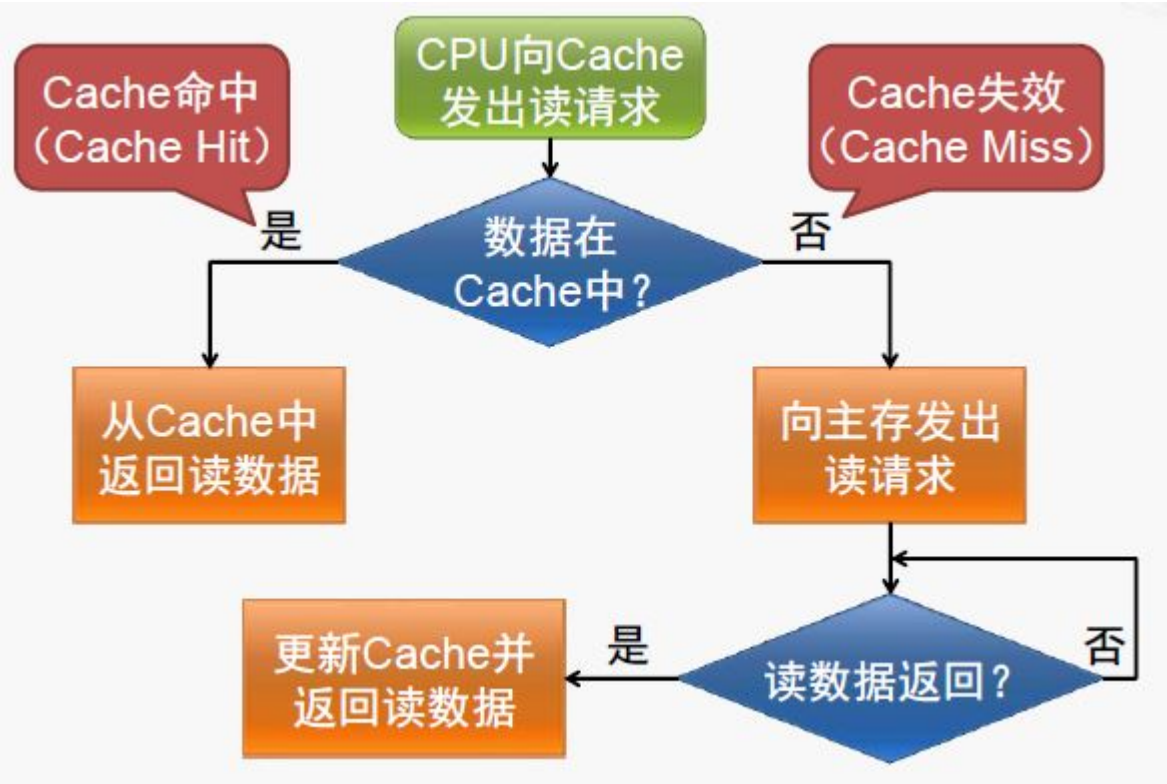
程序局部性原理——计算机程序从时间和空间都表现出局部性

时间局部性：最近被访问的存储器单元（指令/数据），很快还会被访问

空间局部性：正在被访问的存储器单元附近的单元很快会被访问

Cache很好地利用了时间局部性和空间局部性。

Cache的访问过程



Cache的写策略

HIT时的写策略：写穿透，数据同时写入Cache和Mem；写放回，数据只写入Cache，仅当该数据块被替换时才将数据写回Mem。

MISS时的写策略：写不分配，直接将数据写入主存；写分配，将数据所在块读入Cache后再将数据写入Cache。

Cache的重要参数——命中率，命中时间； 失效率，失效代价。

$$平均访存时间 = 命中时间 + 失效代价 \times 失效率$$

降低命中时间，降低失效代价，提升命中率即为Cache设计的目标。

Cache失效的原因——冷启动失效（第一次访问某个数据块），这种失效无法有效避免；容量失效（Cache无法保存程序访问所需的所有数据块），可以通过增加Cache容量缓解；冲突失效（多个存储器位置映射到同一Cache位置），需要设计Cache映射策略来有效解决。

Cache的映射策略——直接映射，多路组相联

Cache替换算法——Random，轮转，LRU

计算机领域容量计算方法

内部存储器容量——$1K = 2^{10}$, $1G = 2^{30}$，eg. Cache，Mem

外部存储器容量——$1K = 10^3, 1G = 10^9$，eg. 优盘，硬盘

数据传输率、时钟频率——$1K = 10^3, 1G = 10^9$，eg. CPU主频，SATA-2传输率，以太网传输率

2. 中断和异常

第一个带有异常处理的计算机系统——UNIVAC, 1951. 算术运算溢出时转向地址0执行修复指令或者停机。之后1955年，UNIVAC 1103增加了外部中断。

第一个带有外部中断的计算机系统——DYSEAC, 1954. 带有两个PC，根据I/O信号切换。

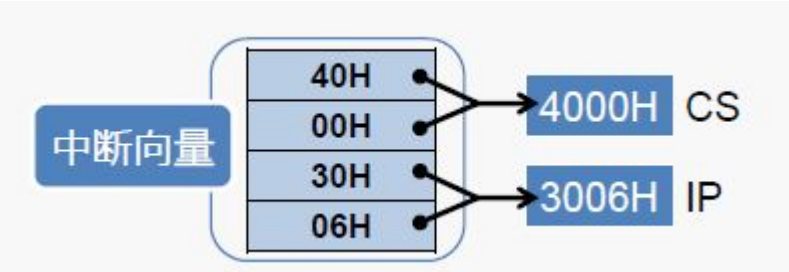
中断/异常——CPU遇到的特殊“事件”

在程序运行时，计算机系统内部、外部或者现行程序本身出现需要特殊处理的事件；CPU立即强制终止现行程序，改变工作状态并且启动相应程序来处理这些事件；处理完成后，CPU恢复到原来的程序运行。

中断向量——中断服务程序的入口地址，每个中断类型对应着一个中断向量，中断向量为4个字节。中断向量储存在中断向量表中，中断向量表位于存储器的专用区域里。

8086在实模式下，地址范围为00000H~FFFFFH，共1MByte，其中00000H至003FFH共1KByte，为中断向量表区，存放256个中断向量。

一个中断向量4个字节，从低到高前两个字节为中断服务程序入口地址的IP，后两个字节为CS，所有存储均低字节在前，高字节在后。如下中断向量实际是43006H。

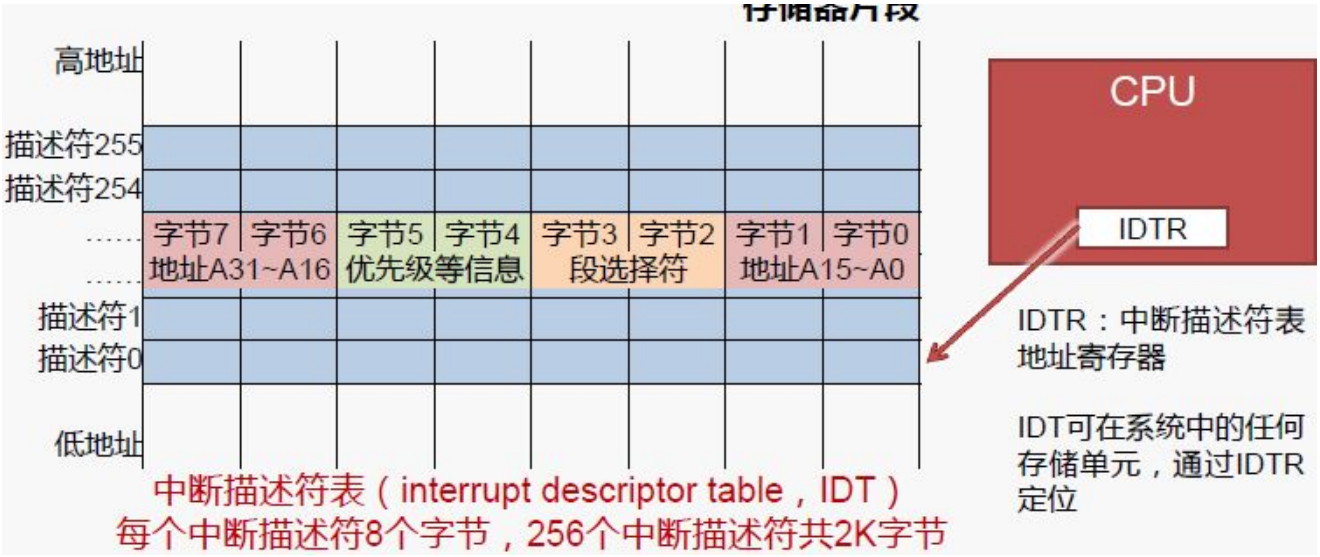


中断向量在中断向量表里顺序存放，00000H~00003H存放0号中断向量，00004H~00007H存放1号中断向量，...，003FCH~003FFH存放255号中断向量。

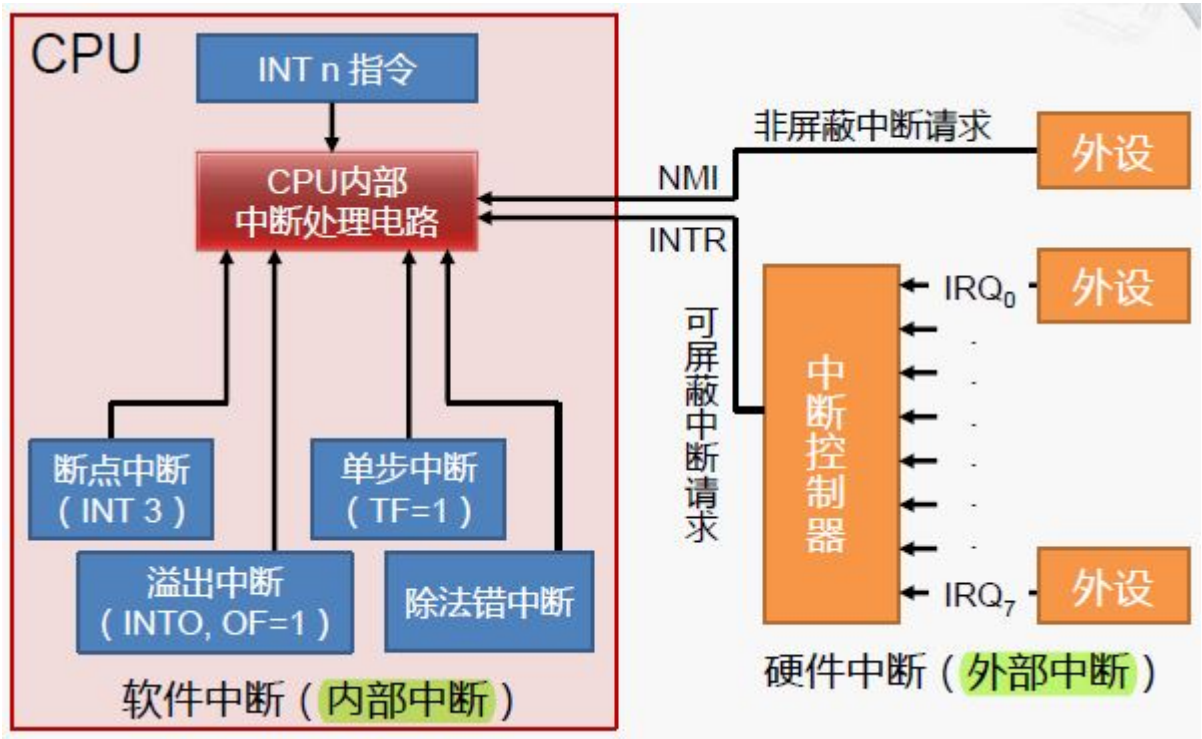
8086的中断向量表——0号为除零异常，1号为单步异常，2号为非屏蔽异常，3号为断点异常，5号为溢出异常，这五个为专用的中断；此后的5-31号共27个保留的中断，位CPU之后的更新而保留；32-255共224个可以用户自定义的中断。

80386~Core2的中断向量表——19个专用的中断，13个保留的中断，224个供用户自定义的中断。

IA-32在保护模式下，段基质通过GDTR和CS共同从内存中的GDT中取出，因而中断向量表也要通过类似的机制得到。通过IDTR(终端描述符表地址寄存器)找到IDT，之后的操作与实模式下类似。IDT中每个中断描述符8个字节，256个中断描述符共2KByte。



中断类型：硬件中断（外部中断）——来自外设，分作可屏蔽中断请求和非屏蔽中断请求；软件终端（内部中断）——来自CPU内部和程序系统调用等。



中断处理过程 (x86示例)——前3项通常由处理中断的硬件电路完成，后三项通常由软件（中断服务程序）完成

关中断——CPU关闭中断响应，不再接受其他外部中断请求

保存断点——将发生中断的指令地址压入堆栈(具体的保存哪个地址和中断类型相关)，以便中断处理完之后能够正确地返回

识别中断源——CPU识别中断来源，确定中断的类型号，找到相应的终端服务程序入口地址

保护现场——将发生中断处的相关寄存器（中断服务程序需要使用的寄存器）即标志寄存器的内容压入堆栈

执行中断服务程序——转到中断服务程序入口开始执行，此时可以在适当时刻重新开放中断，以便允许相应高优先级的外部中断

恢复现场并返回——将压入堆栈的信息弹回原寄存器，然后执行**IRET指令**返回主程序继续运行

IF标志位——软件开放/关闭中断响应的方法。

IF=1，允许CPU相应可屏蔽中断请求；**STI指令**将IF置1

IF=0，不允许CPU相应可屏蔽中断请求；**CLI指令**将IF置0

ID对于内部中断和非屏蔽中断不起作用

IRET指令——中断返回指令

从栈顶弹出三个字，分别放入IP，CS，FLAGS寄存器（进入程序时，硬件会自动压栈这三个值）。之后的扩展还有IRETD指令和IRETQ指令。

x86实模式内部中断——除法错中断，断点中断，单步中断，溢出中断

除法错中断（0号中断）——执行除法指令后所得商超出目标寄存器表示范围，比如除零

溢出中断（4号中断）——执行INTO指令，在OF=1是引起4号内部中断；若OF=1则执行空操作；INTO指令通常用于算术运算之后以便能及时处理溢出，等同于INT 4

单步中断（1号中断）——TF=1时，CPU处于单步工作方式；该方式下，CPU每执行一条指令，就产生一个1号中断，进入类型1中断服务程序；该中断服务程序通常显示CPU内寄存器内容等用于Debug

断点中断（3号中断）——在程序中设置断点，引起断点中断；**所有的INT n形式的指令中只有INT 3是单字节长的指令(否则会影响其他指令执行)，其它所有指令都是两字节的。**

设置断点：用INT 3代替用户程序原有指令，保存用户程序原有指令

发生断电：程序运行到断点，执行INT 3，进入断点中断服务程序

恢复执行：中断服务程序返回前，恢复用户程序原有指令，IP减1，返回后CPU从断点处继续执行

内部中断除单步中断外，都不可以用软件方法禁止（屏蔽）；单步中断可以通过软件设置TF位来允许或禁止。

除了单步中断，所有内部中断的优先级都比外部中断高。

INT指令——“INT n”，n=0~255，进行系统调用，可以直接调用中断服务程序

INT指令执行的操作：将FLAGS寄存器压栈，清除IF和TF，将CS和IP压栈，根据中断类型查找中断向量，将入口地址装入CS和IP

BIOS(Basic Input Output System)中断：装在0FE00H起的8KB ROM中，提供一些功能模块；可以通过INT n进行调用，如果需要的话，使用寄存器传递参数

中断号	功能号	功能	入口参数	出口参数
10H	0	设置显示方式	AL=11 640×480单色图形 =12 640×480彩色图形	
10H	2	设置光标位置	BH=页号 DH, DL=行, 列	
1AH	0	读时钟		CH:CL=时:分 DH:DL=秒:1/100秒
1AH	1	置时钟	CH:CL=时:分 DH:DL=秒:1/100秒	

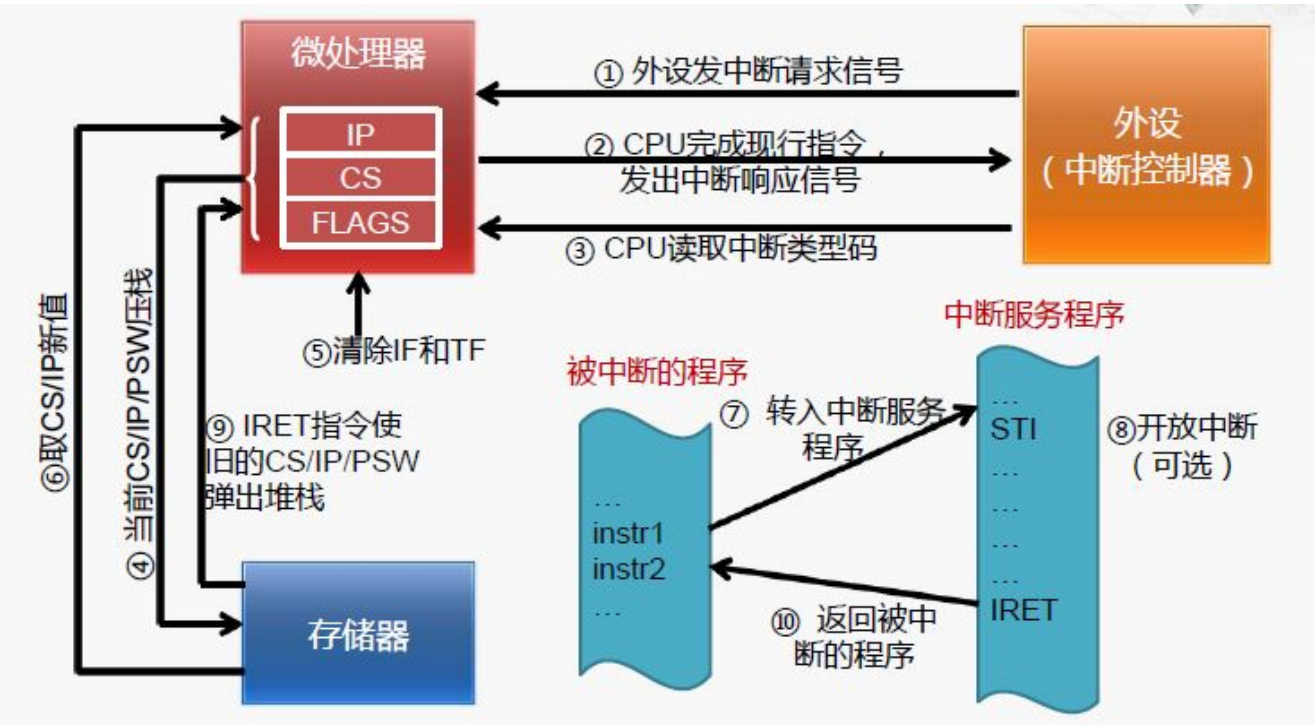
DOS中断：通过INT 21H进行调用，使用AH寄存器设置功能号，以便选择执行不同功能的代码，如果需要的话，可以通过寄存器传递参数；DOS中断比BIOS中断更加齐全、完整

功能号	功能	入口参数	出口参数
06H	直接控制台I/O	DL=FF（输入） DL=字符（输出）	AL=输入字符
09H	显示字符串	DS:DX=串地址 '$'结束字符串	
2CH	取时间		CH:CL=时:分 DH:DL=秒:1/100秒
2DH	设置时间	CH:CL=时:分 DH:DL=秒:1/100秒	AL=00H 成功 AL=FFH 无效

外部中断（硬件中断）——由CPU外部的终端请求信号启动的中断。x86 CPU提供两个引脚，NMI为非屏蔽中断引脚，INTR为可屏蔽中断引脚。

8259A是中断控制器，可以用于外部中断的控制。

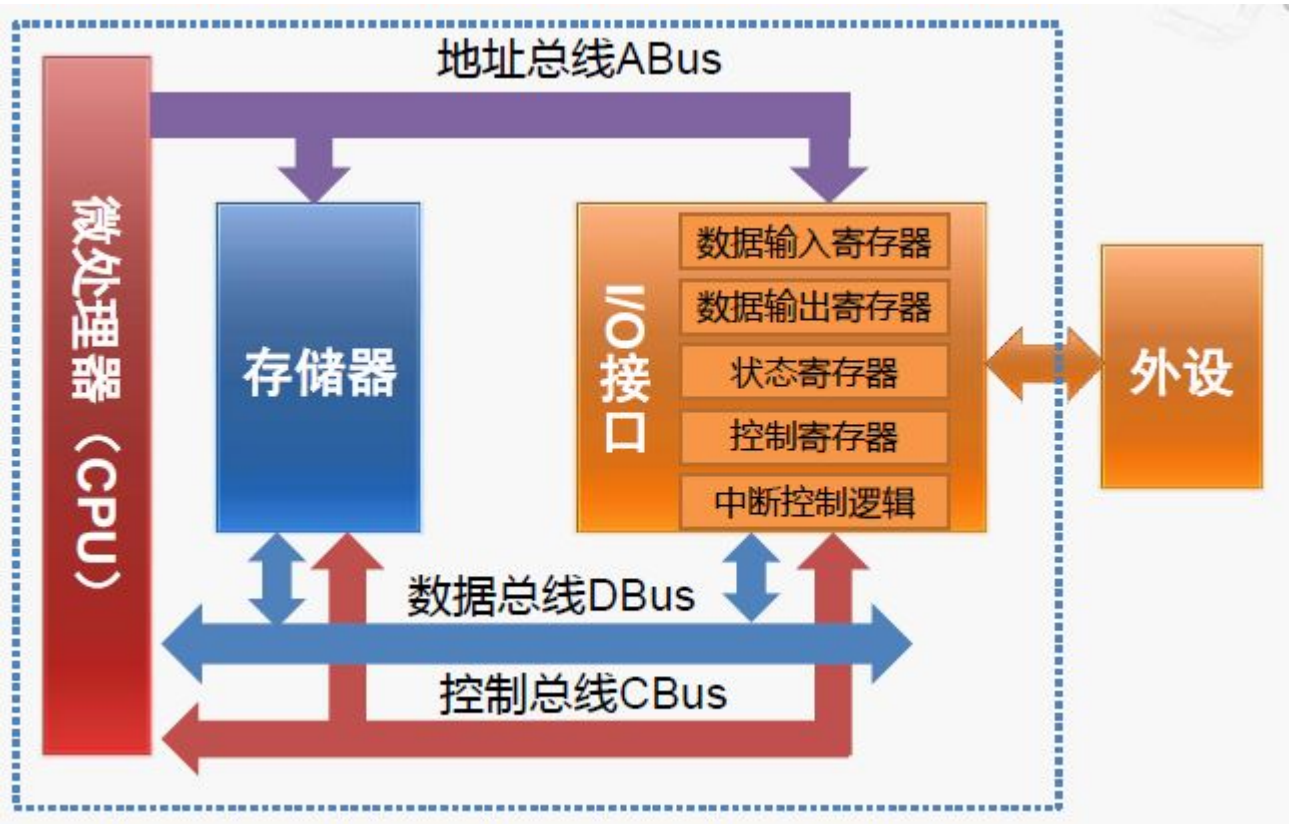
可屏蔽中断的处理过程如下



3. 输入输出设备

I/O接口产生的原因——CPU与外设之间存在比较大的速度差异；外设处理的信息格式和信号形式多样，CPU和外设间需要数据和信号转换

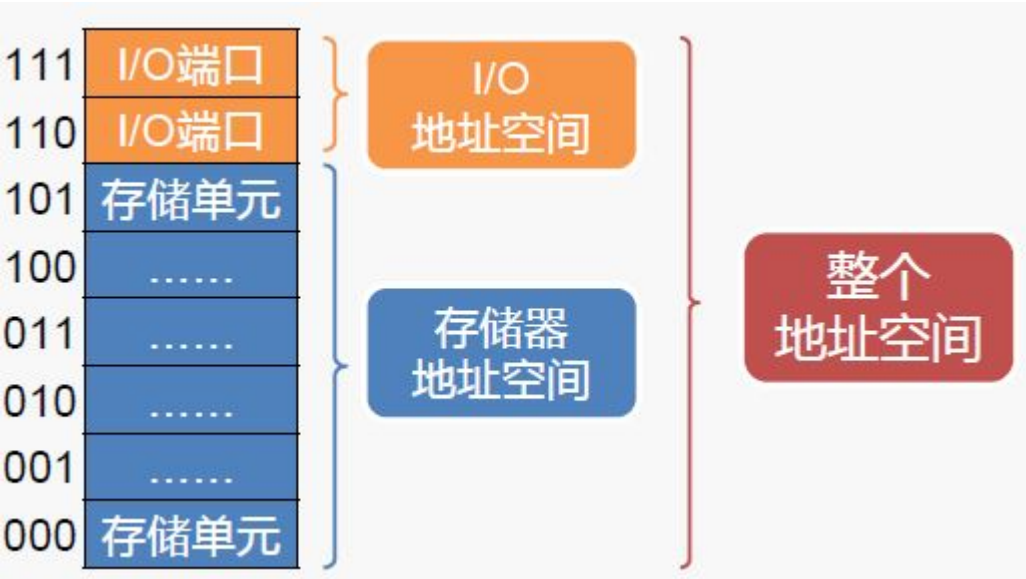
I/O接口的基本功能——数据缓冲，缓解CPU和外设的速度差异；提供联络信息，协调与同步数据交换；信号与信息格式转换，模-数/数-模转换，串-并/并-串转换，电平转换等；设备选择；中断管理；可编程功能。



I/O端口——I/O接口内包含的一组寄存器，每个I/O端口都有自己的端口地址（端口号），以便CPU访问

I/O编址方式——I/O与Mem分开编址方式，x86体系采用该方式；I/O与Mem统一编址方式，MIPS，ARM等采用该方式。

统一编址——地址上的一段被划分出，用于I/O编址。优点：可以使用访存指令来访问I/O端口，现成并且功能齐全；可以在CPU中将访存与I/O操作设计为同一套控制逻辑，简化内部结构。缺点：会使地址空间变小；利用访存指令进行I/O操作，会使得指令长度变长，指令执行时间也变长。



分开编址——Mem编址与I/O端口编址分开。优点：I/O端口不占用Mem，不会减少Mem地址空间；I/O指令编码短，执行快；I/O地址码短，译码方便...；缺点：...

I/O指令——IN指令，输入；OUT指令，输出

IN AC, PORT，将外设端口内容输入到AL或者AX

OUT PORT, AC，将AL或者AX的内容输出到外设端口

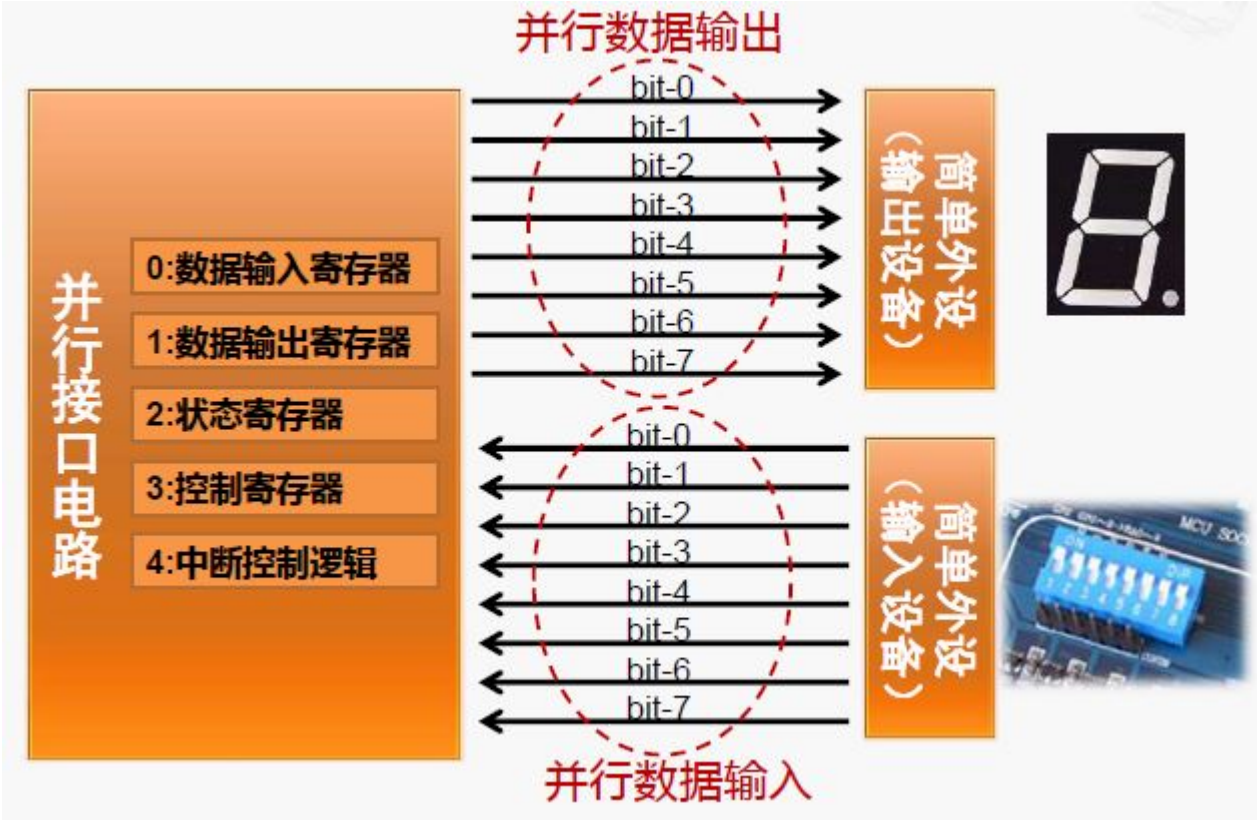
直接寻址方法：直接用一个字节的立即数指定端口地址，仅限0~255的端口地址。

间接寻址方法：使用DX的内容来制定端口地址。

I/O控制方式——CPU控制外设进行数据传送的方式，程序控制方式，中断控制方式，DMA方式。

程序控制方式——在程序控制下进行的数据传送方式。优点：对外设要求低，操作流程简单清晰；缺点：由CPU进行数据传送操作，占用大量运算资源。

无条件传送方式：假定外设时刻准备好，CPU不需要查询外设的工作状态，直接使用指令与外设传送数据。优点：控制程序简单；缺点：只能适用于很简单的外设的操作。

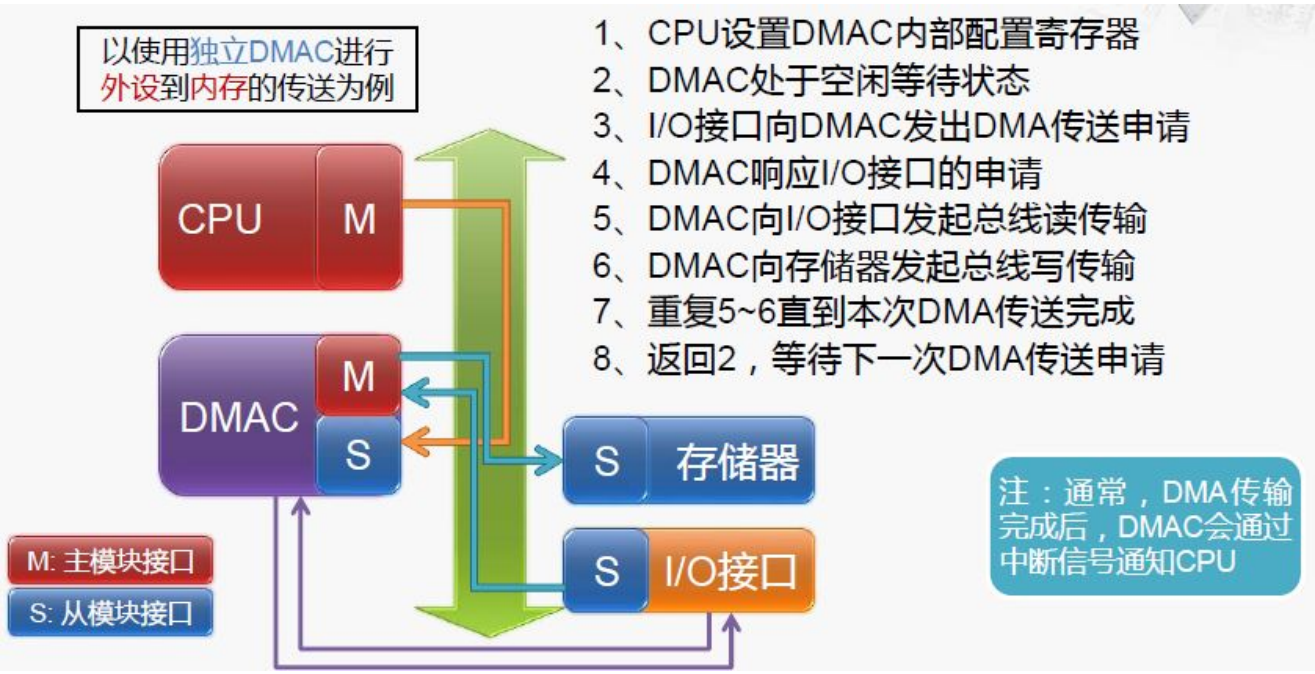


程序查询传送方式：CPU需要不断查询外设的工作状态，只有在确定外设已经准备就绪时才进行数据传送。除了并行数据输入和输出还需要“准备好”和“回答”这样一对握手信号，握手信号起到了定时协调与联络的作用。优点：比无条件传送方式准确可靠；缺点：查询状态会占用大量时间

中断控制方式——将CPU不间断的查询更换为由I/O接口发出中断信号，从而将CPU从不断的查询中解放出来，但是数据传送的任务仍然需要CPU完成。优点：CPU可以同外设并行工作，提高了工作效率；外设具有申请服务的主动权；一定程度上满足了I/O处理实时性的要求。缺点：外设和Mem之间的数据交换仍然需要CPU完成，使用数据传送指令，会占用宝贵的CPU运算资源；进入和推出中断服务程序需要额外的指令。

DMA方式（直接存储器访问）——由专门的硬件控制电路(DMAC)控制外设与Mem之间的直接数据传送，CPU不再需要干预数据传送过程。

DMA的基本工作步骤如下图



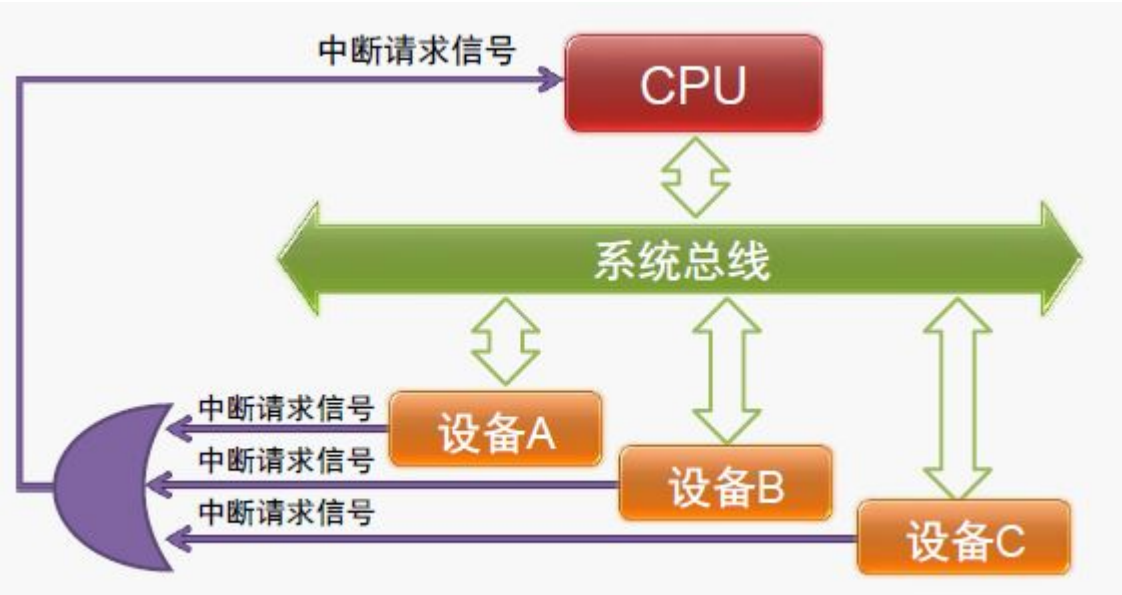
一般的，自带DMA的外设都是一些复杂的，数据传输量很大的外设，比如显卡。

4. 中断控制器和定时器

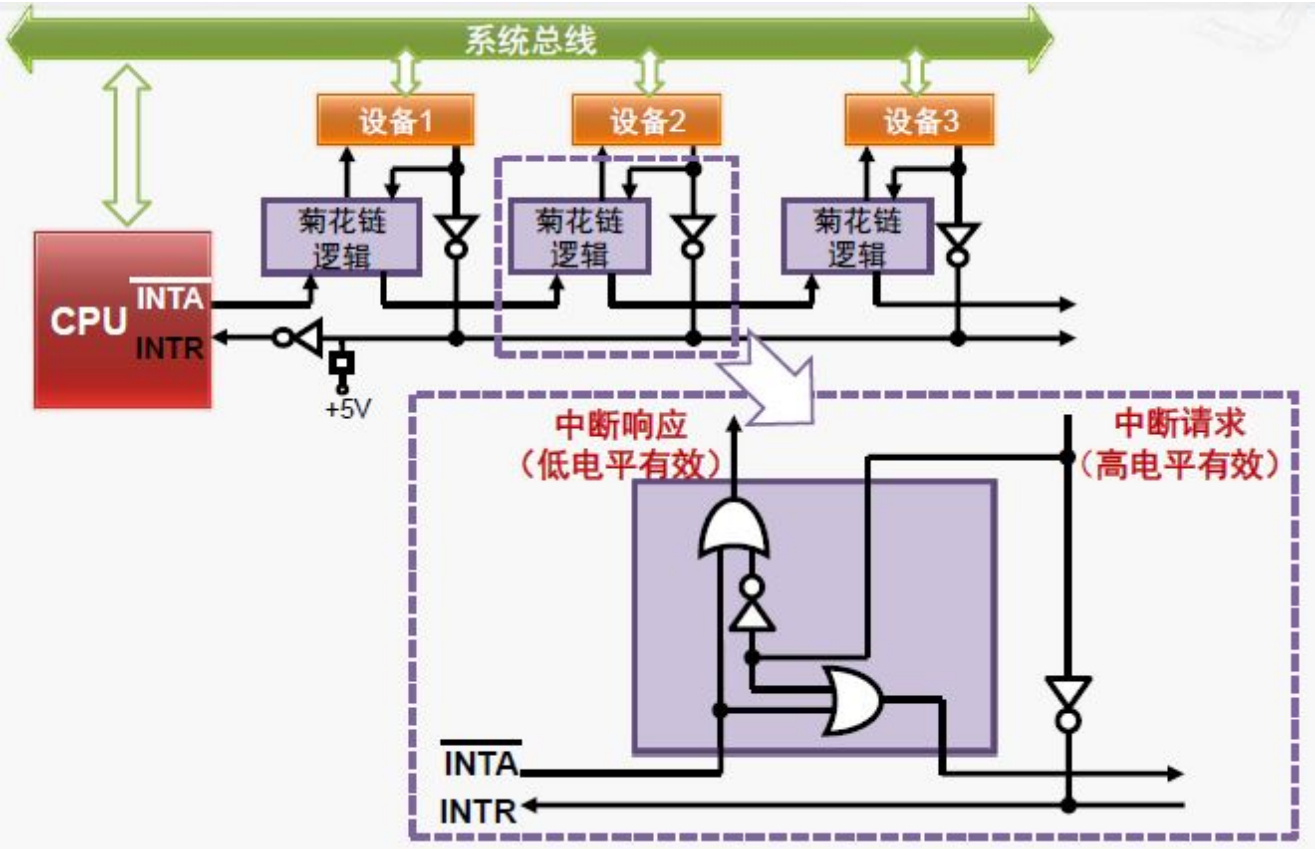
外部中断的优先级：多个中断请求同时出现，响应的次序

外部中断优先级确定：软件查询的方法，硬件中断优先级编码电路，可编程中断控制器

软件查询确定终端优先级——只需要少量硬件电路。将所有外部中断请求取或后产生一个信号从INTR送入CPU；中断服务开始部分，加入一段查询程序，先查询优先级高的设备，一般而言先查询速度较快或实时性较高的设备。

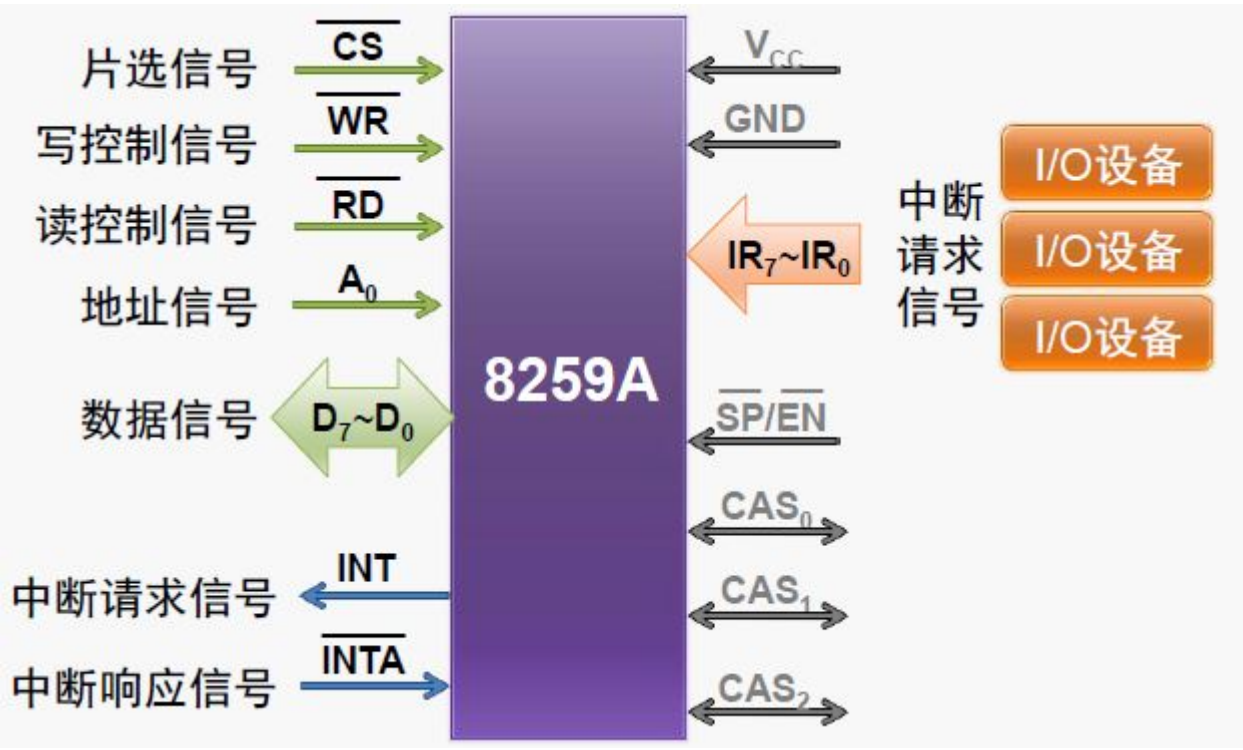


硬件中断优先级编码电路——菊花链优先级排队电路。在每个设备接口设置一个简单的逻辑电路，根据优先级顺序传递或者截留CPU发出的中断响应信号。



可编程中断控制器——管理和控制CPU的外部中断请求；判断中断优先级；为CPU提供中断类型码；选择屏蔽设备的中断请求。

8259A可编程中断控制器——28脚双列直插芯片。八位中断输入(IR7~IR0)，单片方式下，可以管理8级中断；级联方式下，主片连接8个从片，最大可以扩充到64级中断。



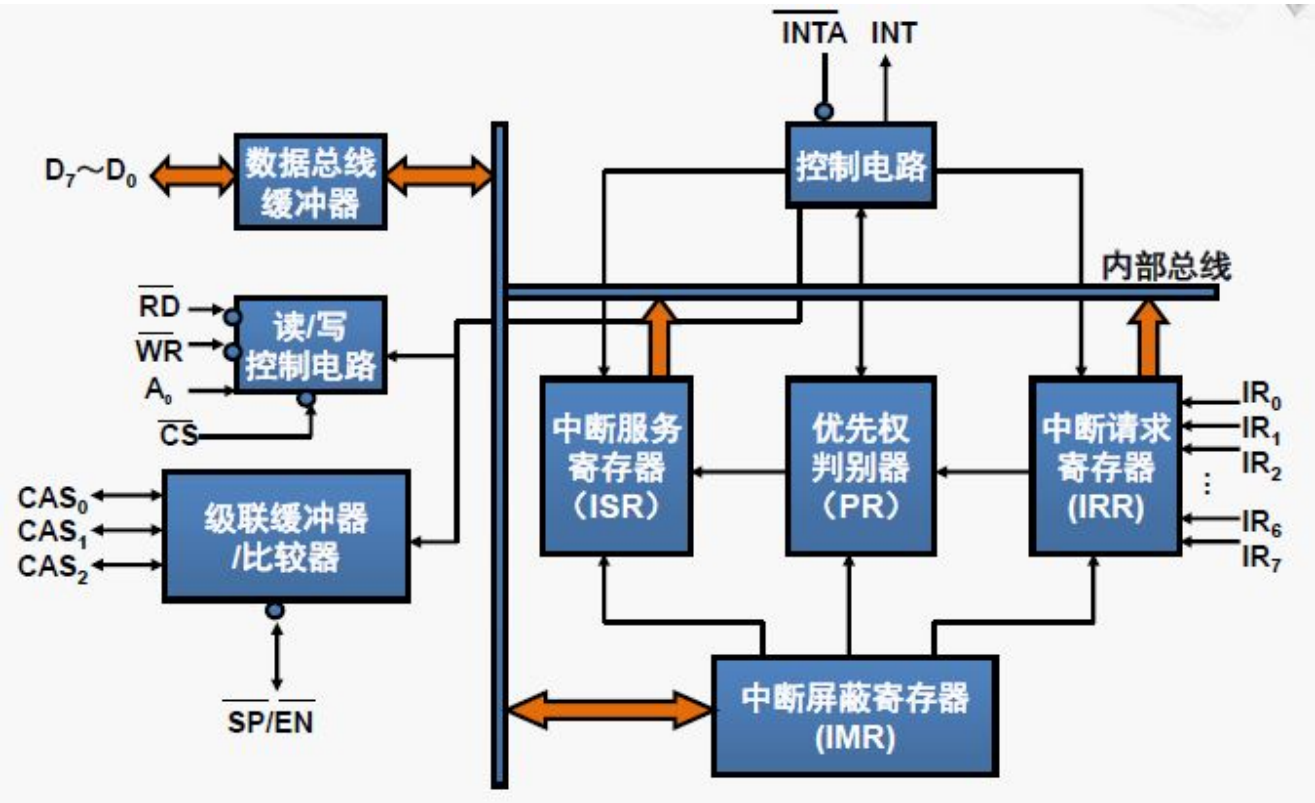
8259A内部结构

IRR——终端请求寄存器，8bit，可以接受并锁存来自IR7~IR0的中断请求

IMR——终端屏蔽寄存器，8bit，可以屏蔽IRR

PR——优先级裁决器，将新进入的中断请求和当前正在处理的中断的优先级进行比较

ISR——终端服务寄存器，8bit，记录当前正在处理的中断请求



IR7~IR0上出现某一个中断请求信号；IRR的对应位被置1；IMR相应位决定是否屏蔽，IMR屏蔽位=1则屏蔽，=0则不屏蔽；中断进入PR进行仲裁，若新进入的中断优先级高，则该中断请求被送到CPU；如果IF=1，则CPU在完成当前指令后，响应中断。

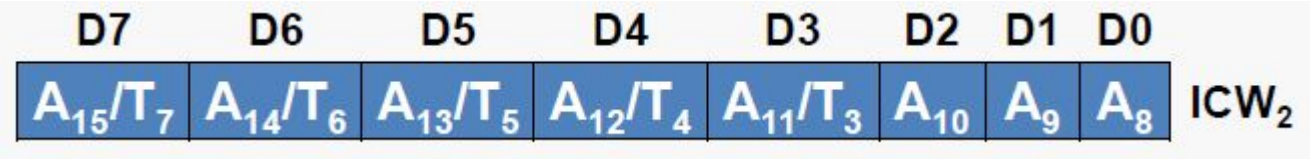
8259A的编程结构——ICW1~ICW4，初始化命令字，在系统初始化时设定依次，设置8259A的基本工作方式；OCW1~OCW3，操作命令字，在系统运行时可以多次编程设定，用来修改8259A的具体工作方式。这七个控制字可以视为7个可编程寄存器，只能使用一位地址A0来寻址。

初始化命令字需要按照规定的顺序进行设置；操作命令字的设置没有规定先后顺序，使用时可以根据需要灵活选择。

ICW1——使用偶地址端口，A0=0，用D4=1表示当前写入ICW1。IC4规定初始化时是否写入ICW4，0表示不写入，1表示写入；SNGL规定是否级联和是否写入ICW3，0表示级联方式并且要写入ICW3，1表示单片方式并且不要写入ICW3；LTIM规定中断检测方式，0表示边沿触发，1表示电平触发。

D7	D6	D5	D4	D3	D2	D1	D0	
A_7	A_6	A_5	1	LTIM	ADI	SNGL	IC_4	ICW_1

ICW2——使用奇地址端口，A0=1，紧跟在ICW1之后设置。D7~D3位确定中断类型码的高5位，D2~D0根据引脚号自动填入，IR0~IR7依次为000~111。

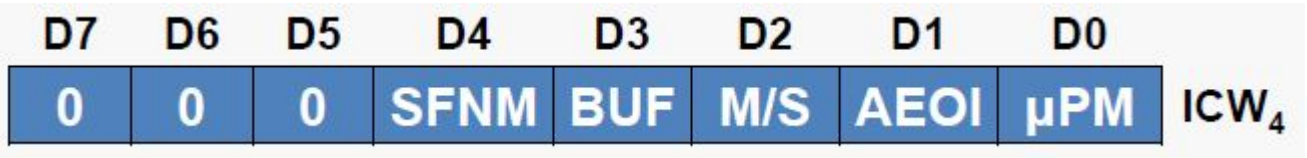


ICW3——使用奇地址端口，A0=1，只会在级联方式中使用。对于主片8259A，S7~S0中为1的位表示对应的IR引脚接有从片8259A；对于从片8259A，ID2~ID0表示本从片接在主片的哪一根IR引脚上。

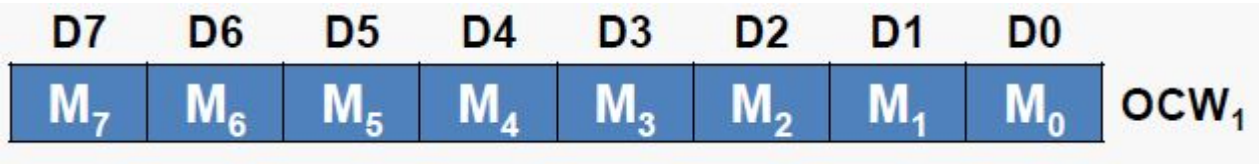
ICW4——使用奇地址端口，A0=1，由ICW1设置是否写入，80x86系统中，必须设置ICW4。μPM规定是那种系统，0表示8080/8085系统，1表示80x86系统；SFNM规定终端嵌套方式，0表示全嵌套方式，1表示特殊全嵌套方式；M/S规定本8259A是主片还是从片，0表示从片，1表示主片。

全嵌套方式——固定的优先级IR0最高→IR7最低，处理某一级中断时，只允许优先级更高的中断请求进入，初始化后默认进入这一方式。

特殊全嵌套方式——处理某一级中断时，不但允许更高优先级的中断进入，也允许同级中断进入，通过ICW4的SFNM设置；级联时，主片应该设置为该方式。



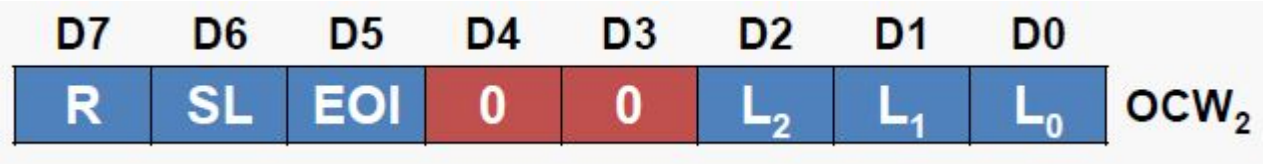
OCW1——使用奇地址端口，A0=1，实现中断屏蔽功能，OCW1的内容会被直接置入IMR中。



OCW2——使用偶地址端口，A0=0，D3D4=00，作为OCW2的标示位，设置优先级循环方式和中断结束方式。R规定中断优先级是否按循环方式，0表示非循环，1表示循环；SL规定L2L1L0是否有效，0表示无效，1表示有效；L2L1L0确定一个中断优先级的编码，000~111与IR0~IR7对应。

优先级自动循环方式——一个设备中断服务完成后，其优先级自动降为最低，而将最高优先级赋给原来比它低一级的中断请求。其实状态的优先级队列为IR0最高→IR7最低。

优先级特殊循环方式——可以设置其实状态的最低优先级，通过写入OCW2实现。

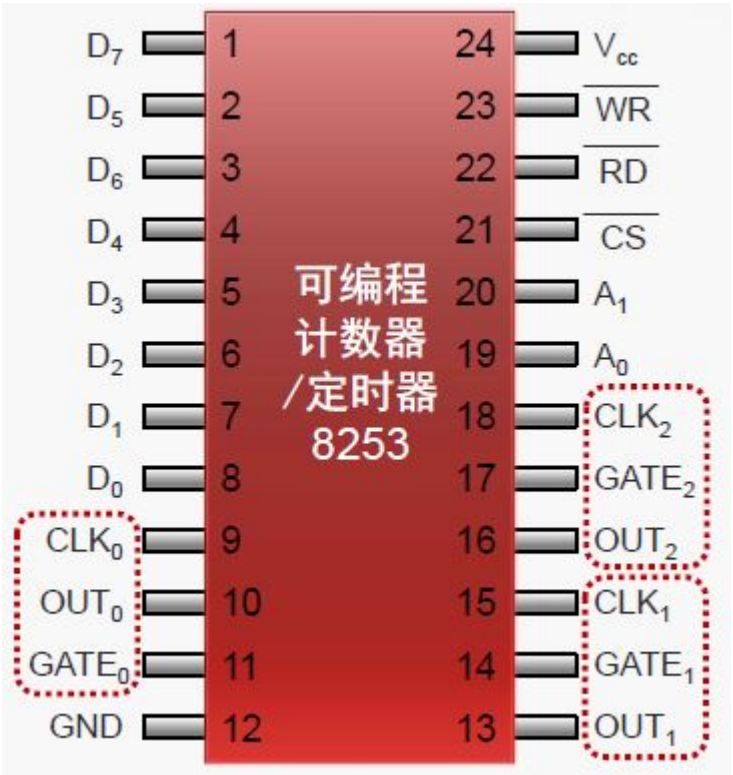


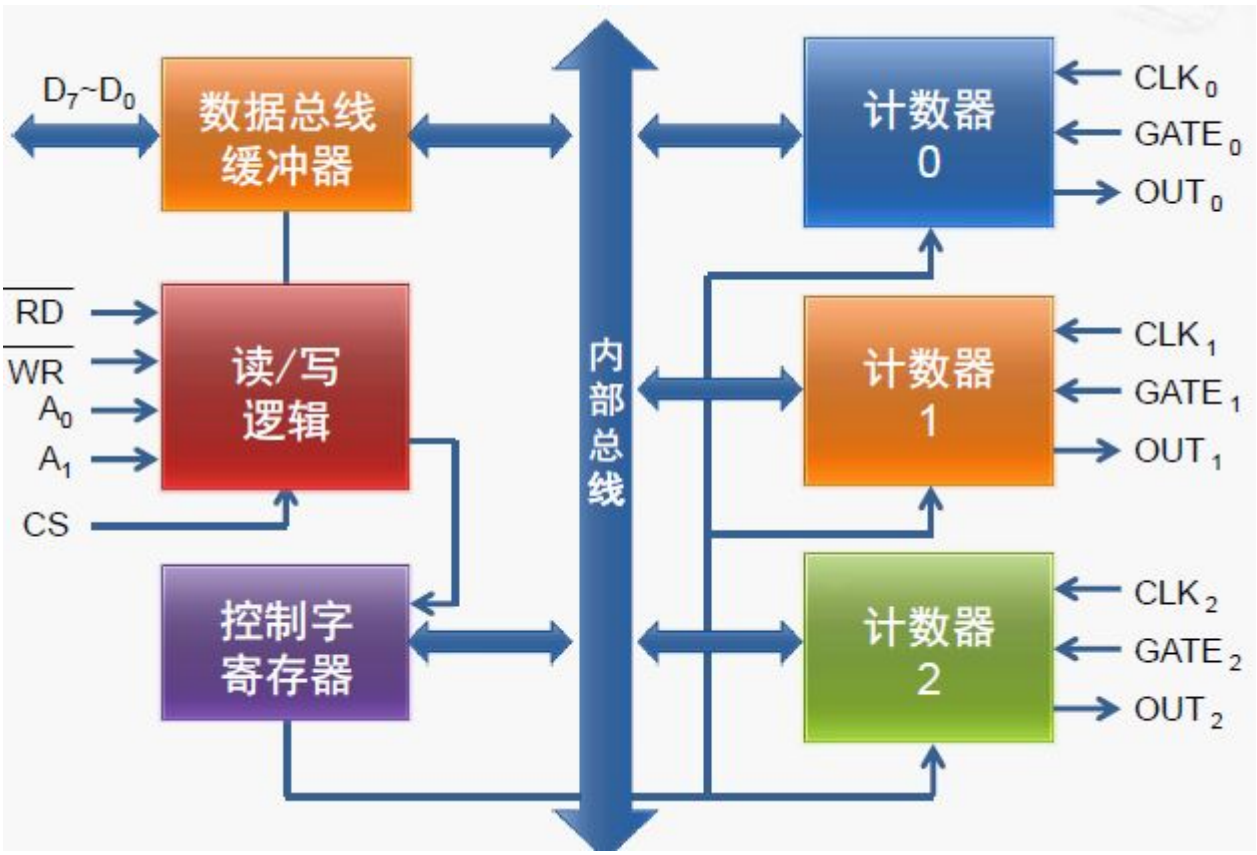
计数——对外部事件发生的次数进行计量

定时——指产生一段准确的时延，本质上是对固定的时间单位进行的计数

可编程计数器/定时器——采用软硬件相结合的定时计数方法，采用专用的定时电路，其定时值通过软件进行控制，功能灵活，使用方便。

可编程计数器/定时器8253——具有三个独立的16位计数通道；每个计数通道有6种工作方式；每个计数通道可以按照二进制或BCD码计数；每个计数通道的计数速率可达2MHz。





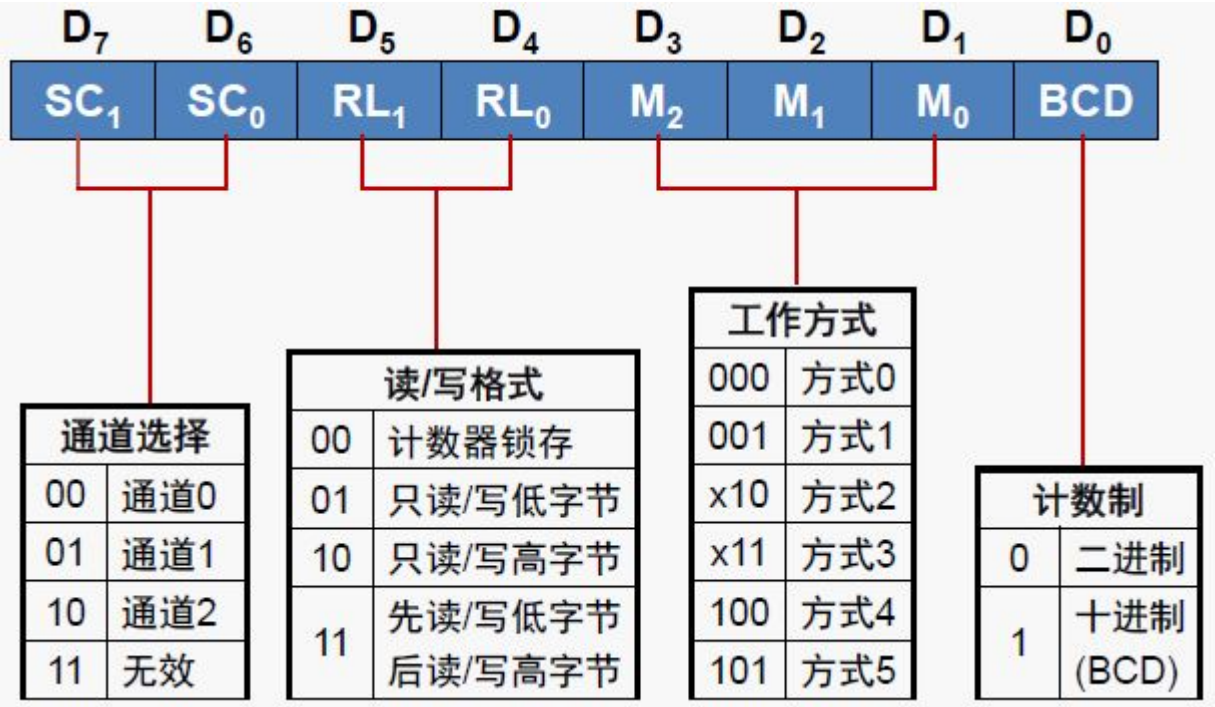
三个计数器为8253内的主要部件。CLK为时钟输入信号，作为计数脉冲，可以使非周期性脉冲，也可以是频率精确的周期性脉冲；GATE为门控输入信号，对计数过程进行控制，由工作方式而定；OUT为技术输出信号，计数到0/定时时间到则输出，具体由工作方式而定。

8253的工作方式：初始化写入控制字，初始化写入计数初值，计数初值装入执行部件，开始计数。

8253的内部寄存器寻址方式

$\overline{RD}$	$\overline{WR}$	A_1	A_0	寄存器选择和操作
1	0	0	0	写通道0计数初值寄存器CR_0
1	0	0	1	写通道1计数初值寄存器CR_1
1	0	1	0	写通道2计数初值寄存器CR_2
1	0	1	1	写控制寄存器
0	1	0	0	读通道0输出锁存器OL_0
0	1	0	1	读通道1输出锁存器OL_1
0	1	1	0	读通道2输出锁存器OL_2

设置控制字——选择计数通道，选择工作方式，指定计数初值长度和装入顺序，指定计数值的码制(二进制还是BCD)

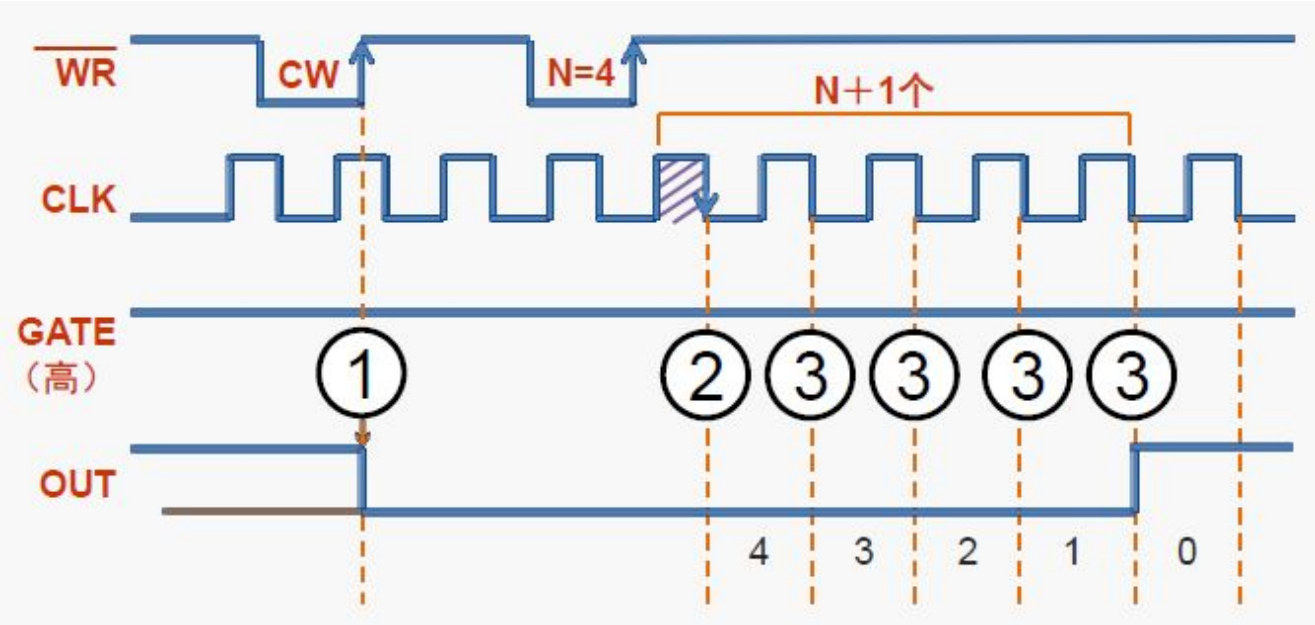


设置计数初值——向选定计数通道的CR写入计数初值，计数通道由A1A0确定。

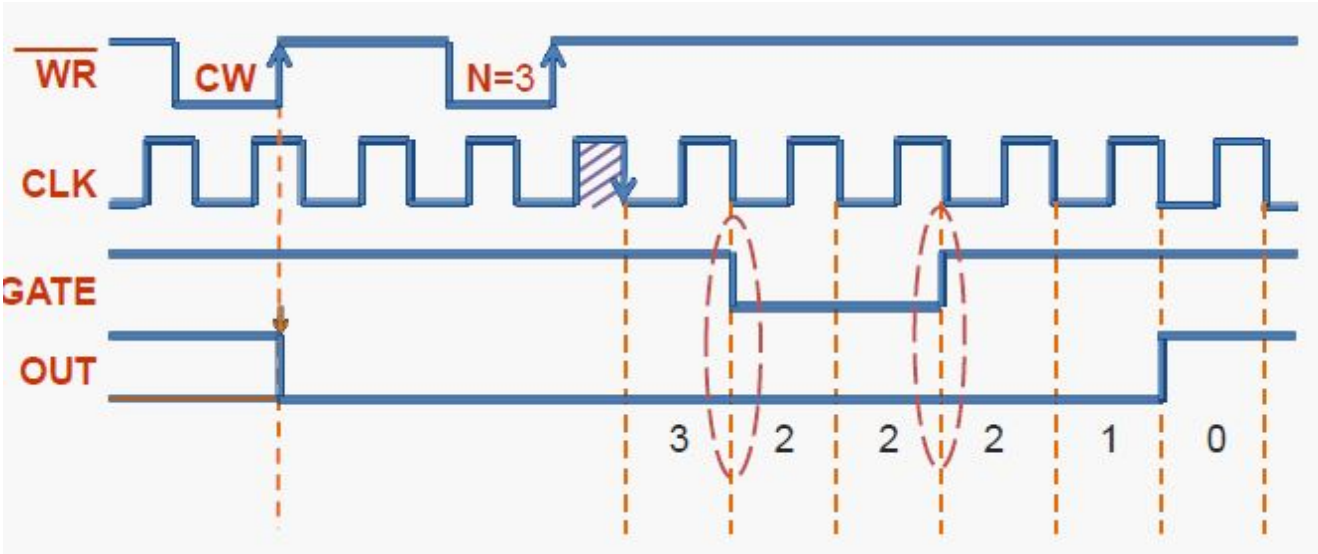
基本操作——写入控制字后，OUT端处于已知的初始状态（高/低电平）；写入计数初值CR后，经过一个CLK下降沿，CR装入CE，开始计数；没输入一个CLK脉冲，CE计数值-1；GATE电平控制和门控触发。

8253的常用工作方式

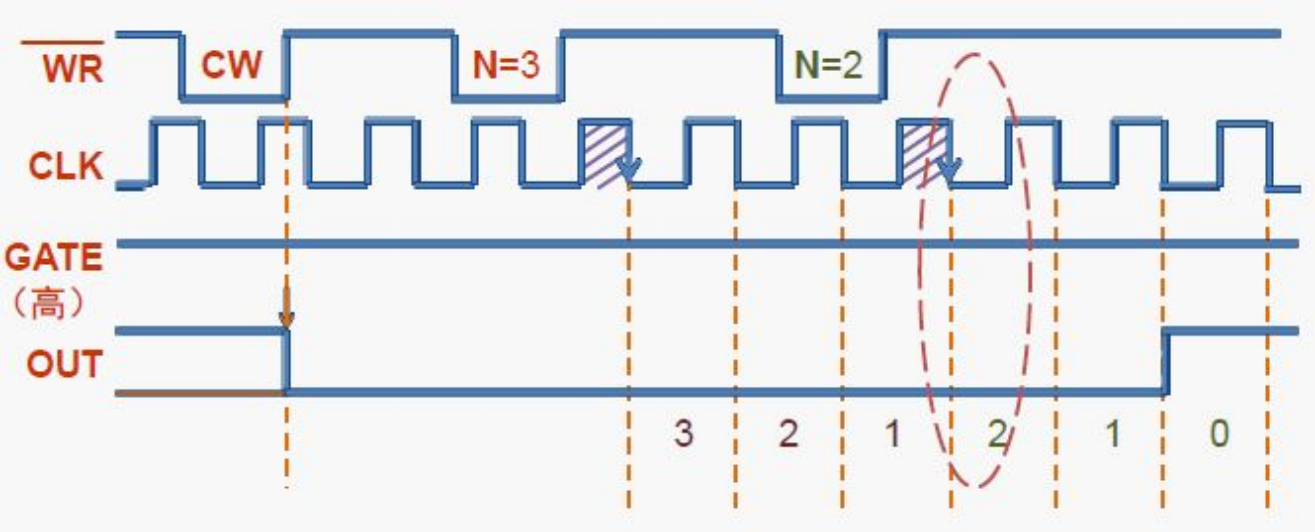
方式0——计数到0，产生中断请求。触发方式为软件启动，计数初值一次有效，不能自动重复；在计数结束时产生的上升沿可作为中断请求信号。可用于事件计数或倒计时。



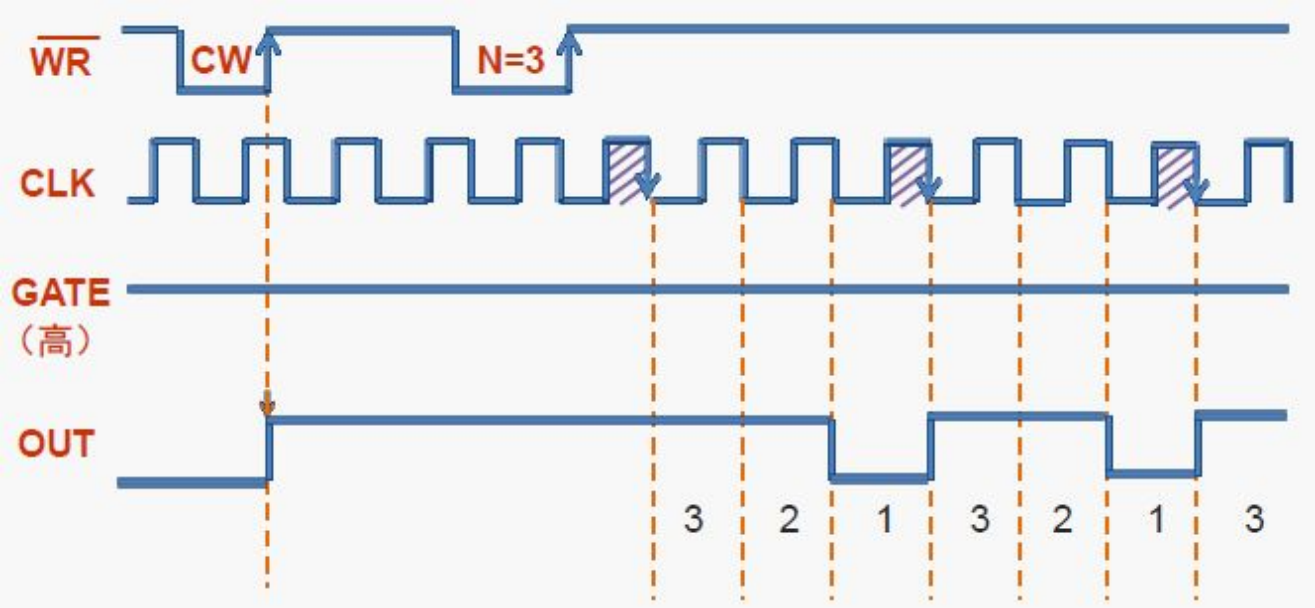
GATE可以控制计数操作，GATE变低时暂停计数，GATE变高时继续计数。



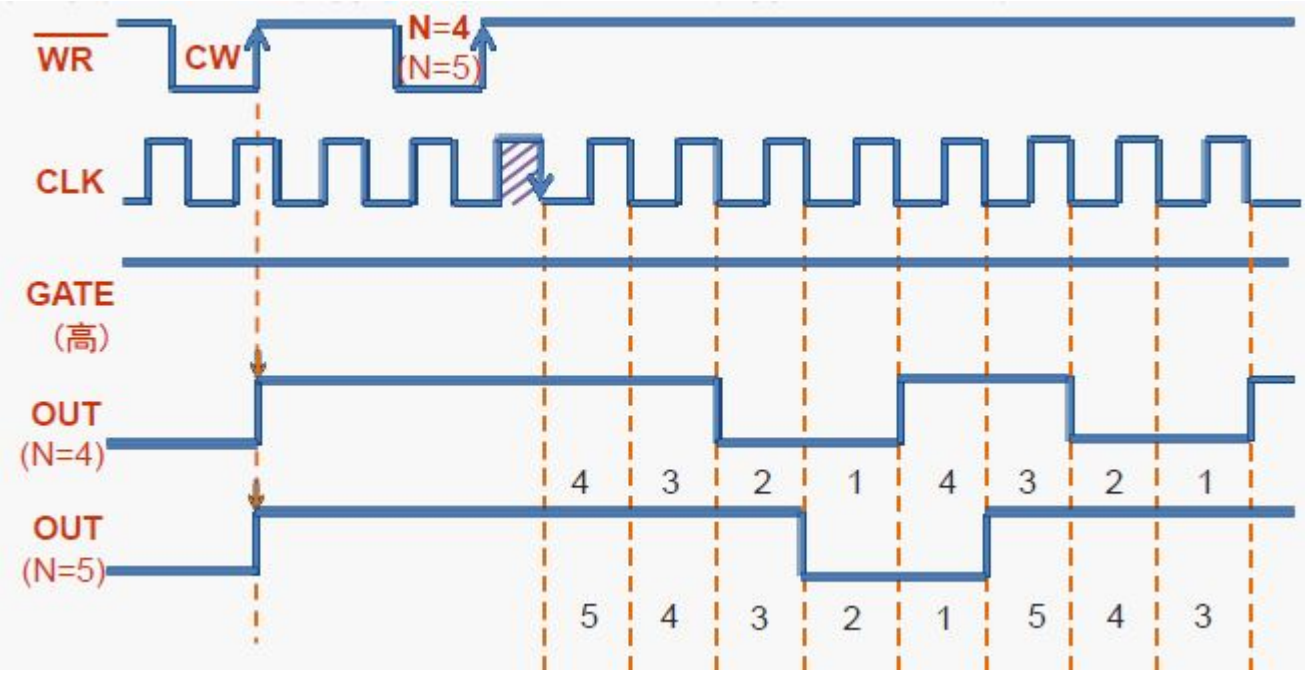
计数过程中重新写入计数初值，会立即生效并重新开始计数。



方式2——分频器。每输入N个CLK脉冲，输出宽度为1个CLK周期的负脉冲。触发方式为软件启动，会自动重复，周期性输出固定频率的脉冲。可用于产生固定频率的脉冲，比如进行DRAM的定时刷新。



方式3——方波发生器。输出对称方波或基本对称的矩形波。触发方式为软件启动，自动重复。可用于控制扬声器发声，产生系统时钟。



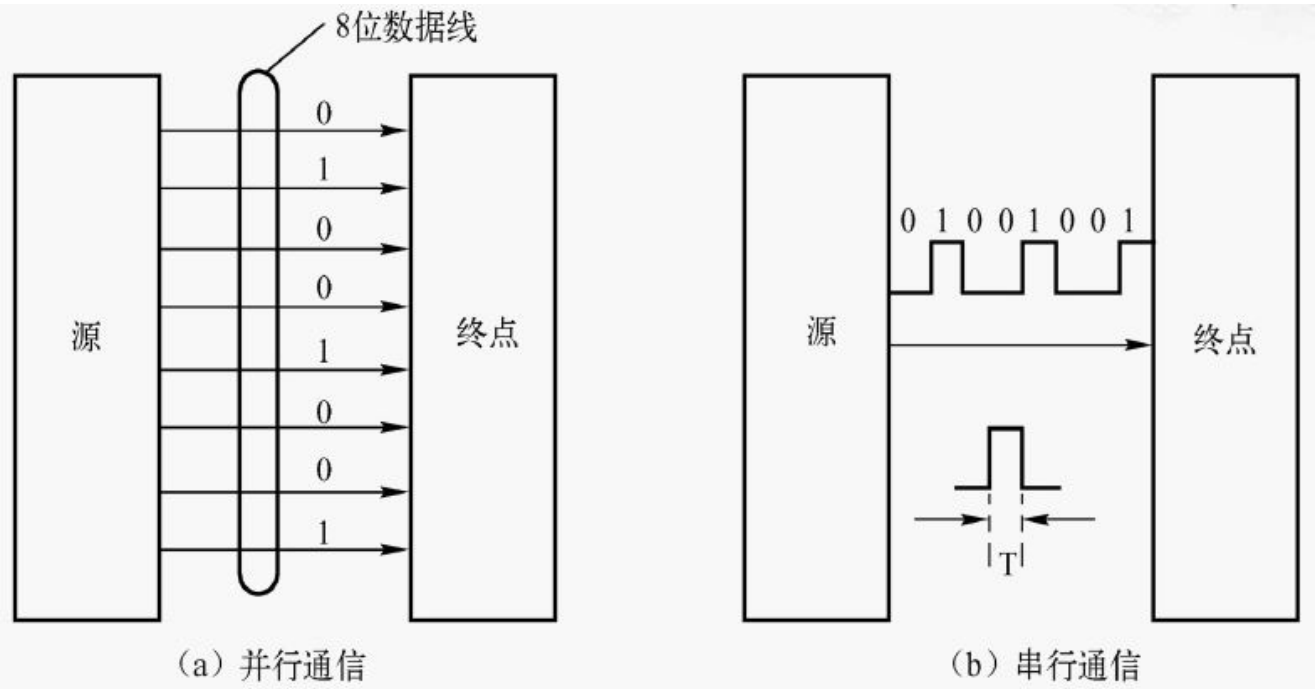
5. 并行和串行I/O接口

并行和串行的主要特点

串行通信——数据在单条一位宽的传输线上按照时间先后一位一位地传送

并行通信——数据在多位宽的传输线上各位同时进行传送

串行	传输线少:)	同频率下，数据传输率低:(	需要进行复杂的串/并转换:(	避免了信号线之间的串扰:)
并行	传输线多:(	同频率下，数据传输率高:)	无需串/并转换:)	存在信号线间的串扰:(



典型串口——RS-232接口（或称COM接口）

典型并口——IEEE-1284接口（或称LPT接口）

并行接口数据输出过程（程序控制方式）

① CPU执行OUT，将控制字写入接口控制寄存器，设置接口工作模式

② CPU执行OUT，将数据写到接口的输出缓冲寄存器

③ 接口将数据发送到并行数据输出信号，并将“输出准备好”信号置为有效(此步也可由CPU写控制字将该信号置为有效)

④ 外设发现输出准备好信号有效，从并行数据输出信号接收数据并将“输出回答”信号置为有效

⑤ 接口发现输出回答信号有效，将状态寄存器中的状态位“输出缓冲器空”置为有效

⑥ 这一过程中，CPU反复执行IN从接口“状态寄存器”中读出状态字，直至发现输出缓冲器空信号有效，之后开始下一输出过程

中断控制方式的变化：⑤ 接口发现输出回答信号有效，通过中断控制逻辑向CPU发出中断，并将状态寄存器中输出缓冲空状态位置为有效；⑥ CPU收到中断请求，进入中断服务程序，执行指令从状态寄存器读出状态自，发现输出缓冲空有效，因此开始下一输出过程

并行接口数据输入(程序控制方式)

① CPU执行OUT，将控制字写入接口的控制寄存器，设置接口工作模式

② 外设将数据发到并行数据输入信号，并将输入准备好信号置为有效

③ 接口发现输入准备好信号有效，从并行数据输入信号接收数据放入输入缓冲寄存器，并将输入回答信号置为有效，阻止外设继续输入

④ 接口将状态寄存器中输入缓冲器满置为有效

⑤ 上述过程中，CPU反复执行IN从状态寄存器中读出状态字，直至发现输入缓冲器满，之后执行IN从输入缓冲寄存器读出数据

⑥ 接口输入回答信号置为无效，等待外设输入新的数据

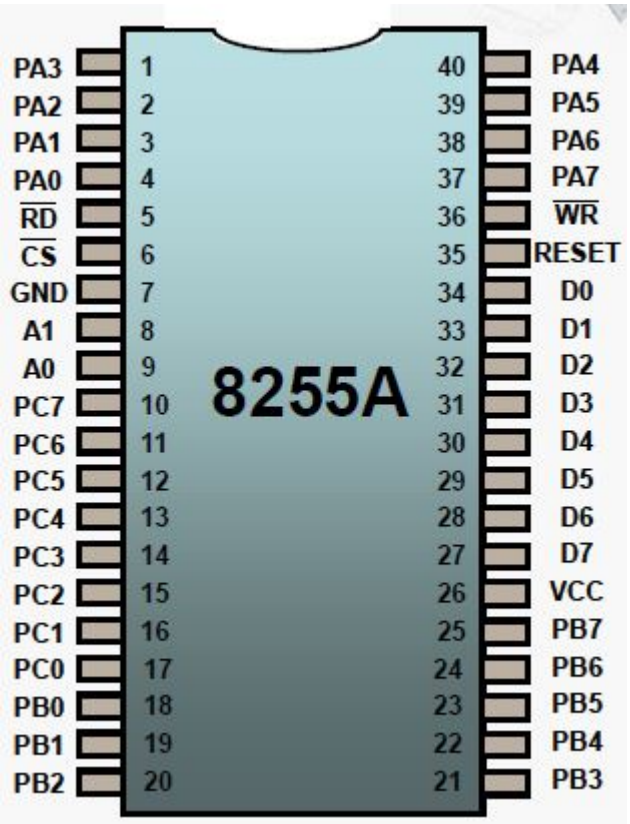
中断控制方式的变化：④ 接口通过终端控制逻辑向CPU发出中断请求信号，并将状态寄存器中的输入缓冲满置为有效；⑤ CPU收到中断请求，进入中断服务程序，执行指令从状态寄存器中读出状态字，发现输入缓冲满，执行IN从输入缓冲读出数据

Intel 8255A芯片——可编程并行接口电路；40脚直插封装，单一+55V；在连接外设时通常无需附加其他电路。

电源和地线——Vcc，GND

与外设的连线——端口A(PA7~PA0)，端口B(PB7~PB0)，端口C(PC7~PC0)

与系统的连线——复位(RESET)，数据线(D7~D0)，地址线(A1,A0)，控制(CS,RD,WR)

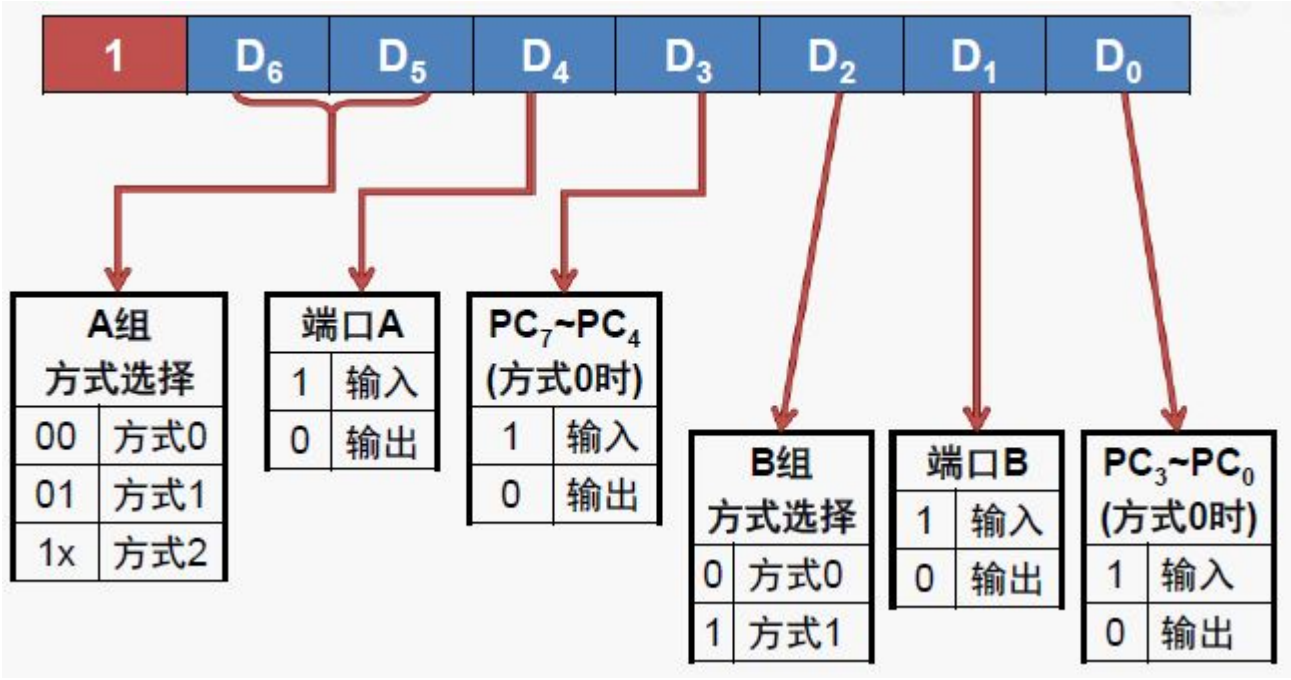


A组：端口A(8位)+端口C对应的握手信号(4位，PC7~PC4)；B组：端口B(8位)+端口C对应的握手信号(4位，PC3~PC0)。端口A和B可以被设定为输入输出端口，端口C被分为两组作为握手信号。

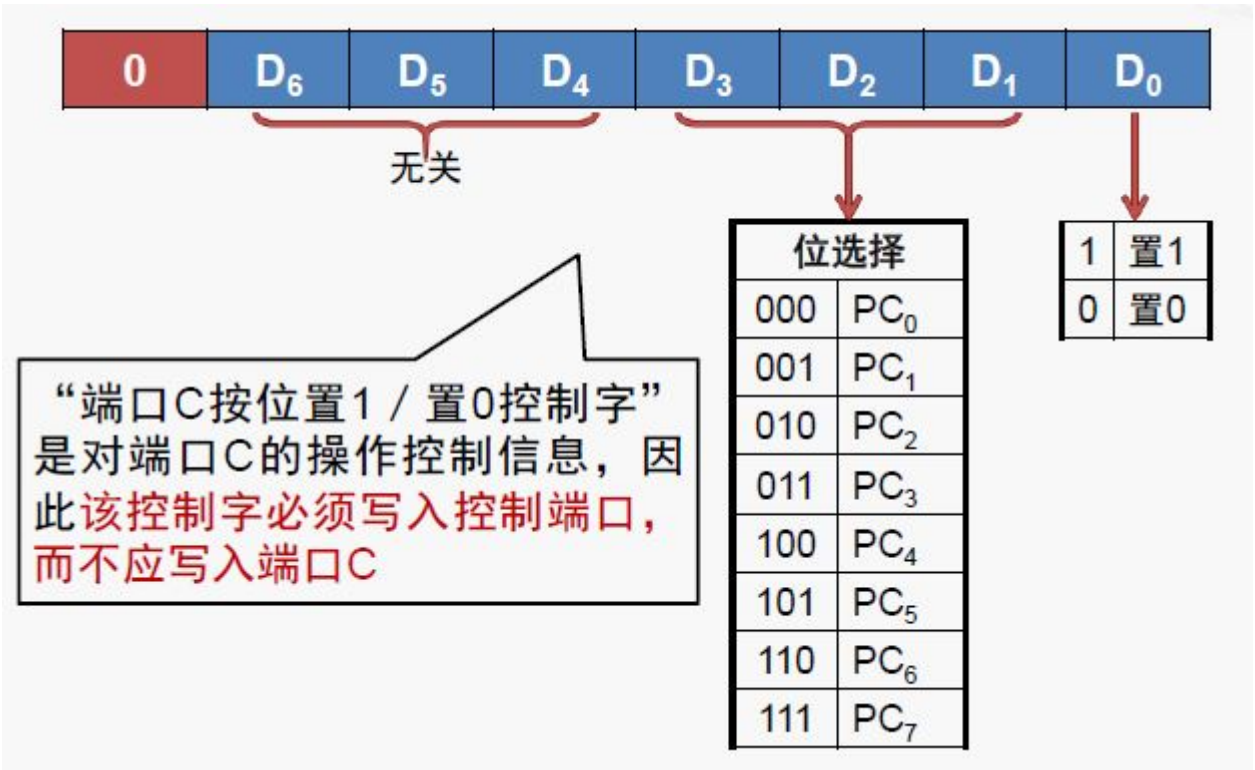
地址(端口选择信号)：当CS有效时，根据A0和A1从4个端口（可编程寄存器）中选择一个。A1=0，A0=0，选择端口A；A1=0，A0=1，选择端口B；A1=1，A0=0，选择端口C；；A1=1，A0=1，选择控制端口。

8255A控制字——方式选择控制字，端口C按位置0/置1控制字

方式选择控制字：确定端口A或B的工作方式，其中端口A具有方式1/2/3，端口B只有方式1/2。方式0，单向I/O，没有专门的握手信号；方式1，单向I/O，端口C专用于握手信号；方式2，双向I/O，端口C专用于握手信号。



端口C按位置0/置1控制字：对端口C的操作控制信息，因而该控制字必须写入控制端口，而不应写入端口C。



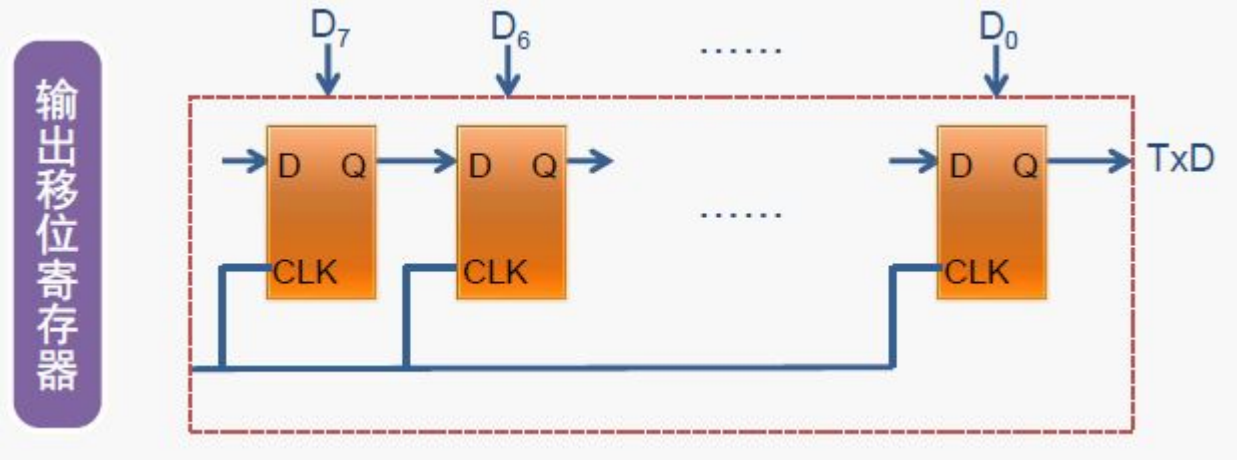
新型串行技术——差分信号传输技术。发送端在两根线上发送差分信号（振幅相等，相位相反）；接收端比较两个信号电压的差值，判断是逻辑0还是逻辑1。

优点：抗干扰能力强；抑制电磁干扰；时序定位准确。

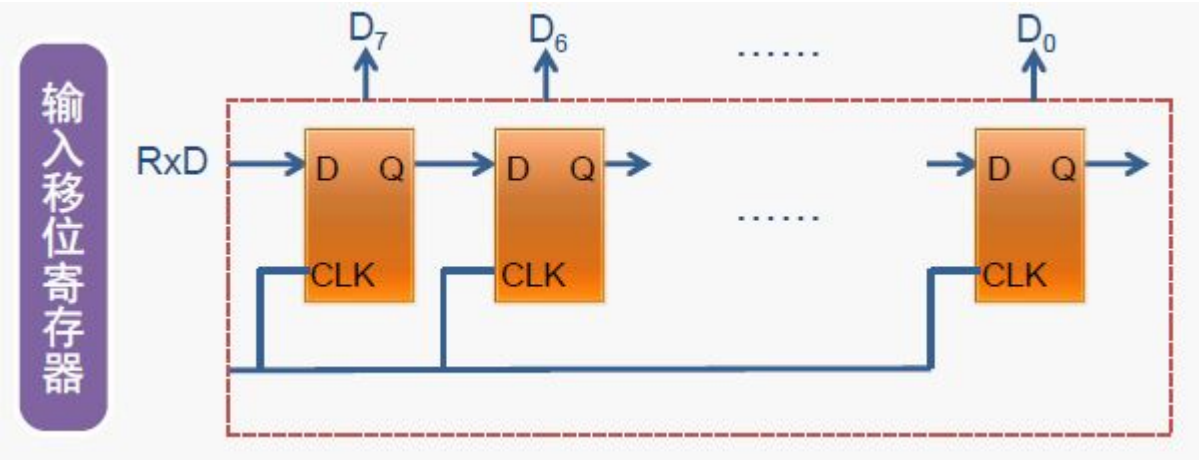
缺点：差分信号一定要走等长等宽紧靠且在同一层的线，布线难度高。

并-串/串-并转换

数据发送，并-串转换：将待发送的并行数据送入输出移位寄存器，输出移位寄存器在发送时钟的控制下，将数据逐位移出，放到串行输出线上。



数据接收，串-并转换：在接收时钟的控制下，对串行输入线进行采样，将采样到的数据逐位移入到输入移位寄存器。



单工——仅能在一个方向上进行数据传送

半双工——能在两个方向上进行数据传送，但是两个方向不能同时进行

全双工——在两个方向上同时进行数据传送

比特率——每秒传输的二进制位数，单位比特每秒，bps

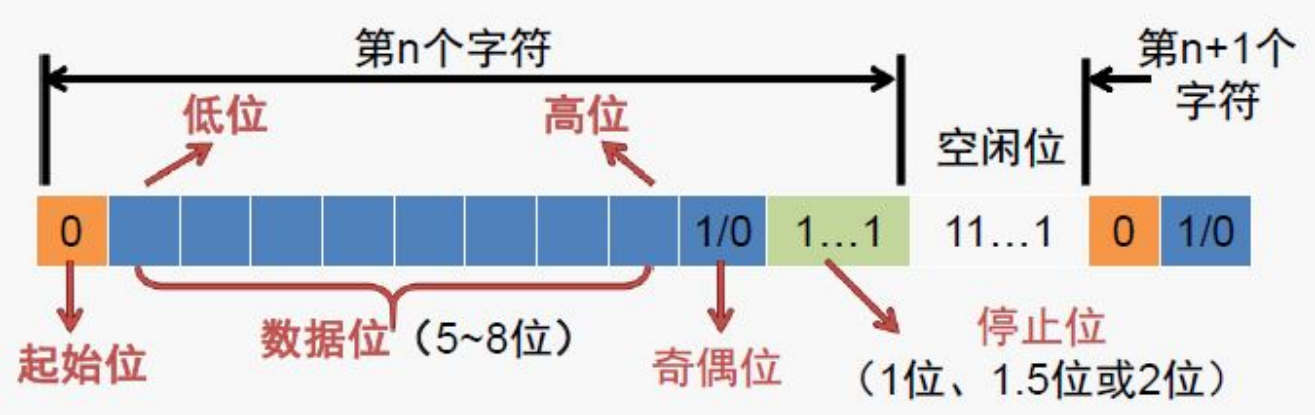
波特率——每秒传输的“符号”的个数，单位波特

计算机中，一个符号（高低电平）的信息量恰好为一比特，此时比特率与波特率一致；通信中采用“相-幅”复合调制技术，一个符号的信息量不是一比特，此时波特率不等于比特率。

波特率因子——时钟频率 = 波特率因子 × 波特率。

同步方式——为了使得串行信息传输准确，发送端和接收端动作必须相互协调配合。异步方式（起止同步方式），同步方式

异步方式：起始位标志着传送一个字符的开始；数据位为被传送字符的有效数据位；奇偶校验位在数据位之后的一位0/1，进行奇校验或偶校验；停止位标志着传送一个字符的结束。优点：传送每个字符的起始位重新定位，通信双方可以没有共同的时钟；缺点：附加信息位数多，通信开销大，效率低。



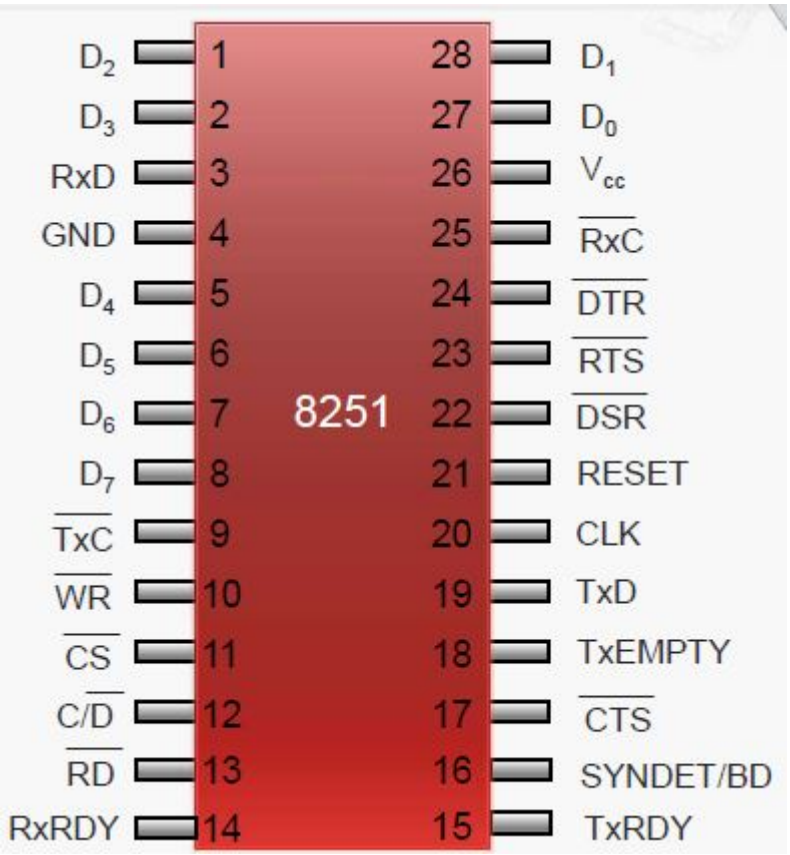
同步方式：数据传输以连续的字符串为单位，字符与字符间连续传送不留空隙，用同步字符表示数据发送开始；要求对传送数据每一位在收发两端严格同步，每个时钟周期发送一位数据，发送方在发送数据的同时需要把时钟信号也发送过去。

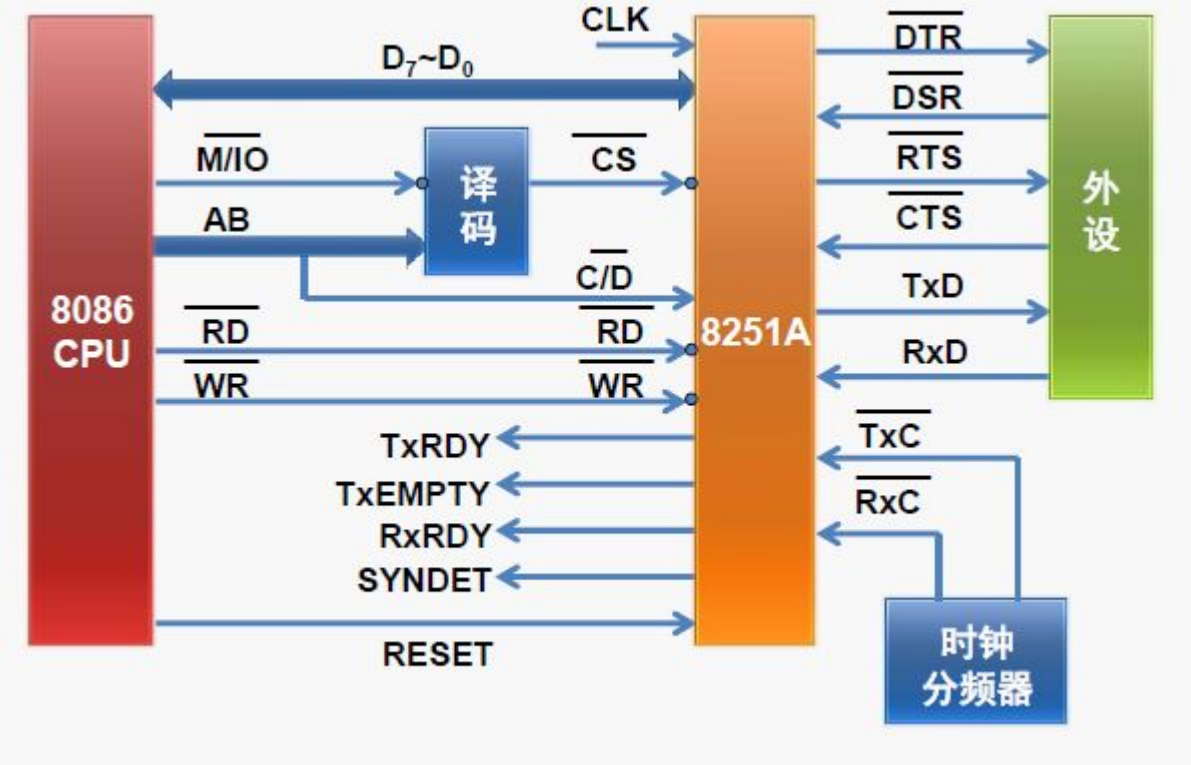
差错校验方法——奇偶校验，CRC校验

奇偶校验：在有效数据位中附加冗余校验位。奇校验使得整个信息位1的个数为奇数；偶校验使得整个信息位1的个数为偶数。优点：简单；缺点：检错能力低，不能纠错。

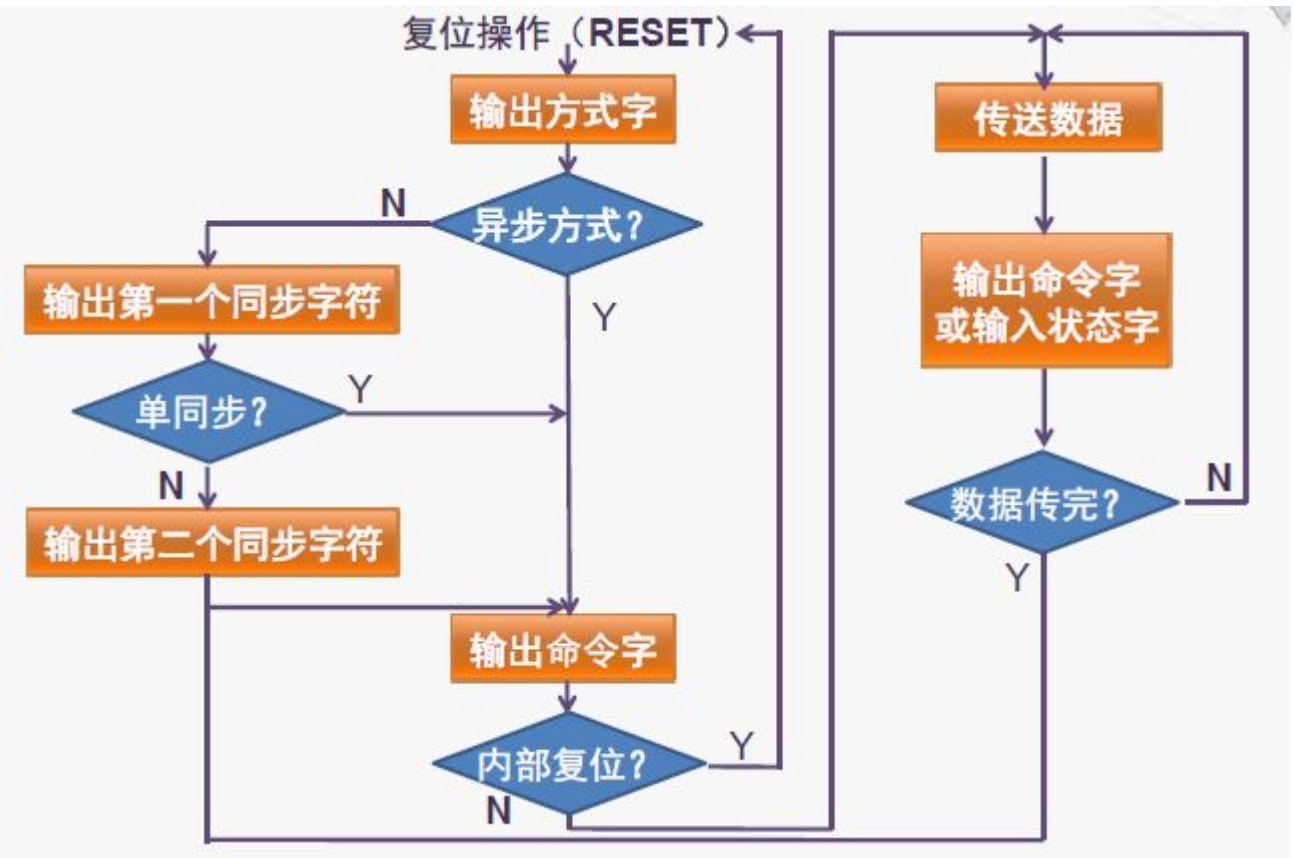
CRC校验：循环冗余校验。

Intel 8251A——可编程串行接口。两组引脚，一组与CPU的接口，一组与外设的接口。

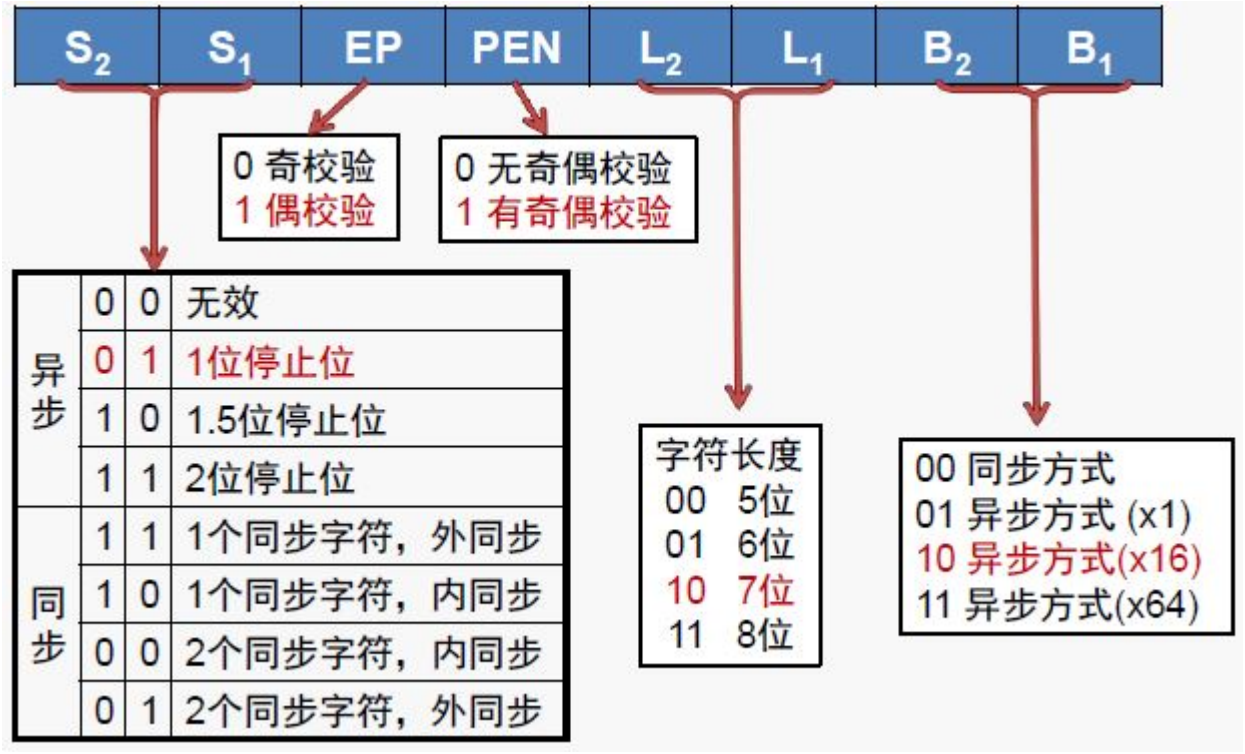




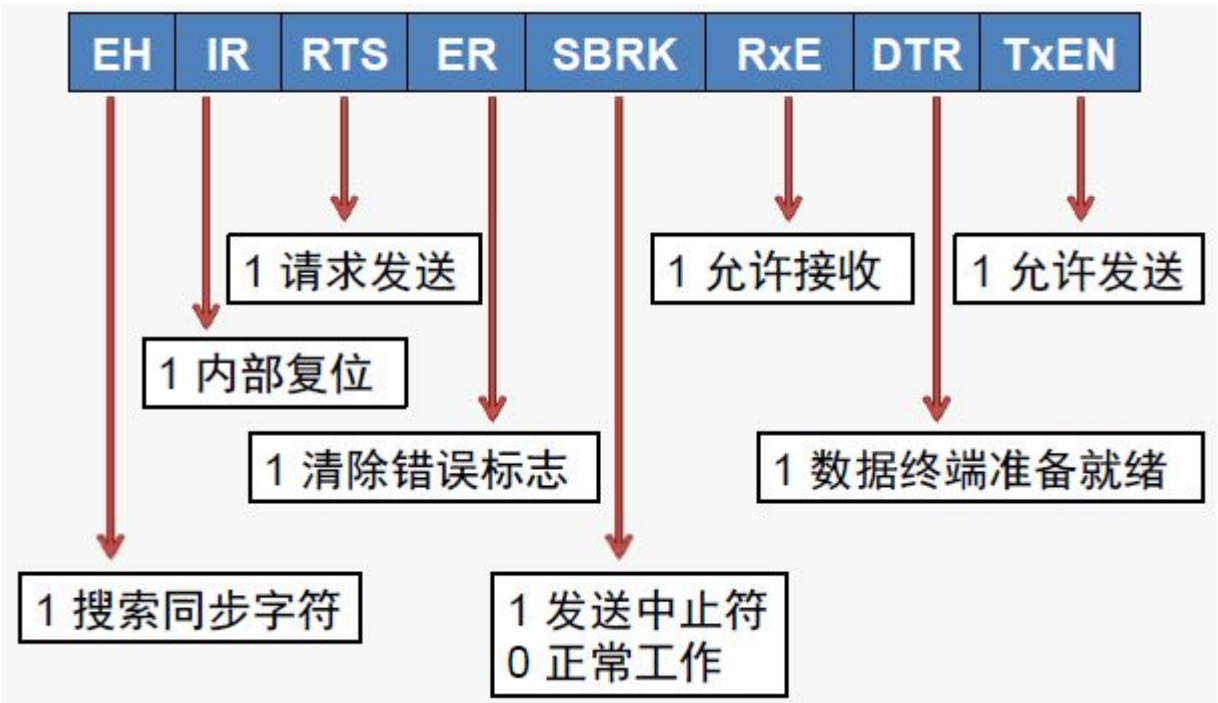
8251A编程流程——规定工作方式，方式选择控制字（方式字），设定一般工作特性；发出操作命令，操作命令控制字（命令字），制定具体操作。



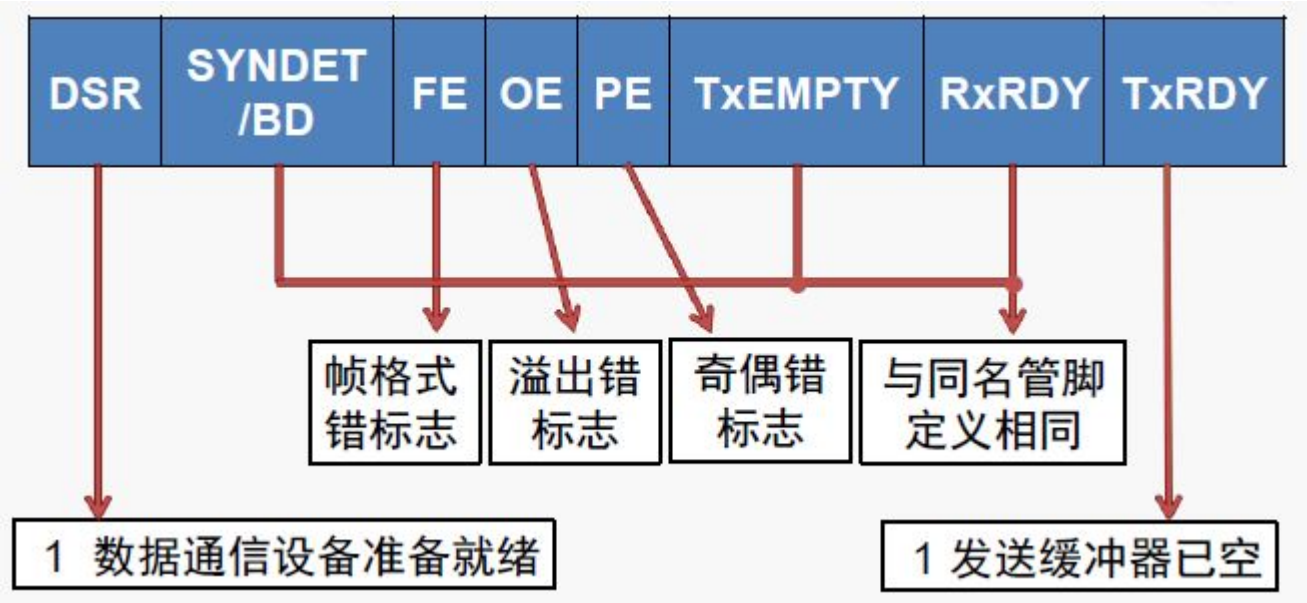
方式字：规定同步/异步，规定奇偶校验，规定字符长度，设定波特率


S2 S1 EP PEN L2 L1 B2 B1
0 奇校验
1 偶校验
0 无奇偶校验
1 有奇偶校验
异步
0 0 无效
0 1 1位停止位
1 0 1.5位停止位
1 1 2位停止位
同步
1 1 1个同步字符，外同步
1 0 1个同步字符，内同步
0 0 2个同步字符，内同步
0 1 2个同步字符，外同步
字符长度
00 5位
01 6位
10 7位
11 8位
00 同步方式
01 异步方式 (x1)
10 异步方式(x16)
11 异步方式(x64)

命令字


EH IR RTS ER SBRK RxE DTR TxEN
1 请求发送
1 内部复位
1 清除错误标志
1 允许接收
1 允许发送
1 数据终端准备就绪
1 搜索同步字符
1 发送中止符
0 正常工作

状态字



6. DMA技术

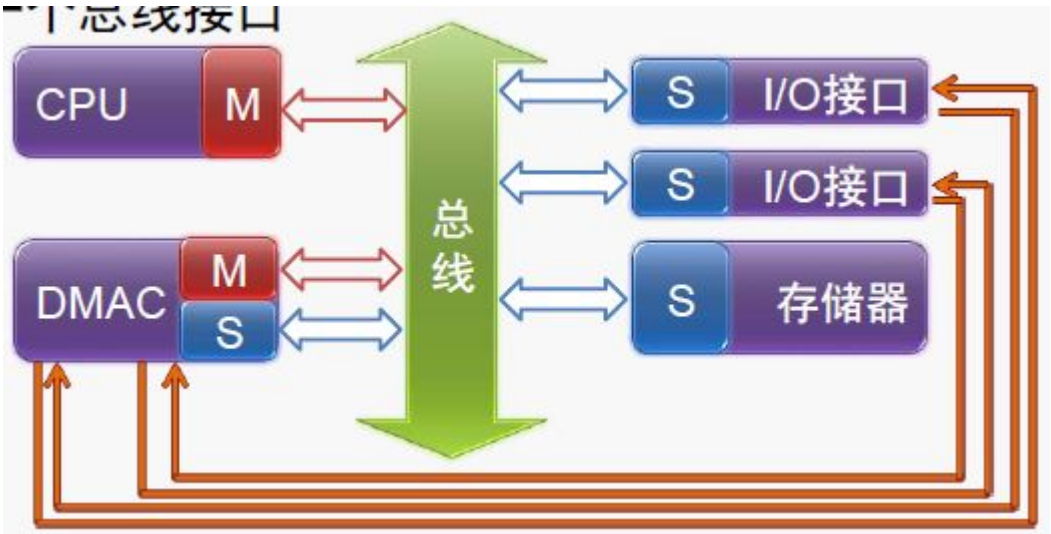
采用DMA的常见外设——USB，以太网，串口，PCI，PATA硬盘，SATA硬盘，显卡等

DMA传送的主要形式——内存与外设之间的传送；内存与内存之间的传送；外设与外设之间的传送。

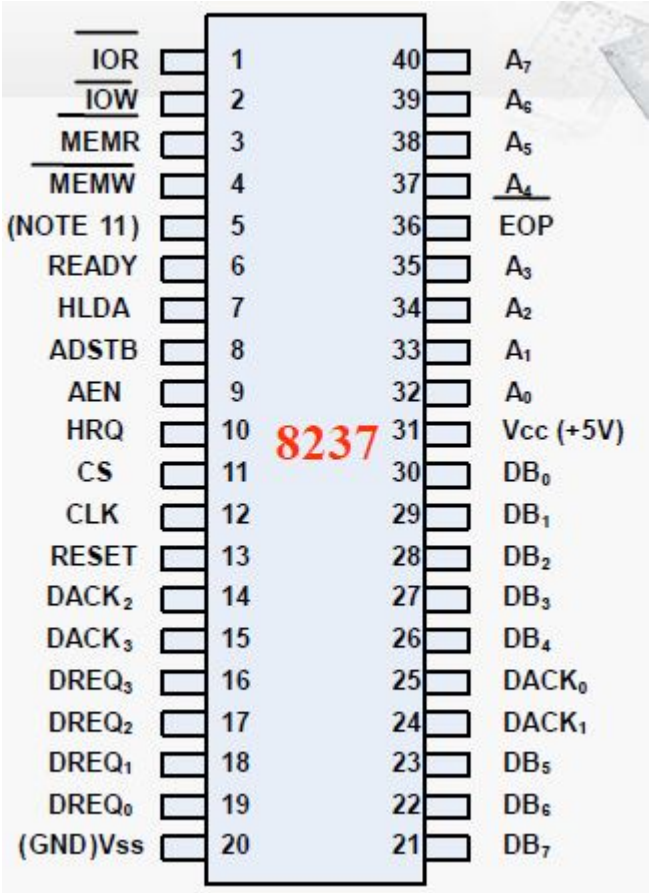
DMAC基本工作步骤（外设→内存）：CPU配置DMAC内部配置寄存器；DMAC空闲等待；I/O接口向DMAC发起DMA传送申请；DMAC响应I/O接口的申请；DMAC向I/O接口发起总线读传输；DMAC向Mem发起总线写传输；重复以上两步直至DMA传送完成；空闲等待直至下一次DMA传送申请。

DMA配置——由CPU完成，设置其工作模式，包括源地址初值及地址增减方式，目的地址初值及地址增减方式，待传送数据长度。**传送数据长度根据需要进行设置，也可以不设置**

DMAC多通道——DMAC可以同时为多个I/O结构提供服务。DMAC内部复制多套相同的配置寄存器，提供多套与I/O结构的申请/响应信号连接，共用一个总线接口。

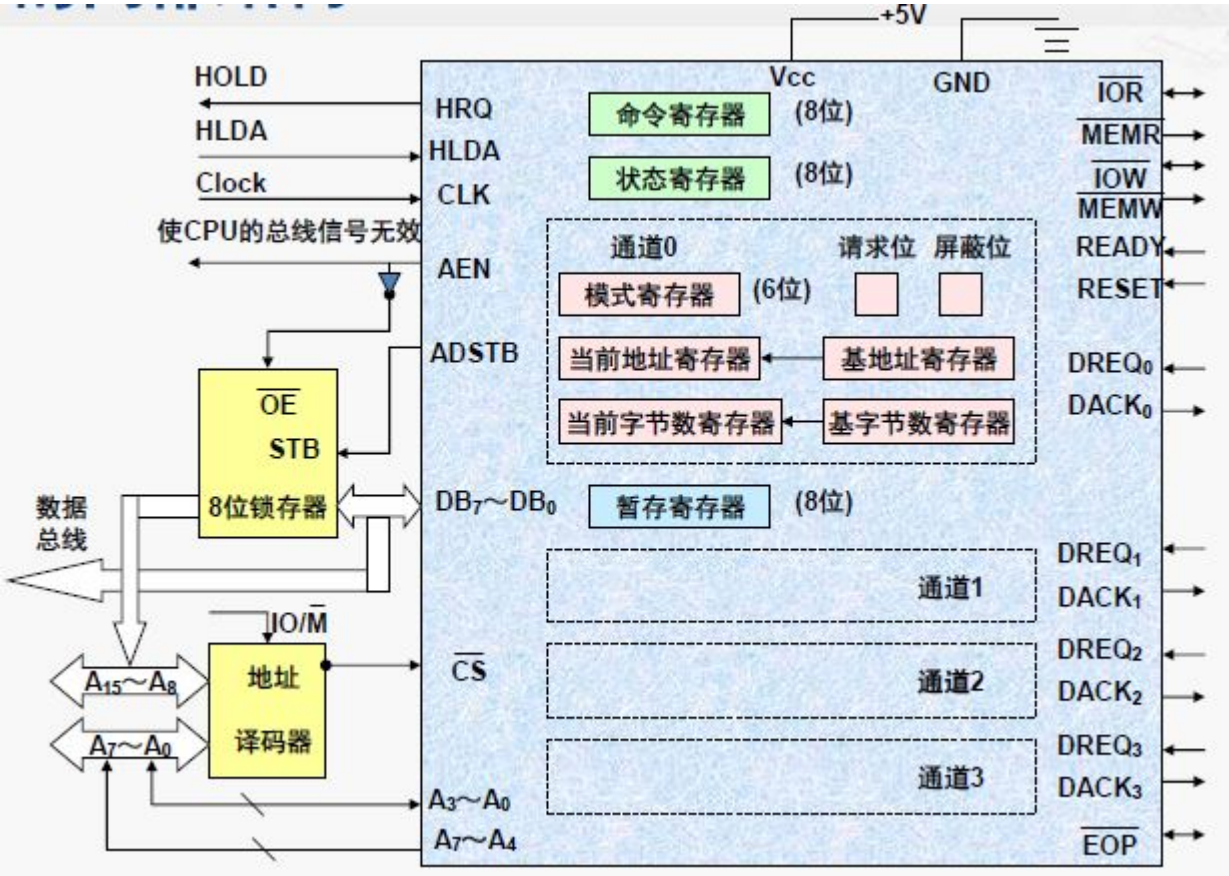


Intel 8237——可编程DMA控制器，现在不再是独立的单个芯片，被集成在南北桥芯片组中。8237是多通道的，一片8237内部有4个独立的DMA通道，每个通道一次DMA传送最大长度为64KByte。



每个通道的DMA请求都可以被允许或禁止；不同的通道的DMA请求有不同的优先级

四个通道可以分时地为四个外设实现DMA传送，也可以同时使用其中的通道0和通道1实现Mem到Mem的直接传送



可以用多片8237级联，构成更多的DMA通道；后裔级的HRQ和HLDA信号连在前一级的DREQ和DACK上

8237的工作方式

单字节传输方式：DMAC每次请求总线只传送一个字节，传送完成后立刻释放总线控制权

块传输方式（成组传输方式）：DMAC每次请求总线连续传送一个数据块，传送完成后再释放总线控制权

请求传输方式：每传输完一个字节，DMAC都要检测I/O接口发来的DMA请求信号是否有效，若有效则继续DMA传输，否则暂停传输并交还总线控制权，直至DMA请求信号再次有效，数据块传输则从刚刚暂停的位置继续进行

级联方式：主片设定在级联方式，从片设定为前三种方式。

8237的内部寄存器

基地址寄存器和基字节计数寄存器，只读不可写的，初始化时将本次传输的起始地址和需要传输的总字节数写入，传输过程中这两个寄存器不变。

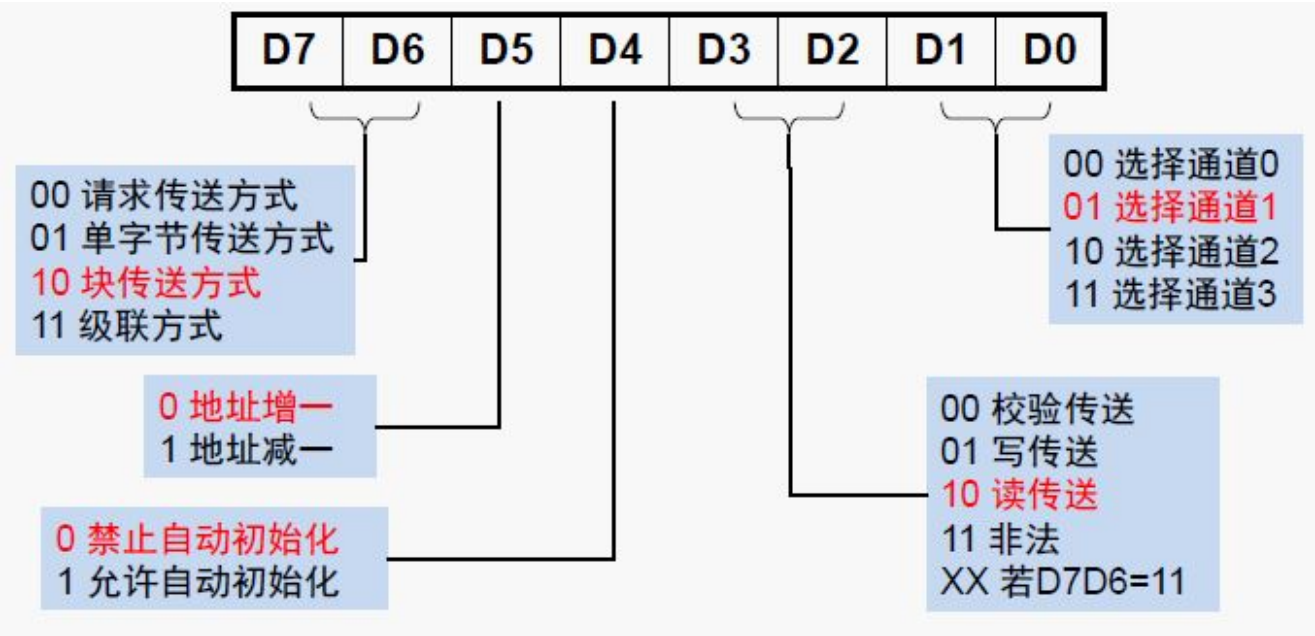
当前地址寄存器和当前字节计数寄存器，可读可写，初始化时，在写入基地址寄存器和基字节计数寄存器的同时也写入了当前地址寄存器和当前字节计数寄存器，传输过程中8237自动更新这两个寄存器。

寄存器名称	位数	数量	CPU访问方式
基地址寄存器	16位	4	只写
基字节计数寄存器	16位	4	只写
当前地址寄存器	16位	4	可读可写
当前字节计数寄存器	16位	4	可读可写
临时地址寄存器	16位	4	不可访问
临时字节计数寄存器	16位	4	不可访问
模式寄存器	6位	4	只写
命令寄存器	8位	1	只写
屏蔽寄存器	4位	1	只写
请求寄存器	4位	1	只写
状态寄存器	8位	1	只读
暂存寄存器	8位	1	只读

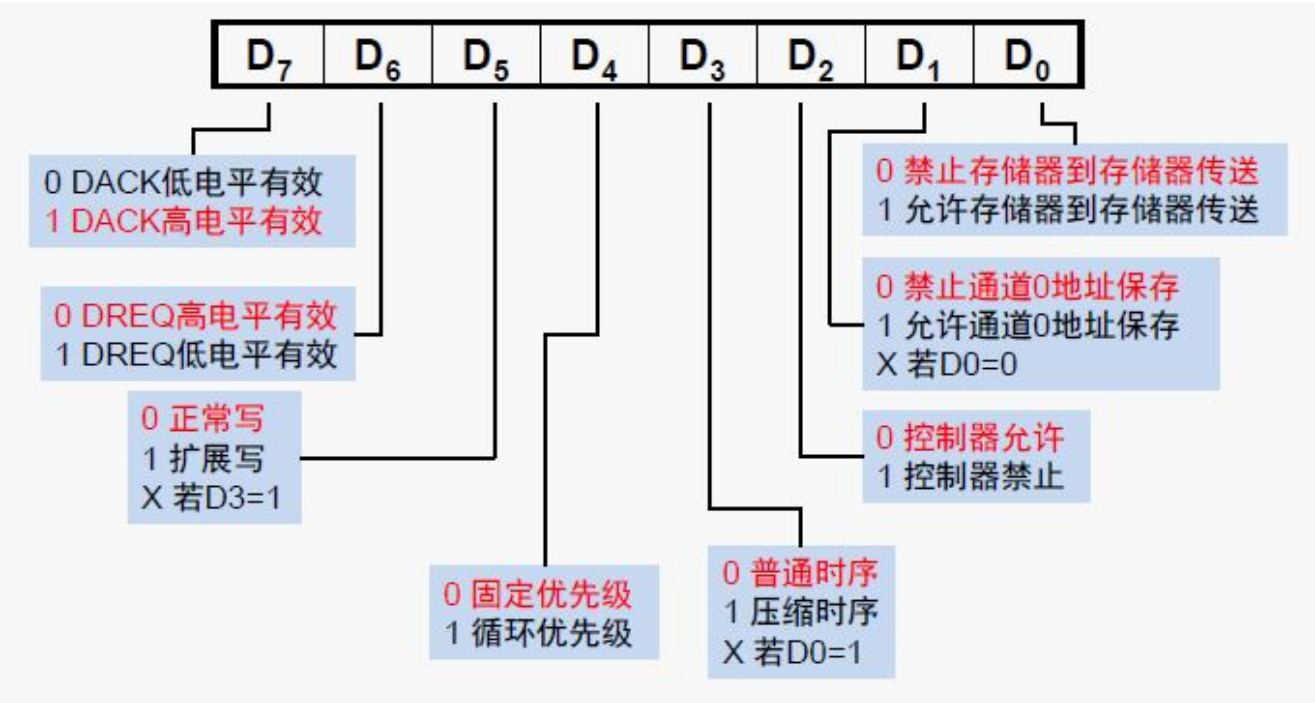
A_3	A_2	A_1	A_0	通道号	读操作(IOR)	写操作（IOW）
0	0	0	0	0	读当前地址寄存器	写基和当前地址寄存器
0	0	0	1		读当前字节计数寄存器	写基和当前字节计数寄存器
0	0	1	0	1	读当前地址寄存器	写基和当前地址寄存器
0	0	1	1		读当前字节计数寄存器	写基和当前字节计数寄存器
0	1	0	0	2	读当前地址寄存器	写基和当前地址寄存器
0	1	0	1		读当前字节计数寄存器	写基和当前字节计数寄存器
0	1	1	0	3	读当前地址寄存器	写基和当前地址寄存器
0	1	1	1		读当前字节计数寄存器	写基和当前字节计数寄存器
1	0	0	0	四个通道公用	读状态寄存器	写命令寄存器
1	0	0	1		——	写请求寄存器
1	0	1	0		——	写屏蔽寄存器某一位
1	0	1	1		——	写模式寄存器
1	1	0	0		——	清除高低位触发器命令
1	1	0	1		读暂存寄存器	主清除命令
1	1	1	0		——	——
1	1	1	1		——	写屏蔽寄存器所有位

一次OUT只能写入8位，因此在写入地址或字节时，需要OUT两次，第一次写入地址的低8位，第二次写入高8位。由于8237地址寄存器寻址空间只有16位，对于20位地址寻址空间不足，每个DMA通道增设了一个4位I/O端口，用于提供高4位地址，称为DMA页面寄存器。在写入地址时也需要写入DMA页面寄存器(0XH)。

模式寄存器(6bit)规定了通道的工作模式，每个通道一个，CPU需要对其写入“模式字”。

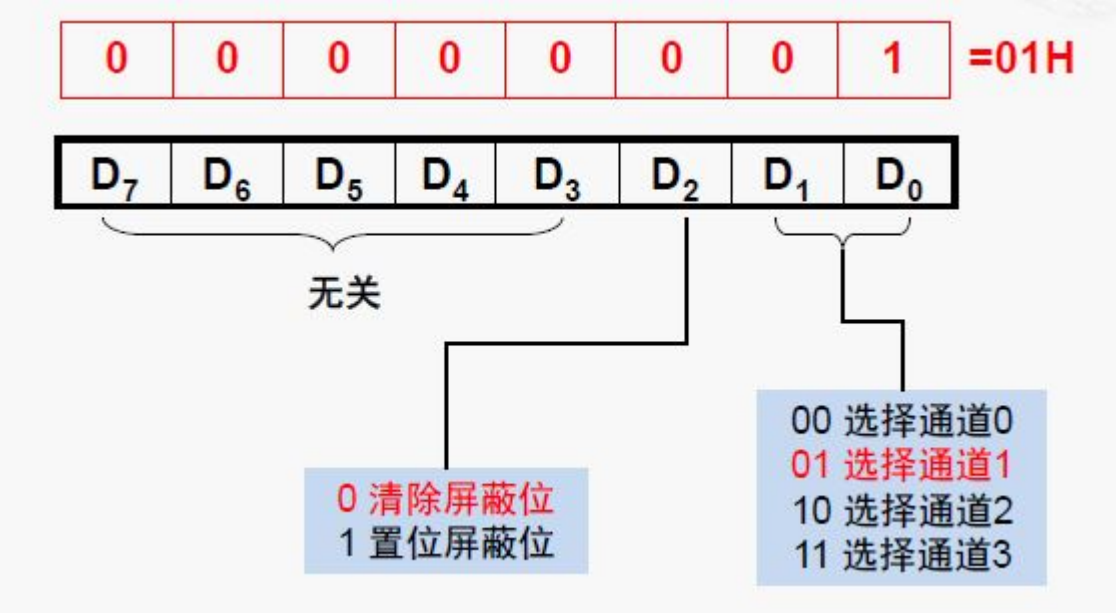


命令寄存器由四个通道公用，8bit，用以控制整个8237.CPU需要对其写入命令字

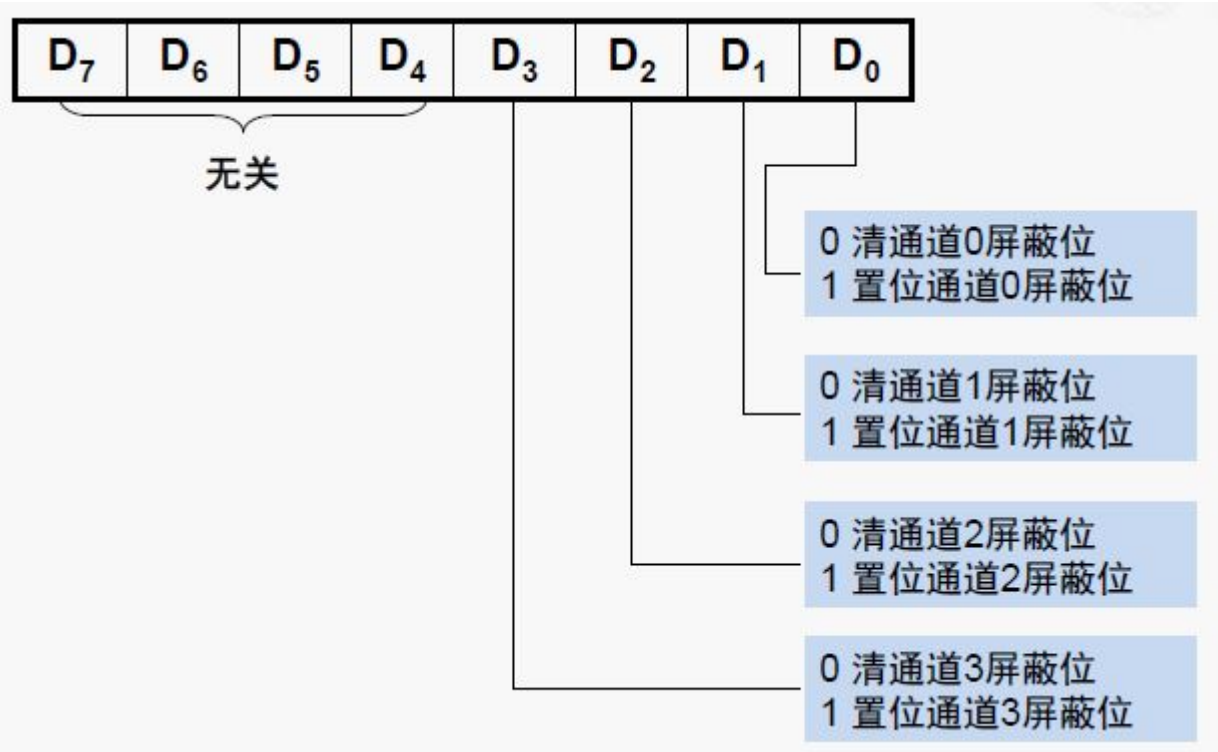


屏蔽寄存器，4bit，每个通道对应一位，若通道的屏蔽位为1，则外部DREQ信号被屏蔽，该通道DMA操作被禁止。编程时由CPU写入屏蔽字。屏蔽字有两种格式，两种格式的寻址不同。

单独设定某一个通道的屏蔽位

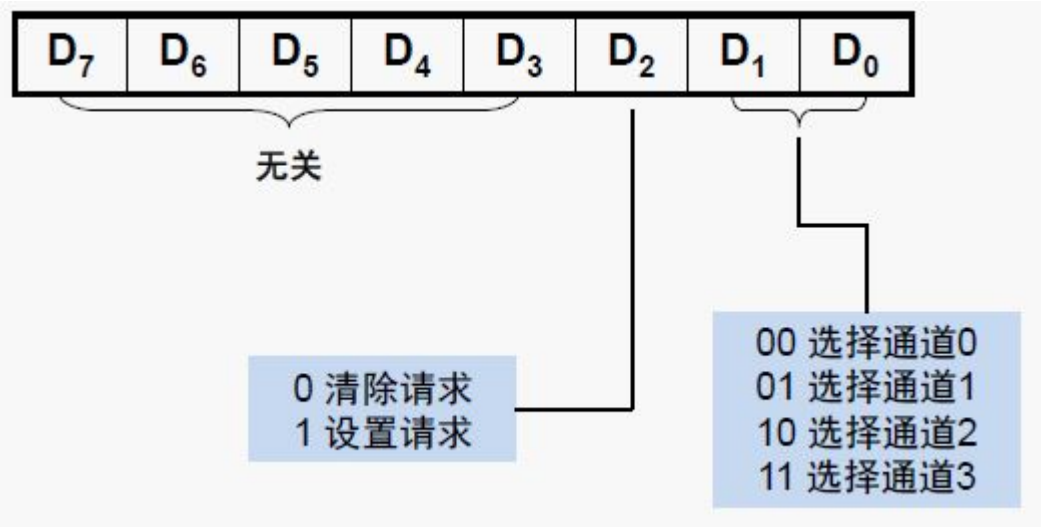


同时设定四个通道的屏蔽位

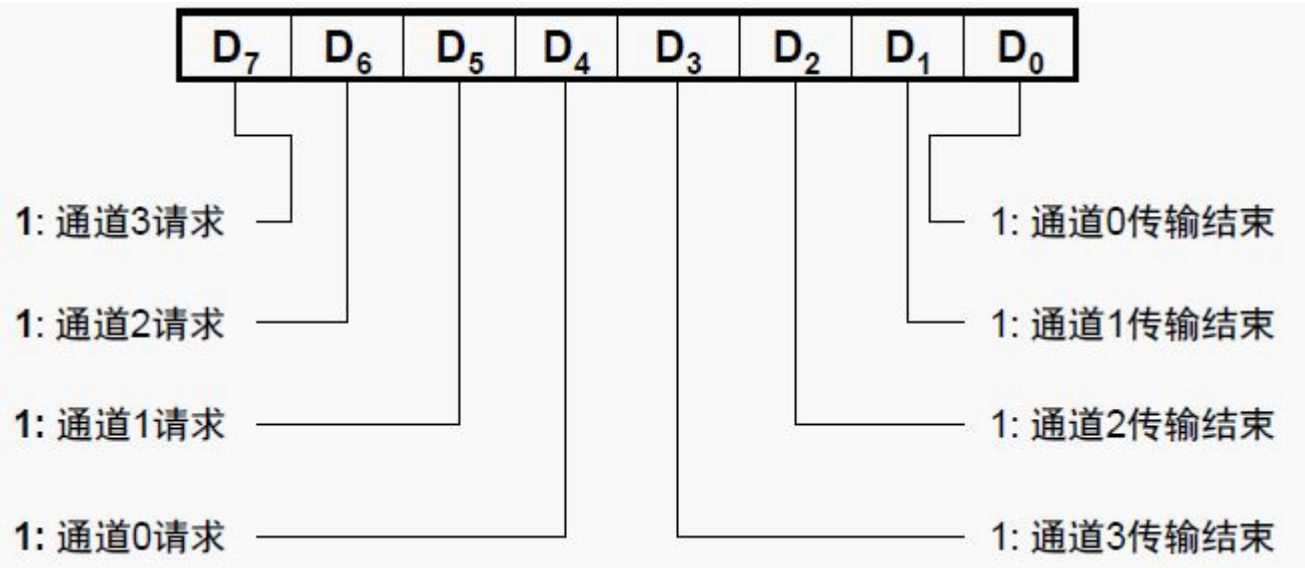


主清除命令——作用于硬件的Reset信号相同。命令、模式、状态、请求等寄存器清零；屏蔽寄存器置0FH。执行主清除命令后，8237会进入空闲周期，以便编程。

请求寄存器4bit，每位对应一个通道。当8237工作在块传送方式时也可以相应软件发出的DMA请求，软件置请求位为1，等效于外部产生了一个有效的DREQ信号。编程时，CPU写入请求字。



状态寄存器，通过读取8237状态寄存器，可以了解DMA运行的情况。



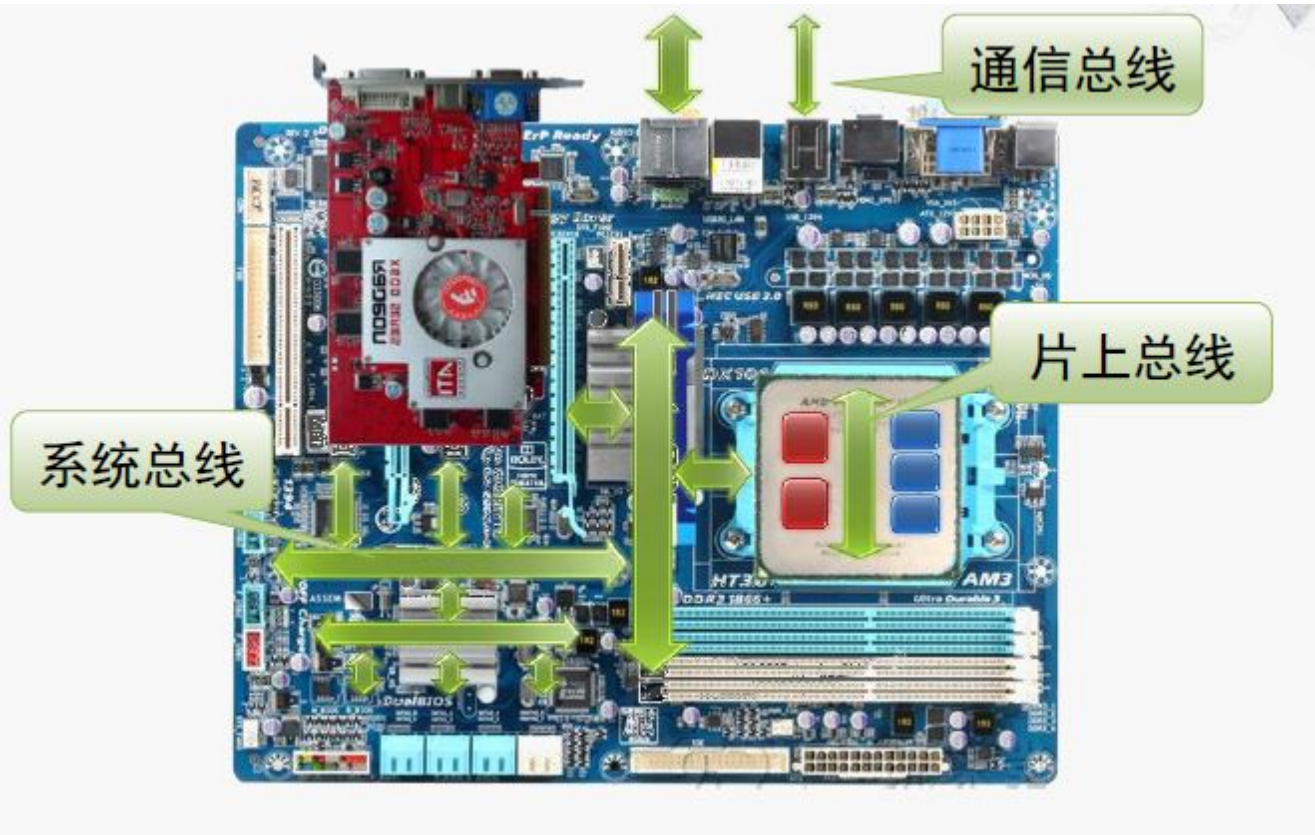
8237的编程步骤——输出主清除命令；置页面寄存器；写入基和当前地址寄存器；写入基和当前字节计数寄存器；写入模式寄存器；写入命令寄存器；写入屏蔽寄存器；写入请求寄存器（硬件DMA请求没有这一步）。

初始化完成后，可以通过读状态寄存器，了解各个通道的传输情况。

7. 总线

总线——在多于两个模块（设备或者子系统）之间传送信息的公共通路。

总线由传输信息的电路和管理信息传输的协议两部分组成。



片总线（器件级总线，片上总线）——CPU内部的总线

内总线（系统总线，板级总线）——个插件板之间信息传输的通路

外总线（通信总线）——计算机系统之间/计算机与其他系统之间信息传输的通路

总线模块

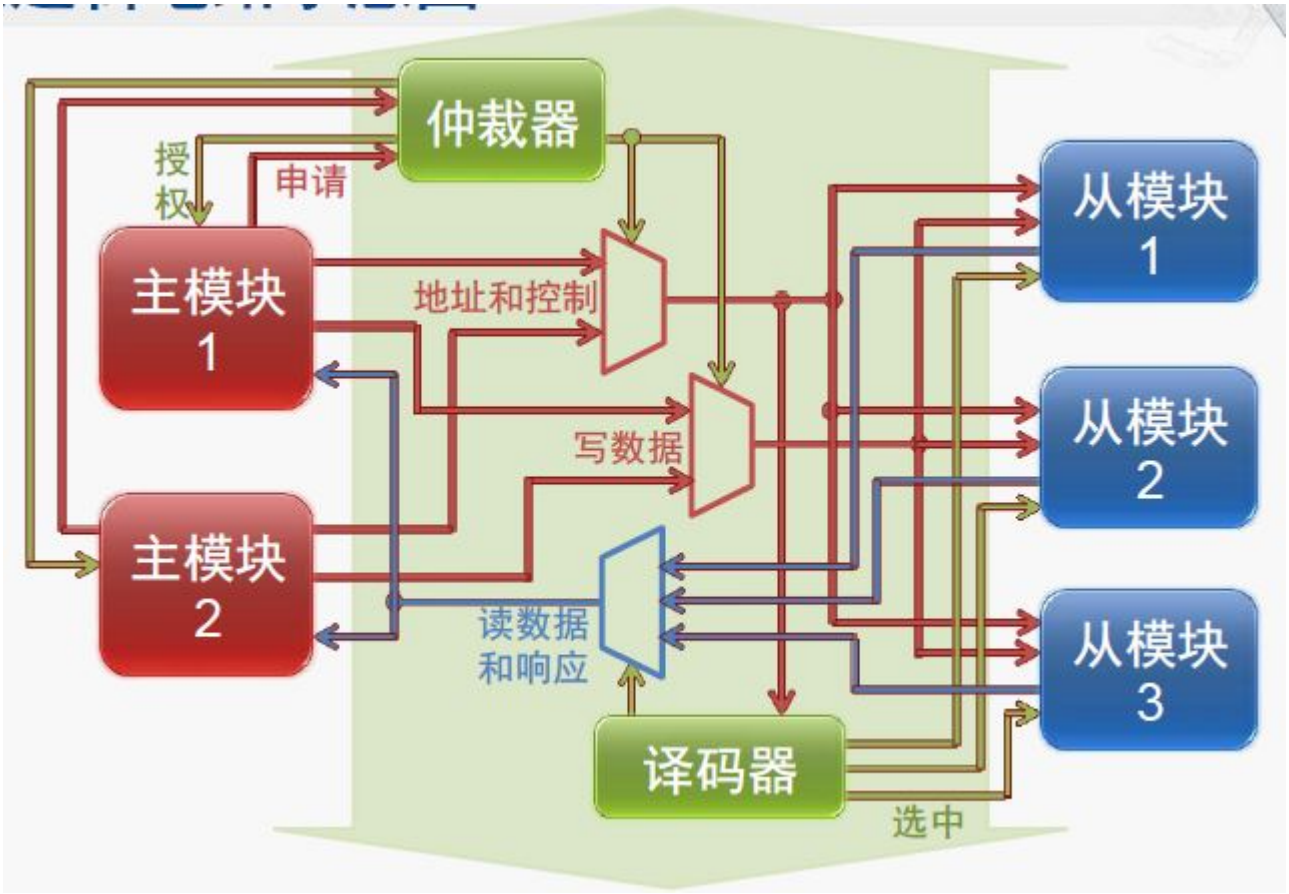
总线主模块（Master）——具有控制能力，在获得总线控制权后能启动总线传输，eg. CPU, DMAC

总线从模块（Slave）——能够对总线传输做出相应，但本身不具备总线控制能力，eg. Mem

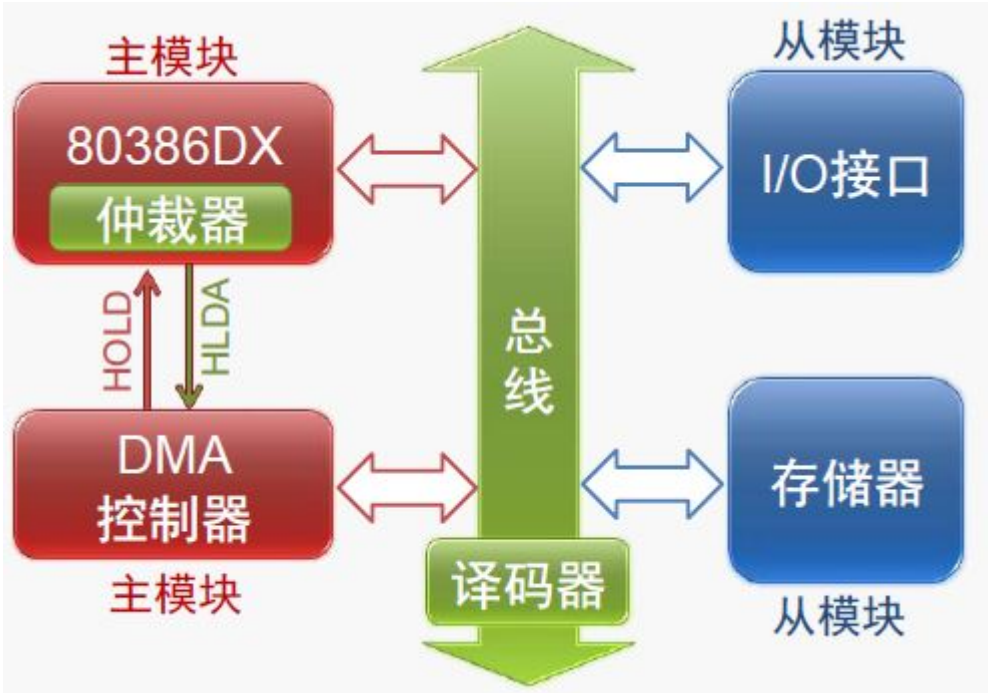
总线译码器（Decoder）——根据当前控制总线的主模块提供的地址，选择作为本次总线传输目标的从模块

总线仲裁器（Arbiter）——当有多个主模块同时请求使用总线时，决定由哪个主模块获得总线的控制权，目的是让总线能被更合理、高效地使用。

总线逻辑示意



80386DX总线模型



总线标准——事实标准，国际标准

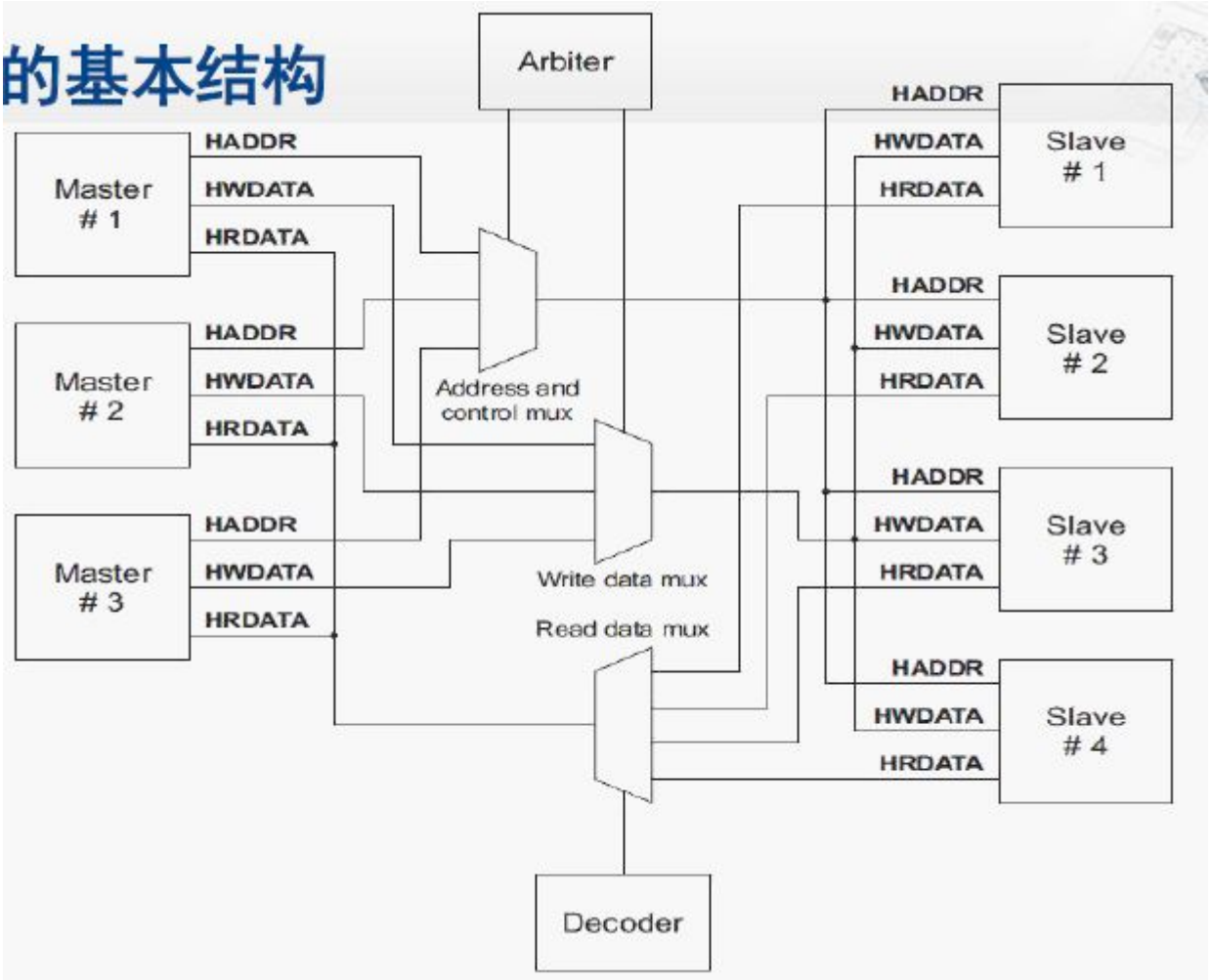
事实标准：计算机系统厂家才用的总线，由于性能优越而被业界普遍支持和承认

国际标准：在国际标准组织/机构主持下开发和制定的总线标准，公布后被厂家和用户使用。

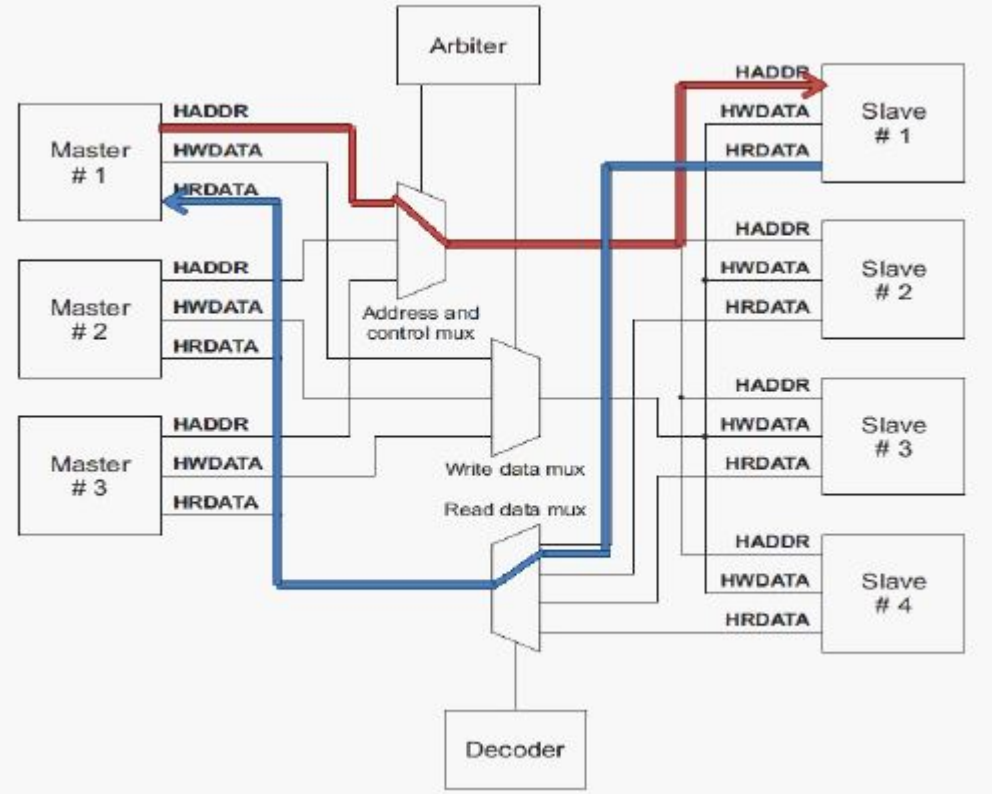
机械标准（规定模块插件的机械尺寸等等）+电气特性（规定总线信号的逻辑电平等）+功能特性（规定总线信号的名称及功能定义）+规格特性（对各总线信号的动作过程及时序关系进行说明）

AMBA总线标准——Advanced Microcontroller Bus Architecture

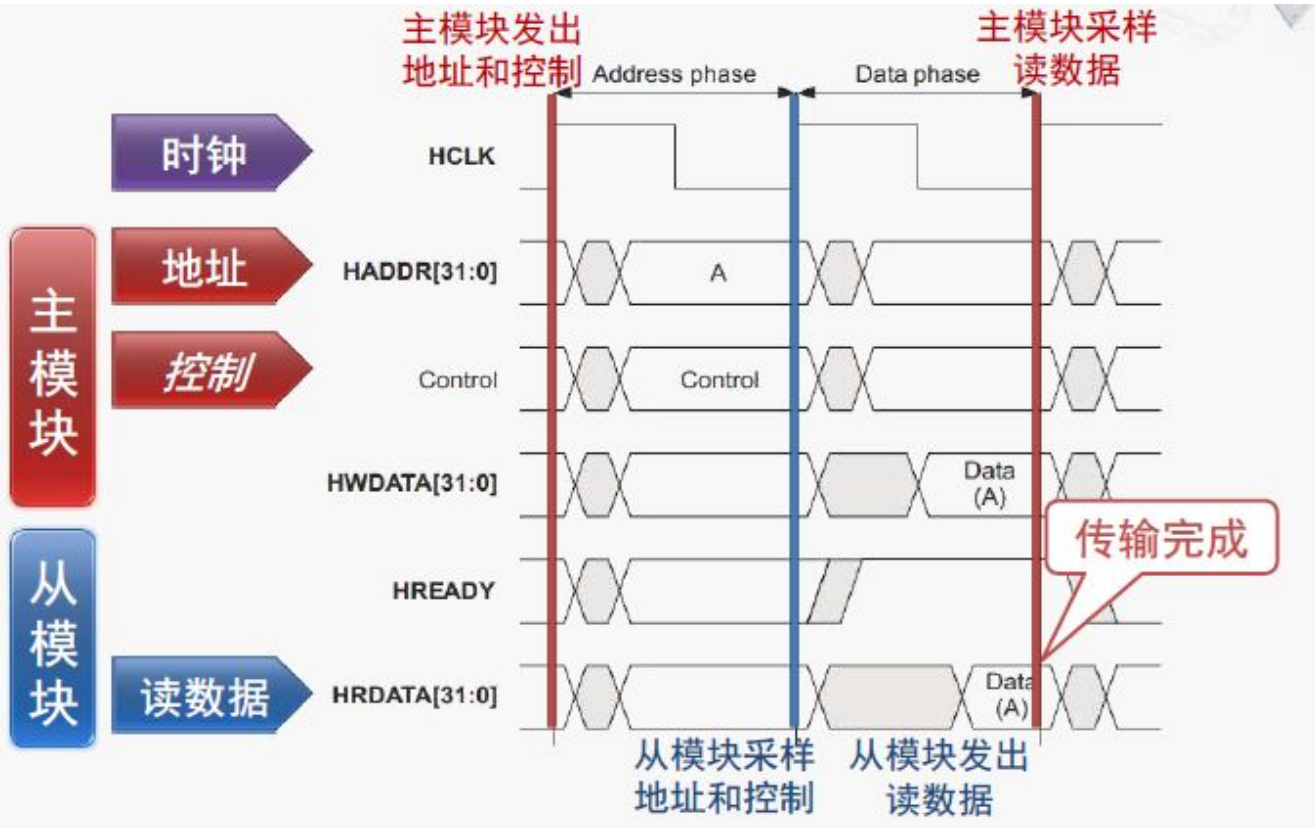
AHB基本结构



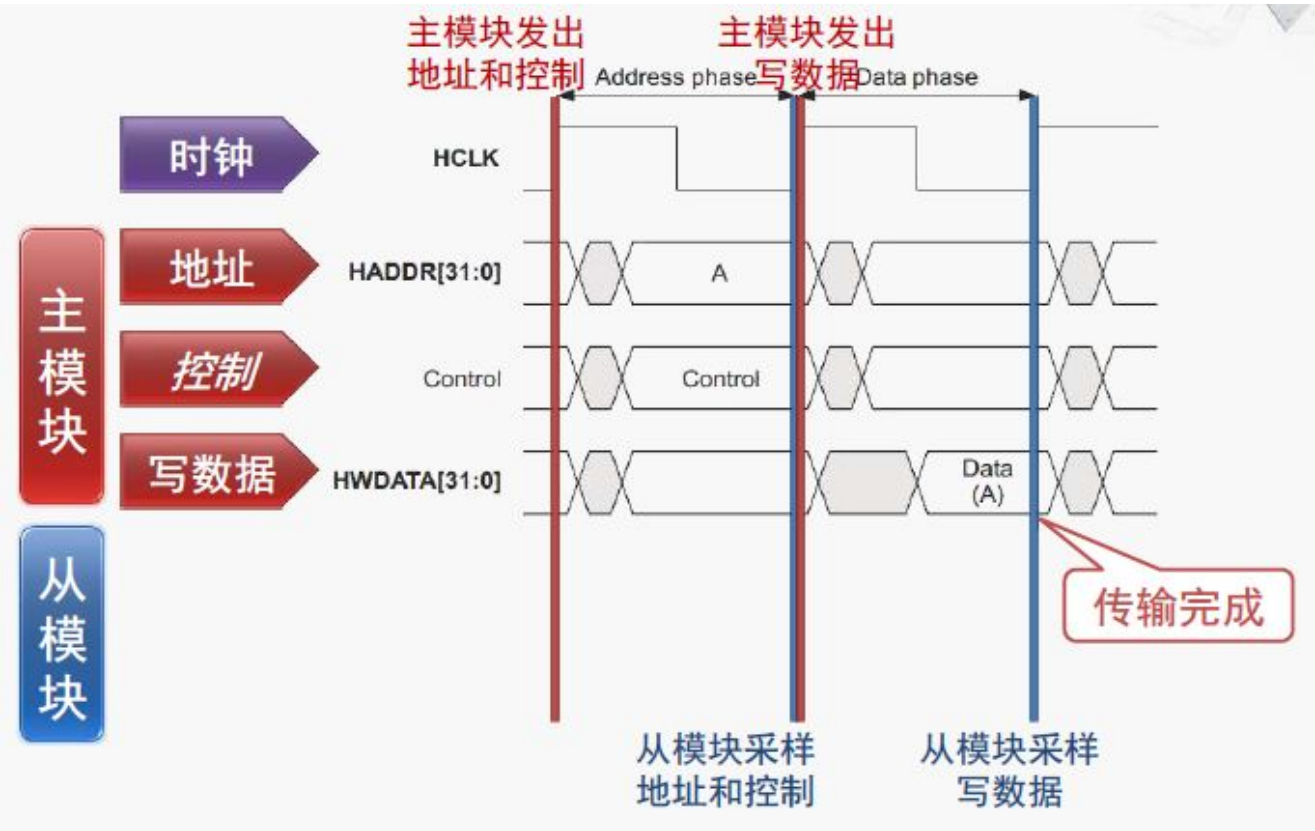
读/写一个数据的传输——主模块读/写一个数据于从模块



读一个数据——第一个时钟上升沿之后，主模块驱动HADDR和Control信号；第二个时钟上升沿，从模块采样HADDR和Control信号；第二个时钟上升沿之后，从模块驱动HRDATA信号；第三个时钟上升沿，主模块采样HRDATA信号，完成读一个数据的传输。



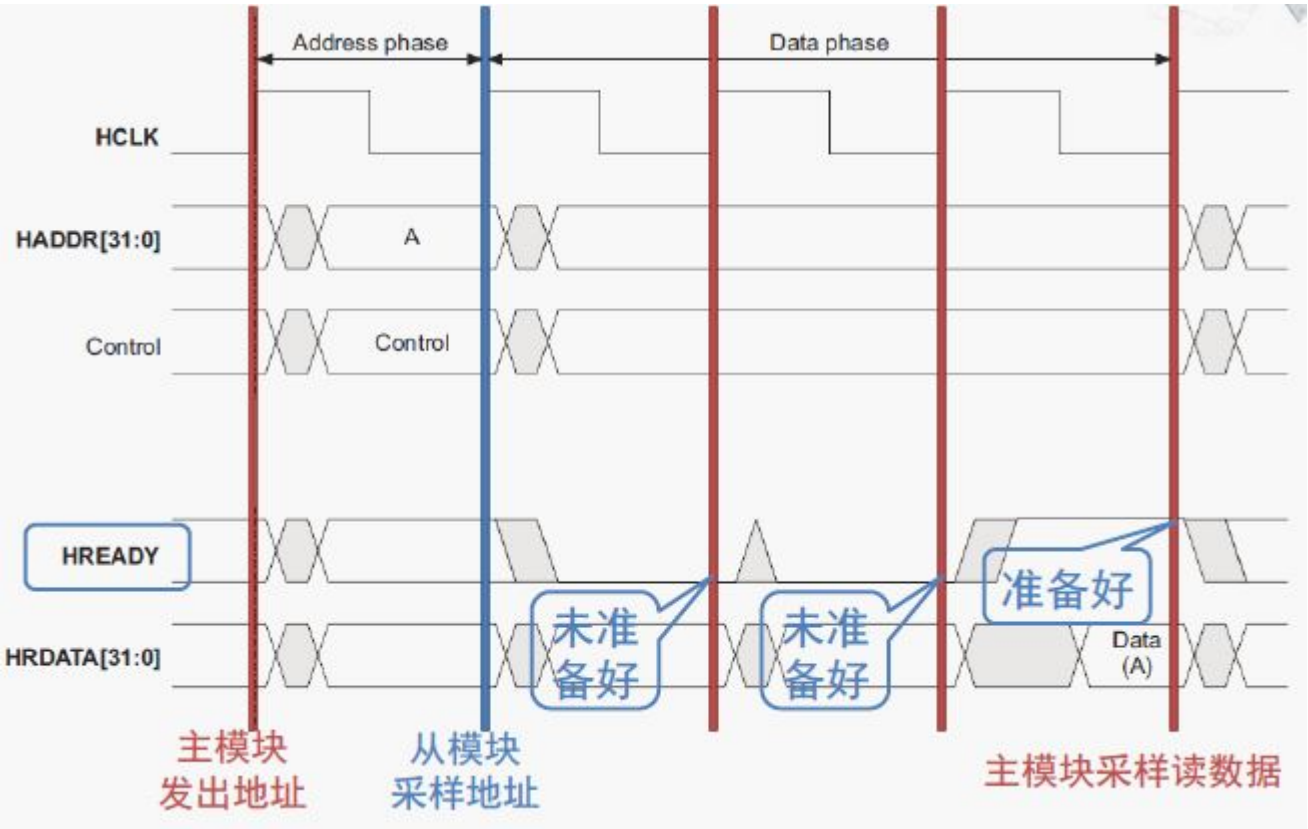
写一个数据——第一个时钟上升沿之后，主模块驱动HADDR和Control信号；第二个始终上升沿，从模块采样HADDR和Control信号；第二个时钟上升沿之后，主模块驱动HWDATA信号；第三个时钟上升沿，从模块采样HWDATA信号，完成写一个数据的传输。



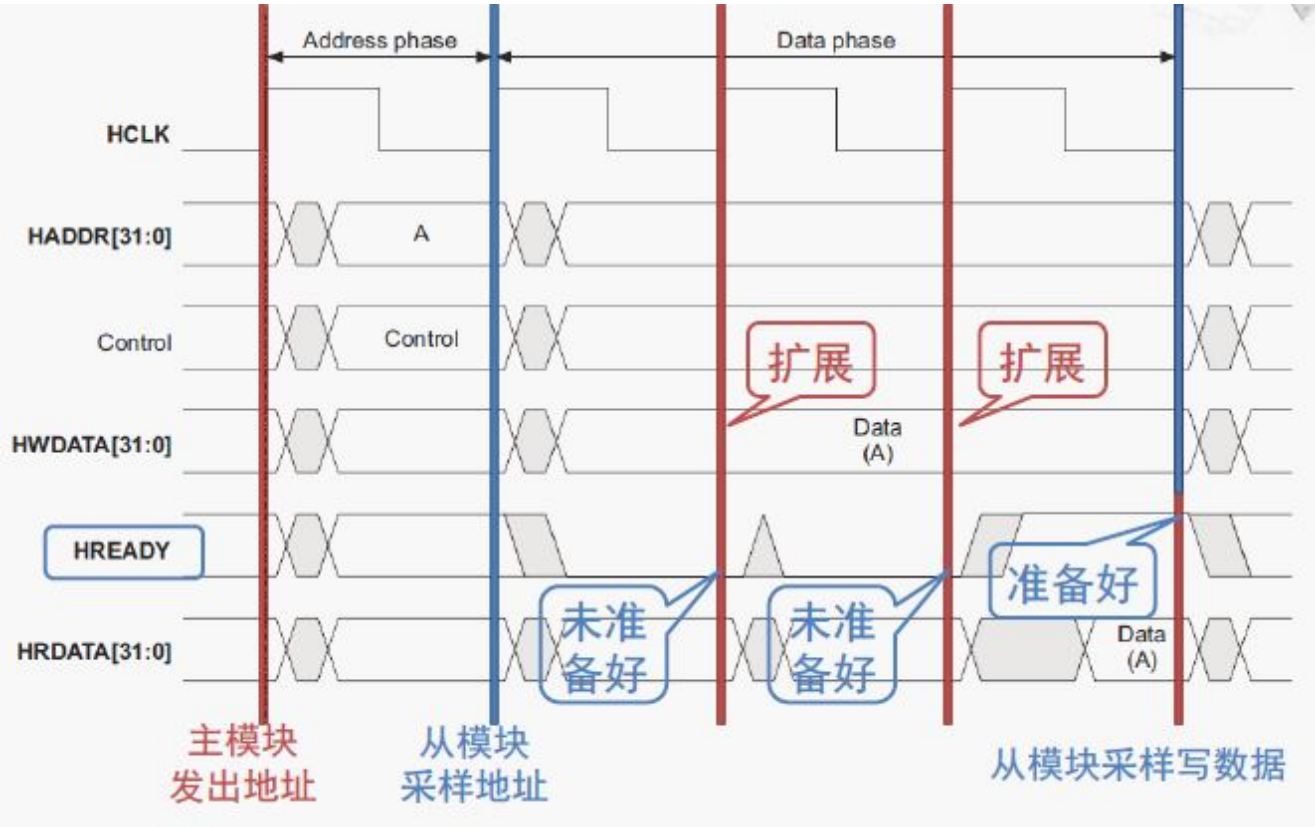
主模块发起读/写一个数据的传输，从模块未准备好。

这一情况非常常见，比如CPU为总线主模块，Mem控制器为总线从模块；一般需要等待数十甚至数百个时钟周期才能返回数据。

从模块无法提供读数据——从模块在传输过程中插入等待周期，以便获得额外的准备时间；在等待周期中，主模块保持HADDR和Control信号稳定。

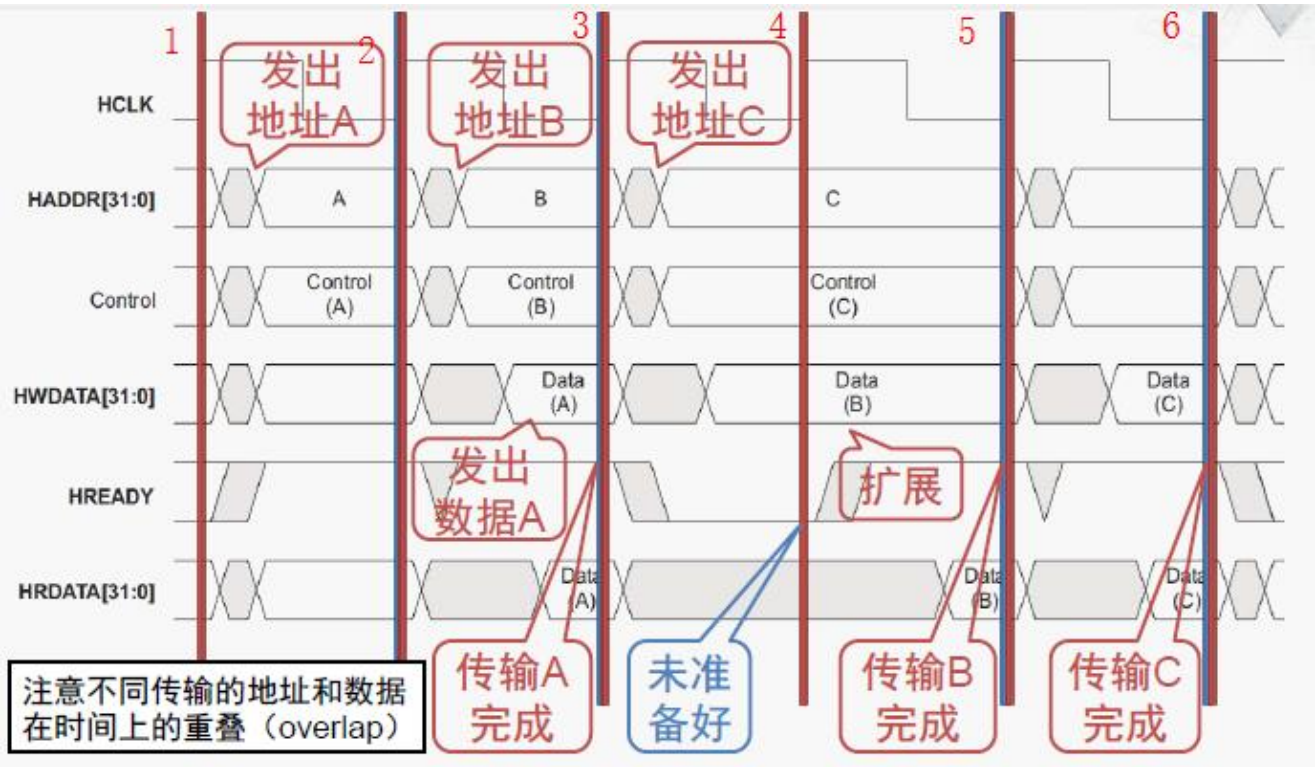


从模块无法提供读数据——同上；等待周期中，主模块保持HADDR，Control和HWDATA信号稳定。



多个连续的传输——一个主模块连续发起多个传输；多个不同的主模块先后发起传输。

一个主模块连续发起多个传输——① 主模块发出地址A；② 从模块采样地址A，主模块发出地址B；③ 从模块接收数据A，传输A完成，从模块采样地址B，主模块发出地址C；④ 从模块未准备好接收数据B，插入等待周期，主模块保持地址C、控制C以及数据B；⑤ 从模块接收数据B，传输B完成，从模块采样地址C；⑥ 从模块采样数据C，传输C完成



时间重叠：不同的传输地址和数据在时间上存在重叠——充分利用地址总线和数据总线

等待周期：地址B的传输中有一个等待周期，在数据阶段扩展了一个周期——副作用是地址C的传输在地址阶段受到影响，也扩展了一个周期

大量连续数据的传输——一次传输的数据量超过了数据总线的宽度，比如CPU需要读出128位的数据但数据总线宽度只有32位。

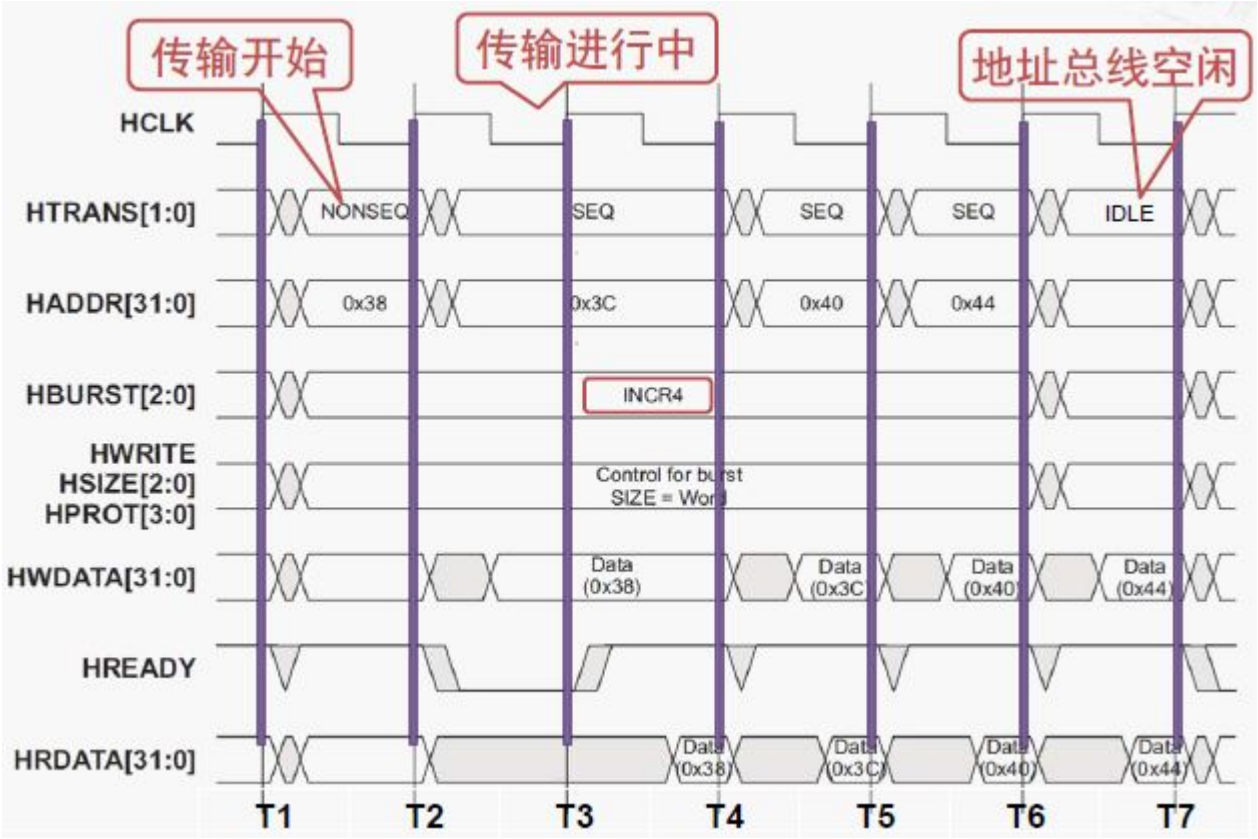
BURST传输——主模块控制信号HTRANS[1:0]和HBURST[2:0]进行控制；HTRANS[1:0]定义传输中各周期的类型，HBURST[2:0]定义传输长度和地址变化方式

编码	类型	说明
00	IDLE	Indicates that no data transfer is required. The IDLE transfer type is used when a bus master is granted the bus, but does not wish to perform a data transfer.
01	BUSY	Indicates that the bus master is continuing with a burst of transfers, but the next transfer cannot take place immediately. The transfer should be ignored by the slave.
10	NONSEQ	Indicates the first transfer of a burst or a single transfer. The address and control signals are unrelated to the previous transfer. (NONSEQUENTIAL)
11	SEQ	The remaining transfers in a burst are SEQUENTIAL and the address is related to the previous transfer. The control information is identical to the previous transfer.

*编码为二进制

编码	类型	说明
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

*编码为二进制



T1：传输开始，主模块发出第一个地址，驱动HTRANS=NONSEQ，HBURST=INCR4

T2：传输继续，主模块发出第二个地址，驱动HTRANS=SEQ，保持HBURST=INCR4

T3：从模块未准备好，插入等待周期，主模块保持信号稳定

…

T6：主模块发出的第四个地址已经被从模块采样，无需再发出新的地址和控制信号，因此驱动HTRANS=IDLE

T7：传输完成

BURST传输可以应用于Cache的行填充(读)或行替换(写)，也可用于外设和主存、主存中不同区域之间的大量的连续数据传输。

希望在BURST传输中先得到某个特定的数据——地址回卷BURST传输

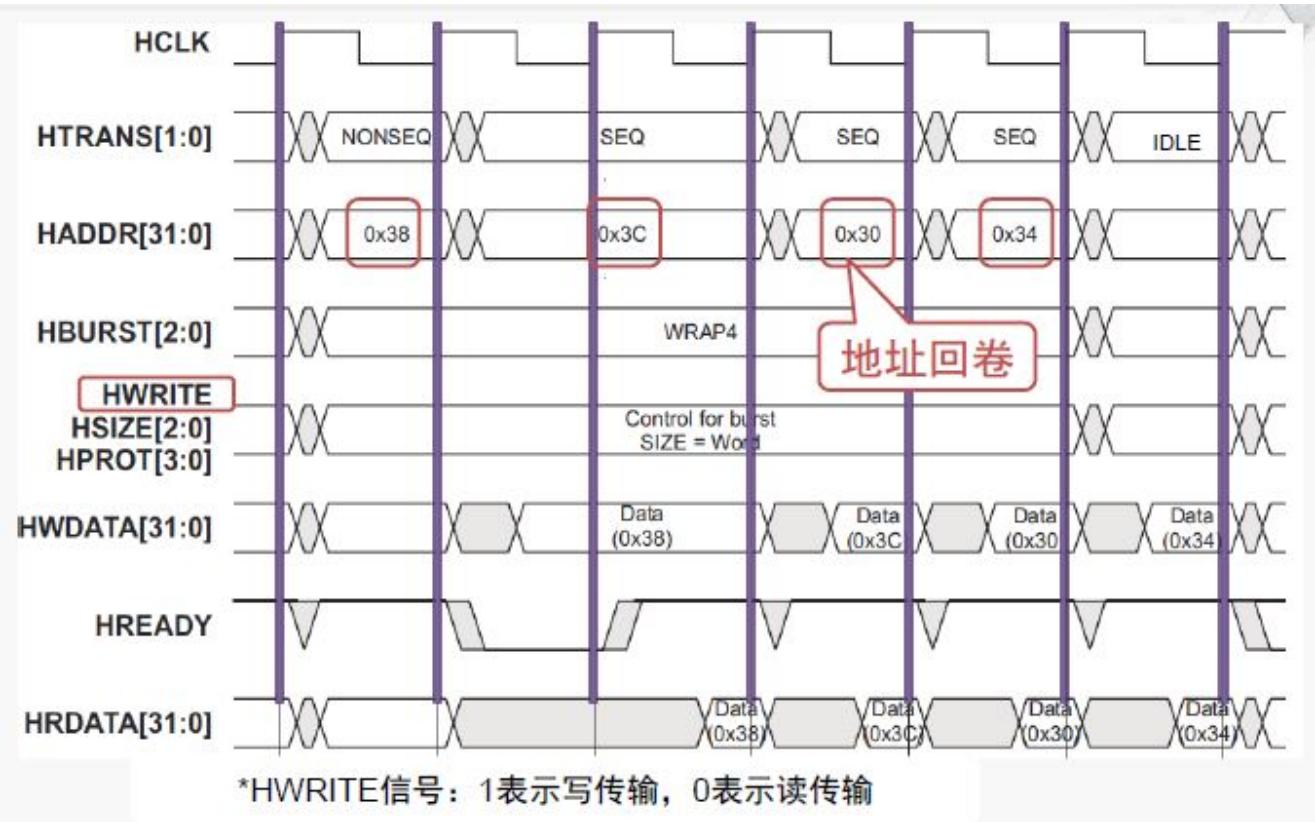
比如Cache在行填充时，尽管需要BURST传输，但CPU还是希望能先得到所需的那一项，而不是等待BURST传输，因为这是影响Cache性能的重要的因素。

地址回卷方式WRAPx——由HBURST[2:0]定义。WRAP4在16的整数倍的地址处回卷（4x4）；WRAP8在32的整数倍的地址处回卷（4x8）；WRAP16在64的整数倍的地址处回卷（4x16）。如下使用WRAP4地址回卷，2与1的起始地址相同，2与3的访问区域相同。使用这一方式即可达到有限得到某个数据的目的。

INCR4: 0x38, 0x3c, 0x40, 0x44 (1)

WRAP4: 0x38, 0x3c, 0x30, 0x34 (2)

INCR4: 0x30, 0x34, 0x38, 0x3c (3)



*HWRITE信号：1表示写传输，0表示读传输