第7章　串行通信接口

第1节　串行通信及接口基础

一、有关概念

二、典型异步串行通信接口简介

三、PC机的串行通信接口

第2节　MSP430通用串行通信模块

一、MSP430 UART 模块的两种工作方 式

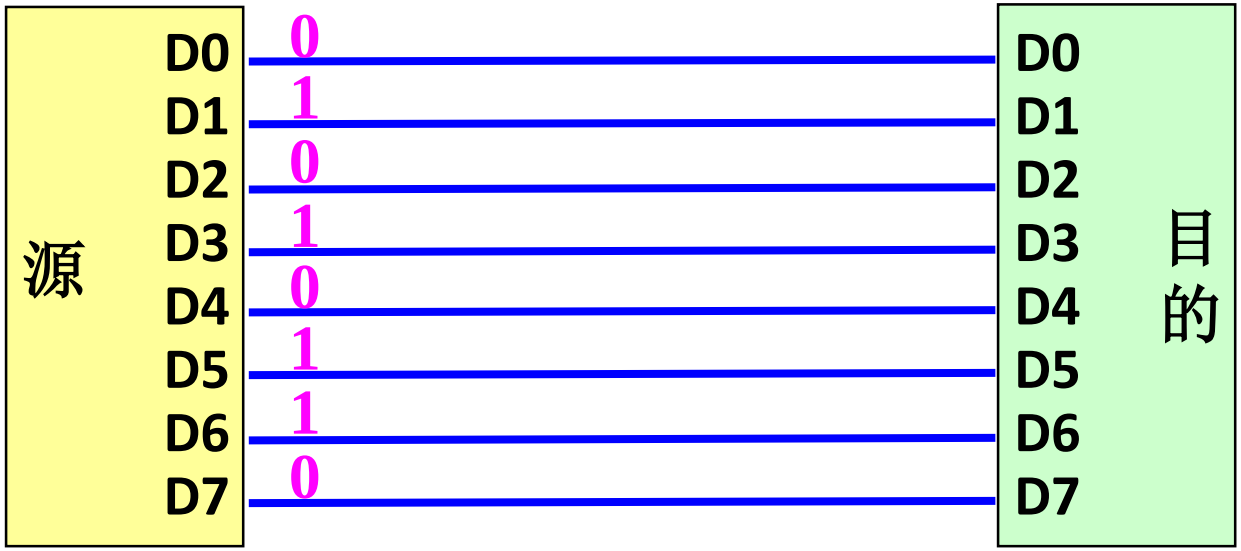
二、MSP430 异步串行方式编程结构

三、利用MSP430 异步串行方式通信

1. 并行通信和串行通信

■ 并行通信

将数据的各位同时在多根并行传输线上进行传输。



数据的各位同时由源到达目的地 → 快

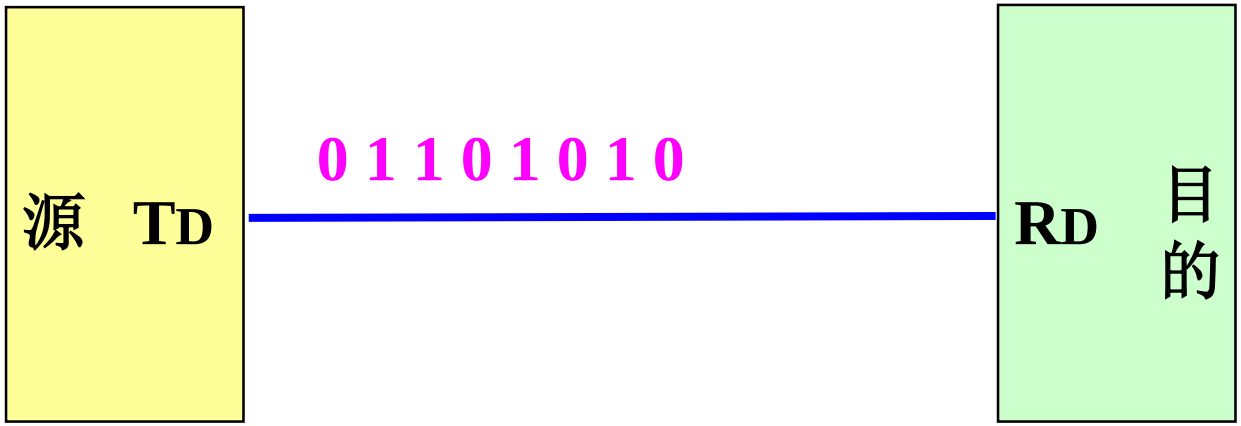
多根数据线 → 远程费用高

并行通信适于短距离、高速通信

例如： **PC机的标准并行接口LPT1 (打印机接口)、PCI总线、MSP430的P1~P6均可做8位并行通信**

■ 串行通信

将数据的各位按时间顺序依次在同一传输线上传输。

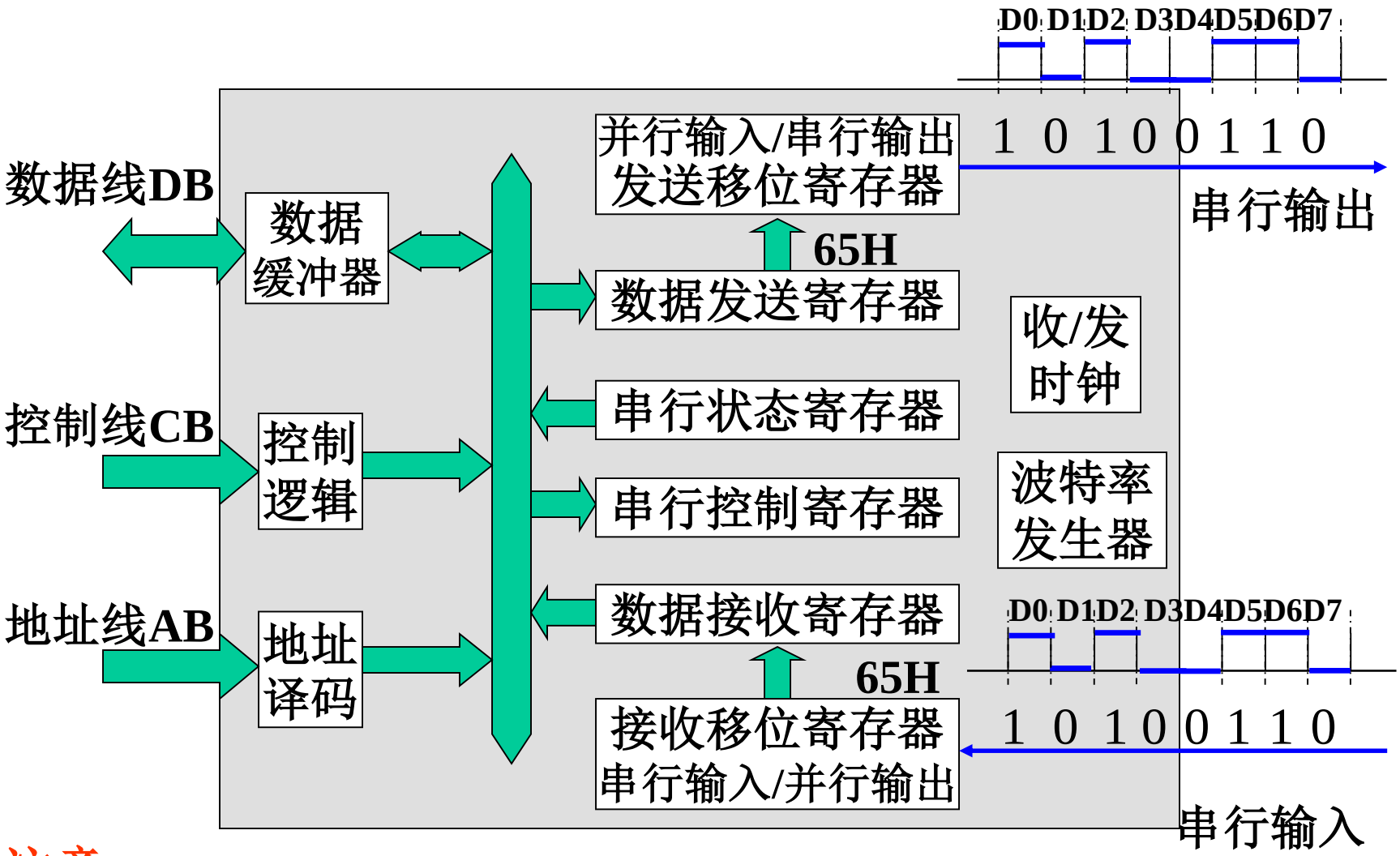


数据的各位依次由源到达目的地

数据线少 → 远程, 费用低

串行通信适于中、长距离通信

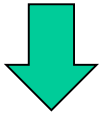
例如： **PC机的标准串行通信接口COM1, COM2**
USB、以太网



注意:

“串行”是外设与接口电路之间的数据传送方式，在CPU与接口之间仍是并行工作方式。

数据发送过程



65H

	D7	D6	D5	D4	D3	D2	D1	D0
数据发送寄存器	0	1	1	0	0	1	0	1

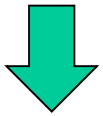
	D7	D6	D5	D4	D3	D2	D1	D0
发送移位寄存器								

空闲位

数据发送过程

数据发送寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1



65H

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

空闲位

数据发送过程

数据发送寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

空闲位 B

起始位

数据发送过程

数据发送寄存器

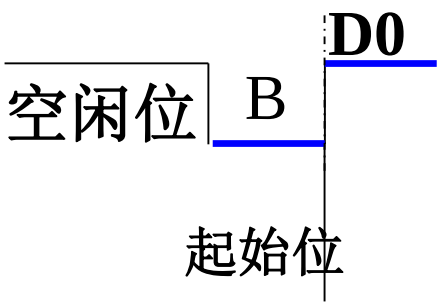
D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
	0	1	1	0	0	1	0

1

>>



数据发送过程

数据发送寄存器

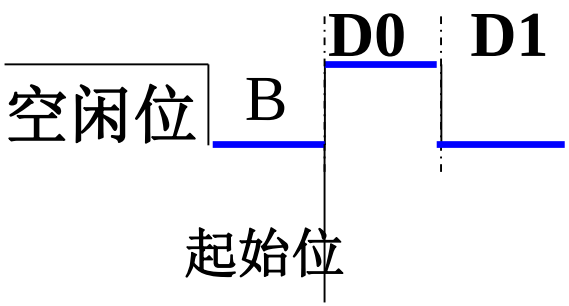
D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
		0	1	1	0	0	1

0

>>



数据发送过程

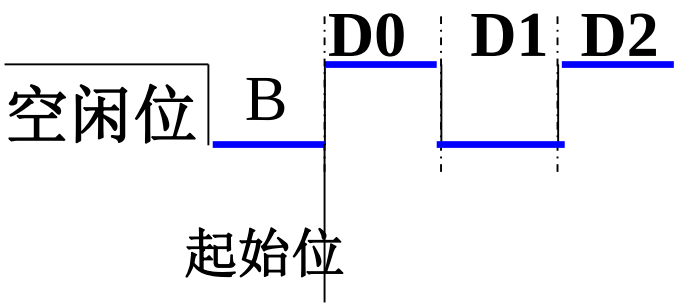
数据发送寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0	
			0	1	1	0	0	1

>>



数据发送过程

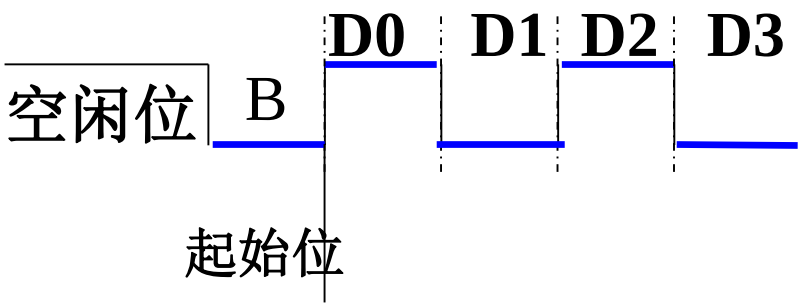
数据发送寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0	
				0	1	1	0	0

>>



数据发送过程

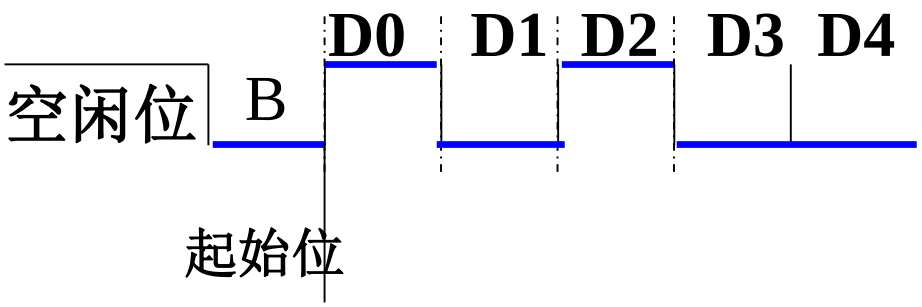
数据发送寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0	
					0	1	1	0

>>



数据发送过程

数据发送寄存器

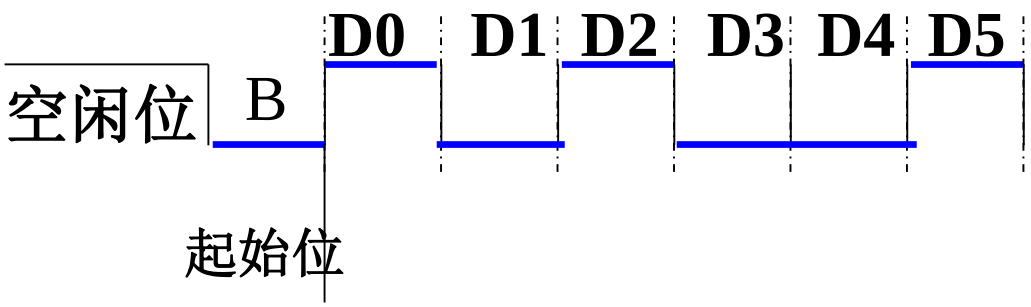
D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
						0	1

1

>>



数据发送过程

数据发送寄存器

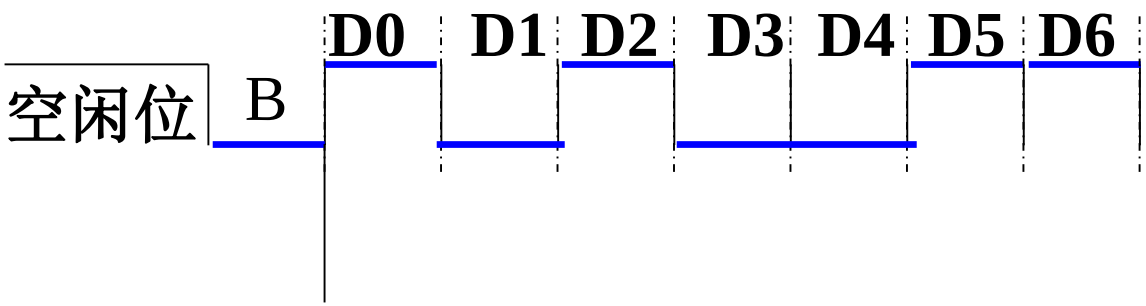
D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
							0

1

>>



数据发送过程

数据发送寄存器

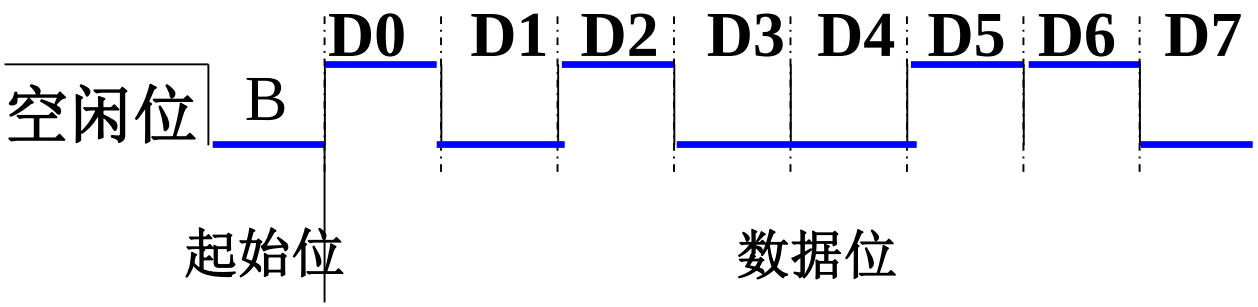
D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

发送移位寄存器

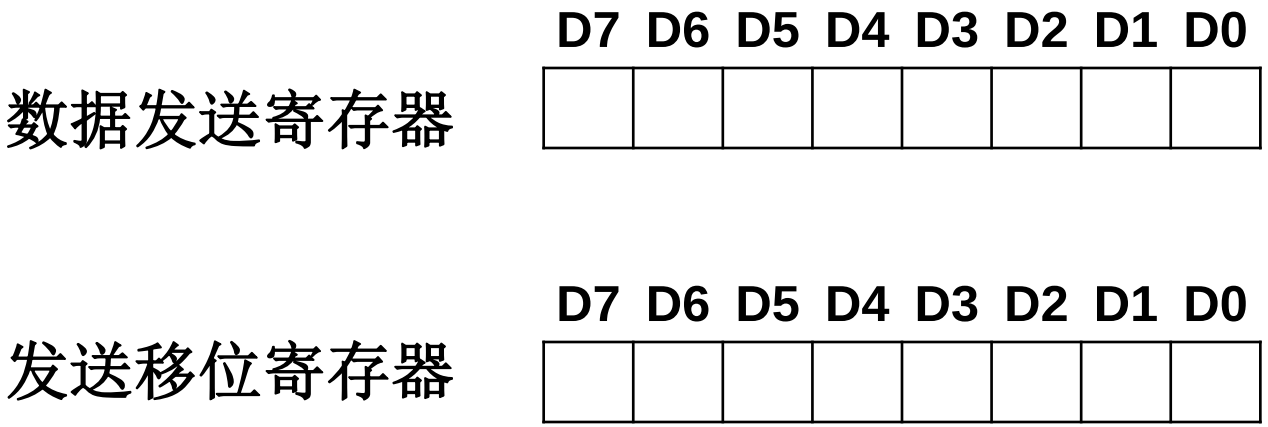
D7	D6	D5	D4	D3	D2	D1	D0

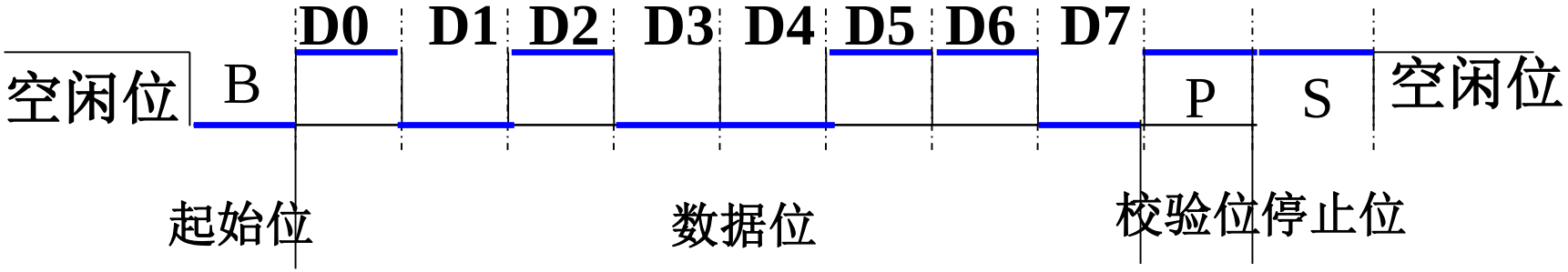
0

>>



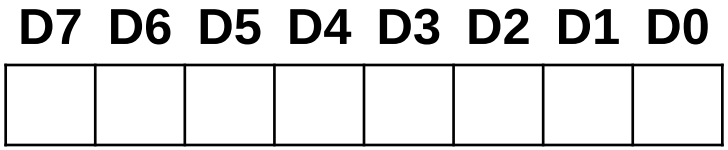
数据发送过程


D7 D6 D5 D4 D3 D2 D1 D0
数据发送寄存器
D7 D6 D5 D4 D3 D2 D1 D0
发送移位寄存器

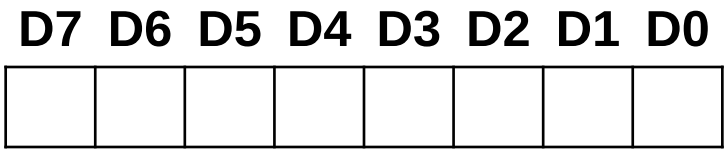

空闲位
B
D0
D1
D2
D3
D4
D5
D6
D7
P
S
空闲位
起始位
数据位
校验位停止位

数据接收过程

数据接收寄存器



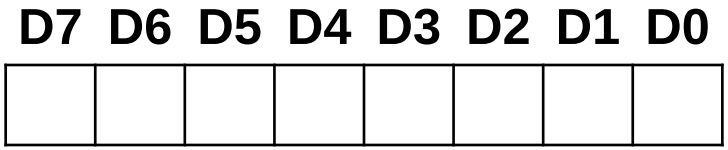
接收移位寄存器



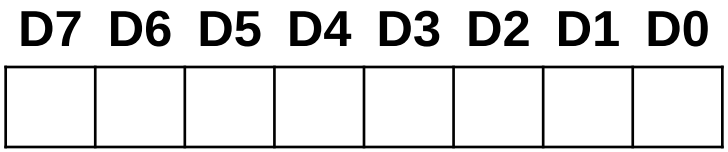
空闲位

数据接收过程

数据接收寄存器



接收移位寄存器



空闲位 B

起始位

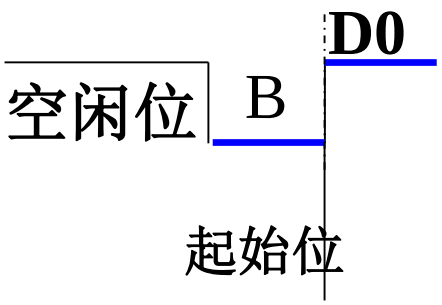
数据接收过程

数据接收寄存器

D7	D6	D5	D4	D3	D2	D1	D0

接收移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
1							



数据接收过程

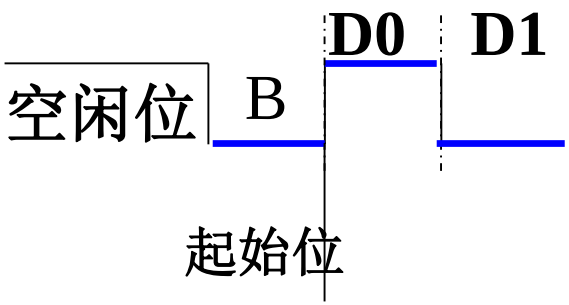
数据接收寄存器

D7	D6	D5	D4	D3	D2	D1	D0

接收移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1						

>>



数据接收过程

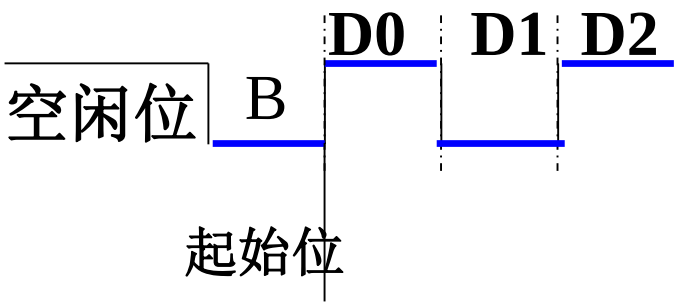
数据接收寄存器

D7	D6	D5	D4	D3	D2	D1	D0

接收移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1					

>>



数据接收过程

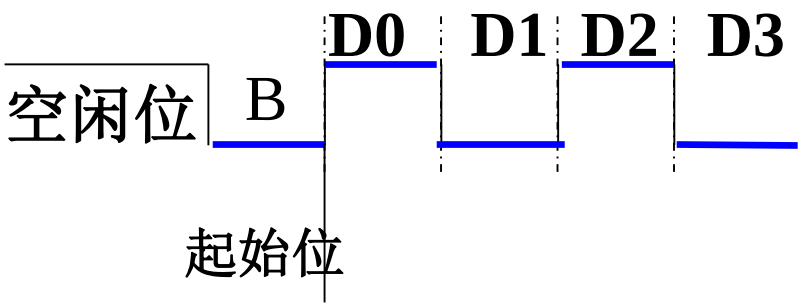
数据接收寄存器

D7	D6	D5	D4	D3	D2	D1	D0

接收移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	1				

>>



数据接收过程

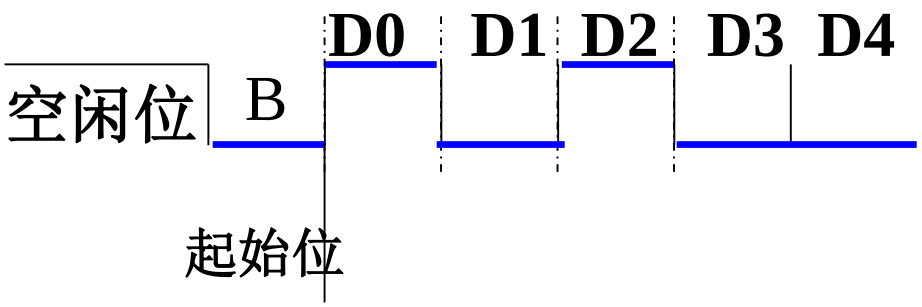
数据接收寄存器

D7	D6	D5	D4	D3	D2	D1	D0

接收移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	1			

>>



数据接收过程

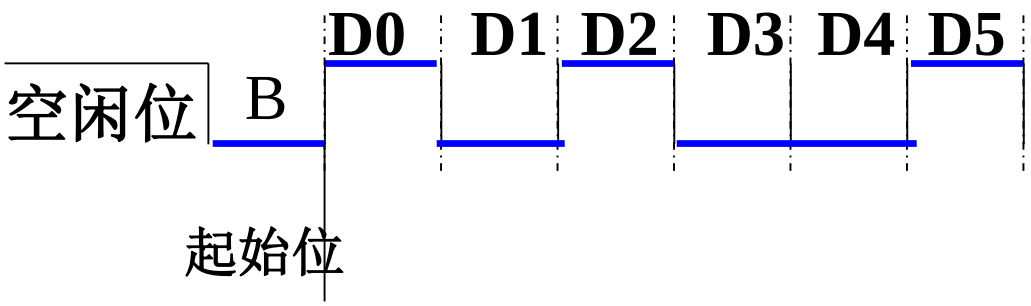
数据接收寄存器

D7	D6	D5	D4	D3	D2	D1	D0

接收移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	1	0	1		

>>



数据接收过程

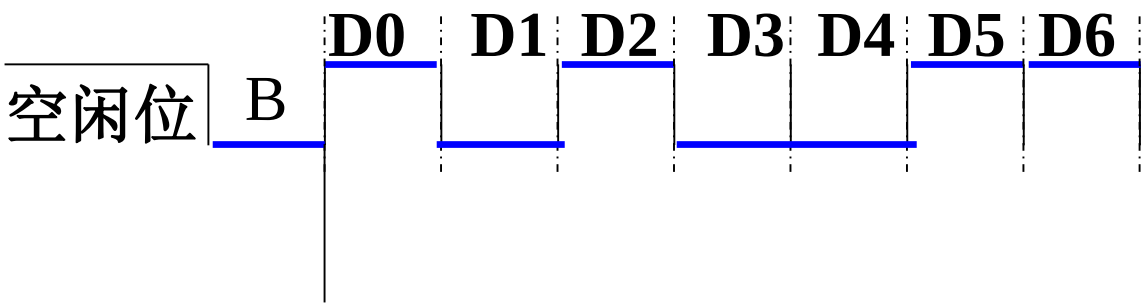
数据接收寄存器

D7	D6	D5	D4	D3	D2	D1	D0

接收移位寄存器

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	1	0	1	

>>



数据接收过程

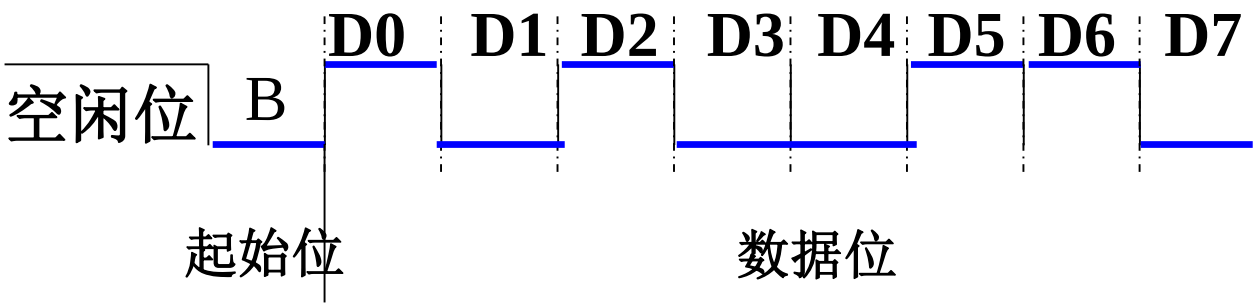
数据接收寄存器

D7	D6	D5	D4	D3	D2	D1	D0

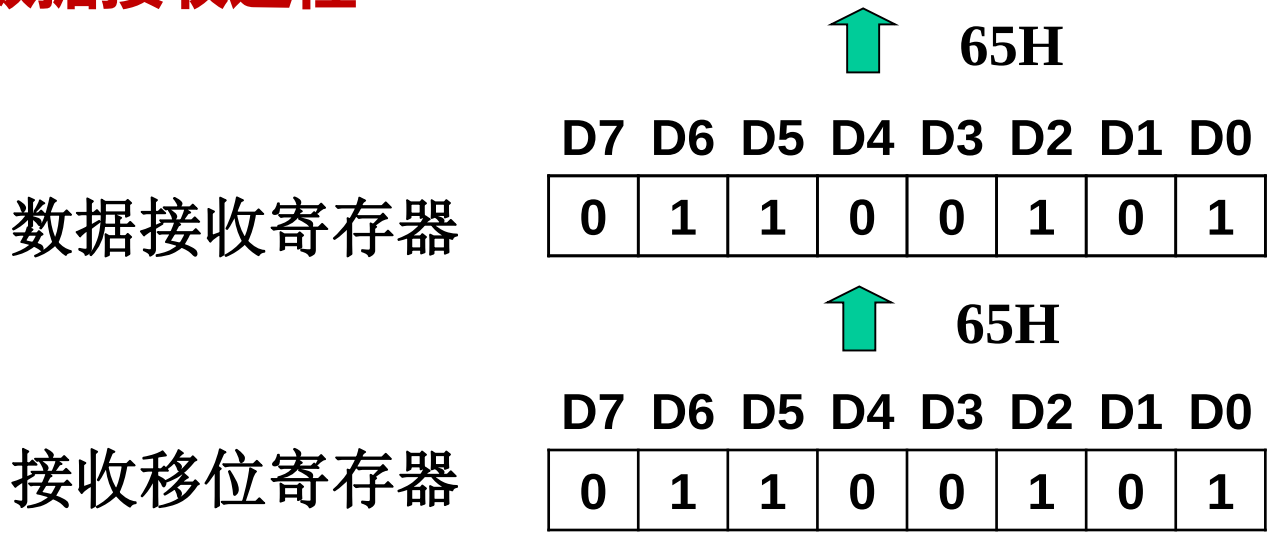
接收移位寄存器

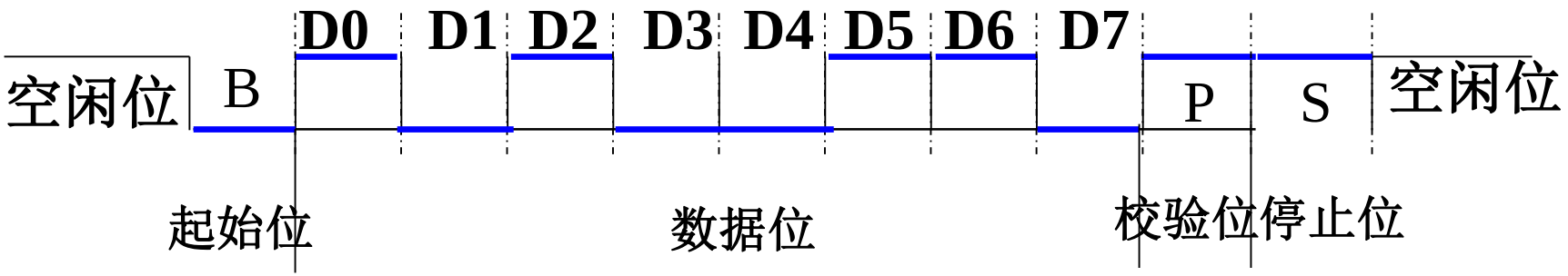
D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	0	0	1	0	1

>>



数据接收过程


65H
D7 D6 D5 D4 D3 D2 D1 D0
数据接收寄存器
0 1 1 0 0 1 0 1
65H
D7 D6 D5 D4 D3 D2 D1 D0
接收移位寄存器
0 1 1 0 0 1 0 1


空闲位
B
D0
D1
D2
D3
D4
D5
D6
D7
P
S
空闲位
起始位
数据位
校验位停止位

2. 信息传输的检错和纠错

- 串行数据在传输过程中，由于干扰可能引起信息的出错，

  如何发现传输中的错误，叫检错；

  发现错误后，如何消除错误，叫纠错

- 最简单的检错方法是奇偶校验，

  即在传送字符的各位之外，再传送1位奇/偶校验位。

  可采用奇校验或偶校验：

  奇校验：使所有传送的数位(含校验位)中1的个数为奇数

  偶校验：使所有传送的数位(含校验位)中1的个数为偶数

- 奇偶校验能够检测出1位误码，但是不能纠错。

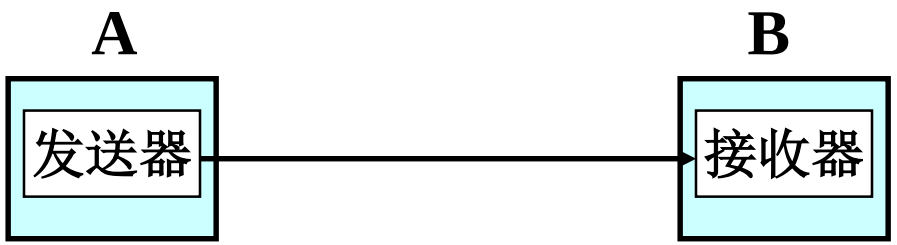
3. 串行数据传送方式

按照数据流的方向，
分成三种基本的传送方式：

- 单工方式
- 全双工方式
- 半双工方式

单工方式

只允许数据按照一个固定的方向传送.



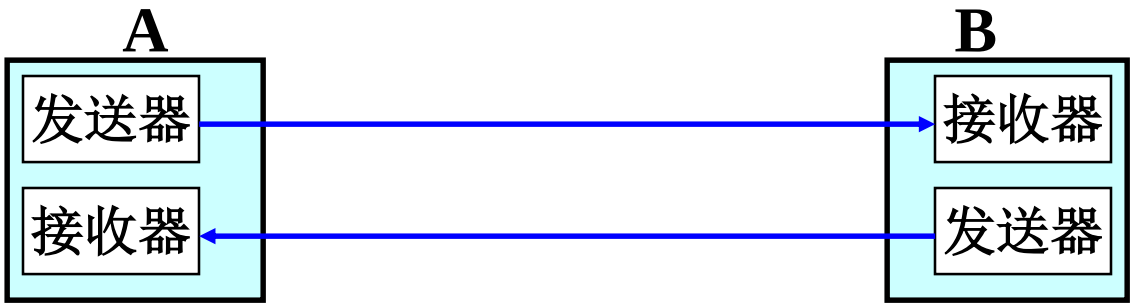
A方只能发送，称发送器

B方只能接收，称接收器

如： 鼠标与主机的通信采用单工方式.

全双工方式

允许通讯双方同时进行发送和接收操作

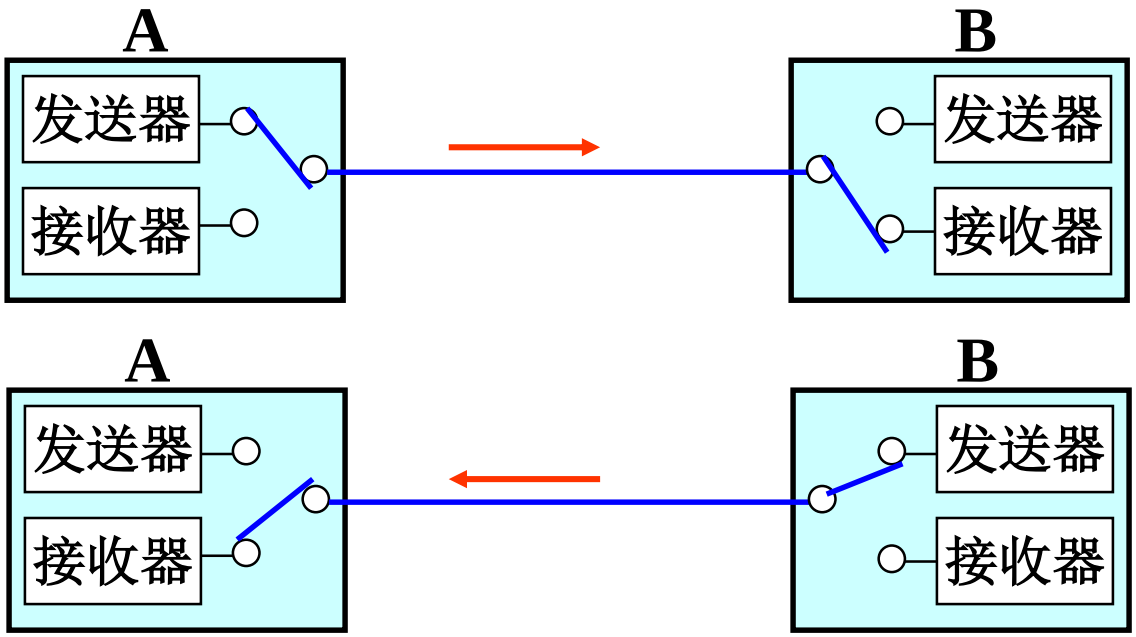


相当于把两个方向相反的单工方式组合在一起，需要两条传输线。

全双工方式主要应用于实时性较强的交互式应用中，如计算机之间的通信等。

半双工方式

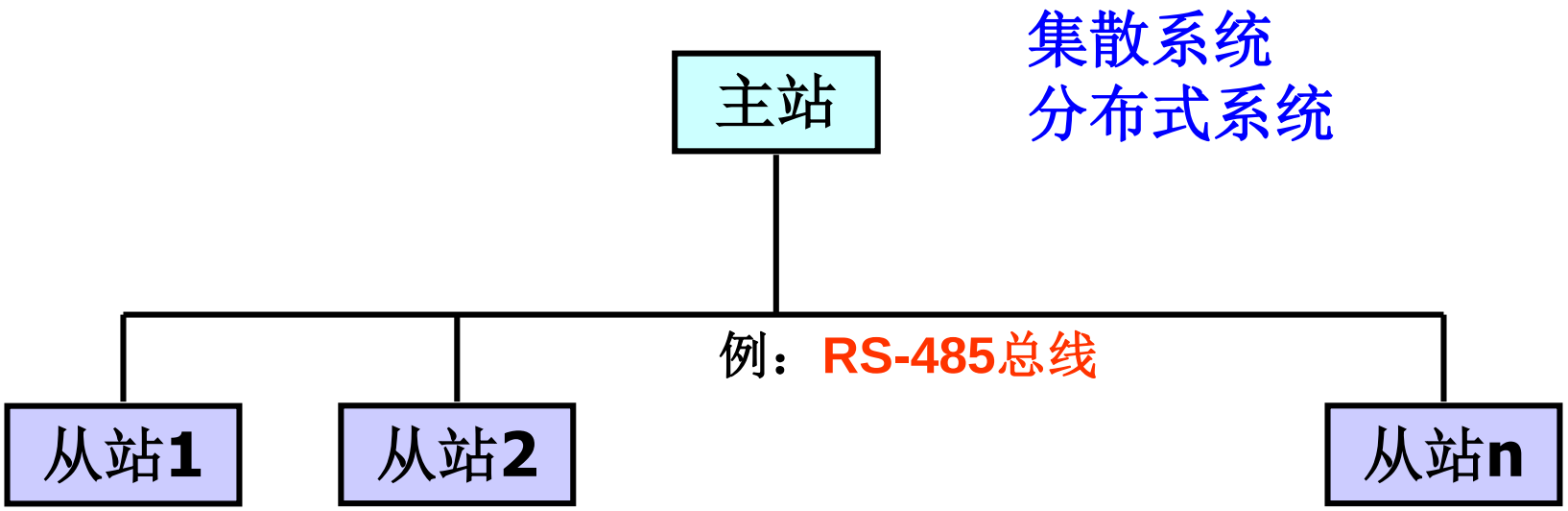
数据能从A方传送到B方，也能从B方传送到A方，
但是不能同时在两个方向上传送，
每次只能由一方发送，另一方接收。



通信双方通过软件控制的电子开关，进行方向的切换，
轮流地进行发送和接收。

一些简单的外部设备如键盘与主机的通信采用半双工方式.

半双工方式广泛地应用于主从结构的系统中



- 每个站均有通信地址(定址原则）
- 平时从站都处于接收状态，等待来自主站的命令
- 当从站接收到来自主站的命令后，若需要响应，
  则将自己置为发送状态，并发送响应，发送完毕后再置为接收状态
- 当主站要发送命令时，置主站为发送状态
  当主站发送命令完毕后，置主站为接收状态，以接收从站的响应

4. 波特率 (Baudrate)

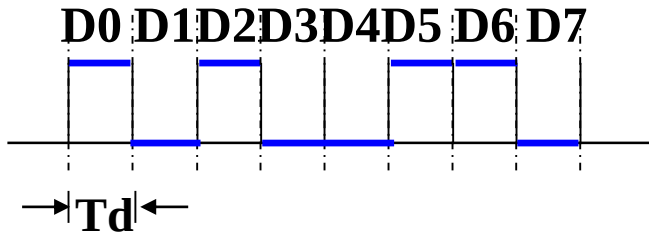
并行通信中，传输速率是以每秒多少字节(B/s)表示。

串行通信中，衡量数据传输速率的单位是波特率，

即每秒传送的二进制数据的位数，以位/秒(bps)表示。

有时也用“位周期”Td表示传输速率，波特率是位周期的倒数

波特率= 1/Td



标准的波特率系列有：

110、300、600、1200、1800、2400、4800、9600和19200等.

5. 通信协议

要想保证通信成功，通信双方必须有一系列的约定，如：
作为发送方，必须知道什么时候可以发送信息，对方是否收到，
收到的内容有没有错，要不要重发、怎么通知对方结束等
作为接收方，必须知道对方是否发送信息、发的是什么、
收到的信息是否有错、如果有错怎么通知对方、怎么判断结束等

通信协议就是通信双方为了保证通信正确，

事先对数据传送控制规定的必须共同遵守的一种约定，

包括对数据格式、同步方式、传送速率、传送步骤、

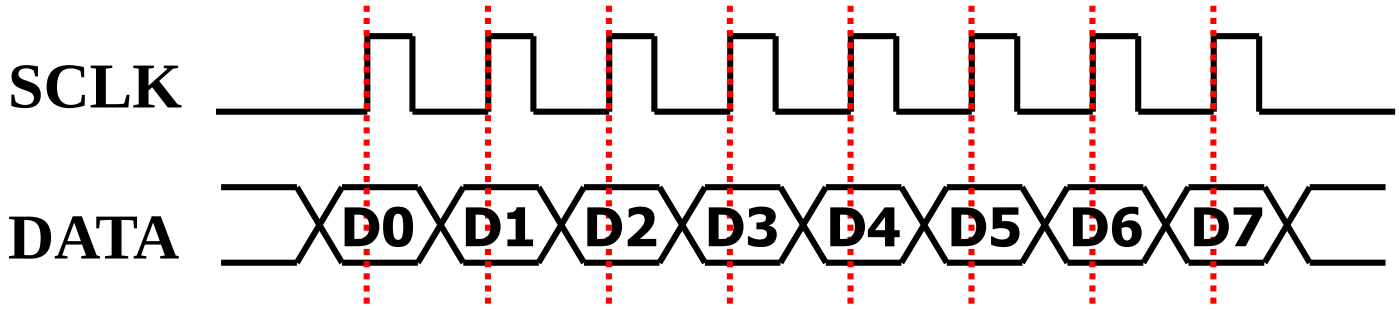
检纠错方式等问题做出的统一规定。

通信协议的实现可由硬件或软件实现.

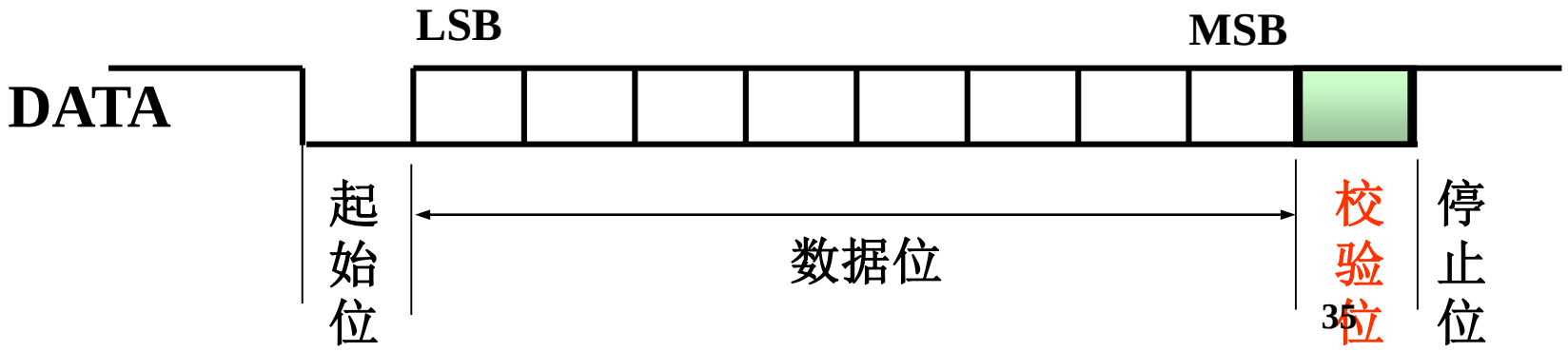
6. 串行通信基本方式

根据对数据流的分界、定时和同步方式的不同，串行通信的基本方式可以分为两种类型：

同步串行：使用独立的同步时钟信号线来实现位同步，用时钟控制数据的传送。



异步串行：不使用独立的同步时钟信号线，位同步是由事先约定好的波特率、并在传送的信息中设置起始位、停止位来实现



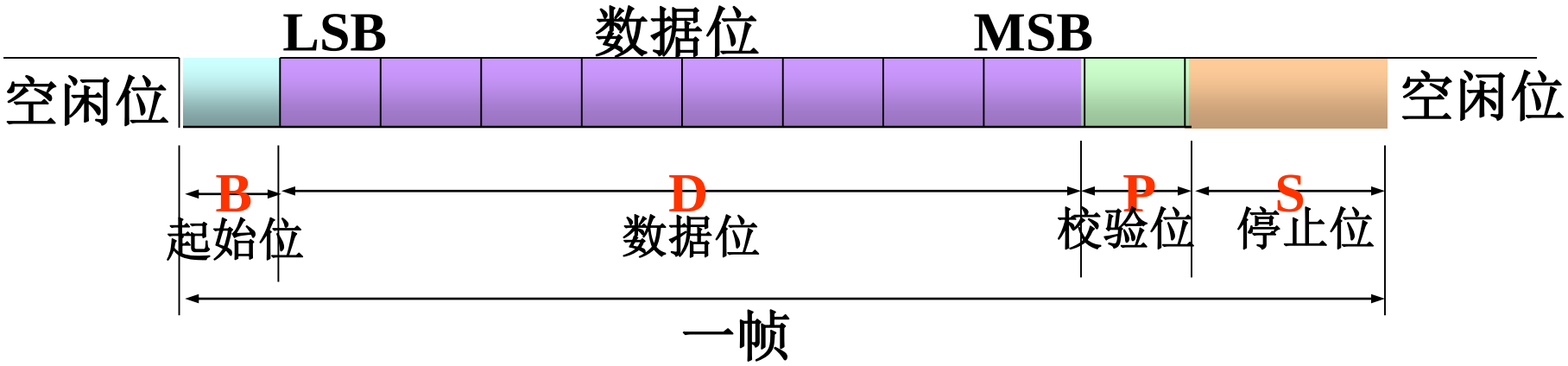
7. 异步串行通信

数据是一帧(Frame)一帧传送，

每一帧包含起始位、数据位、校验位、停止位，

帧与帧之间可有任意个空闲位。

异步串行数据格式为：

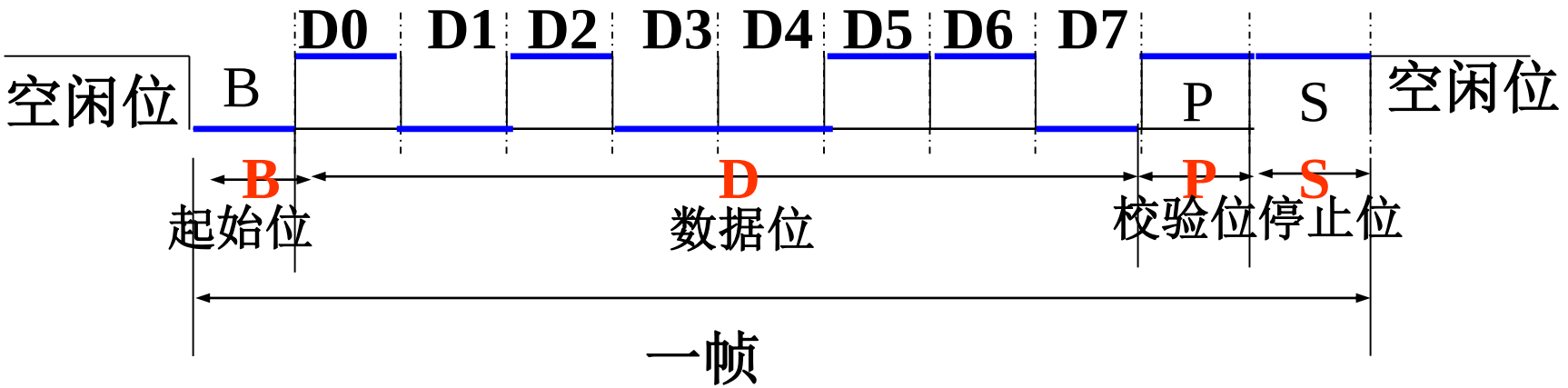


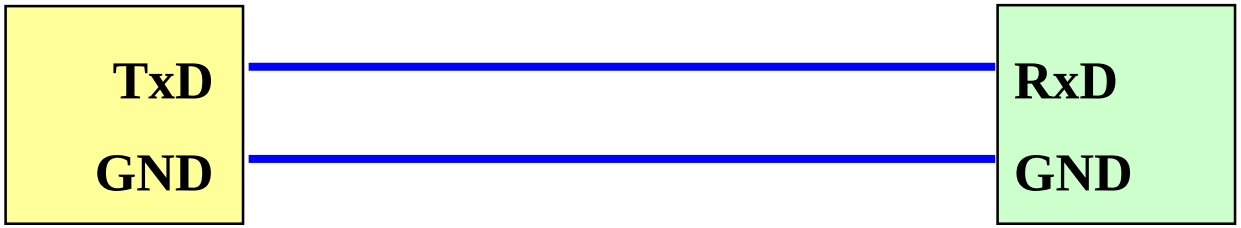
起始位B	逻辑0	1位
数据位D	逻辑0或1	5位~8位
校验位P	逻辑0或1	1位或无
停止位S	逻辑1	1~2位
空闲位	逻辑1	任意数量

例

传送8位数据65H(01100101B)，先低位后高位发送，奇校验，1个停止位，

信号线上的波形为：



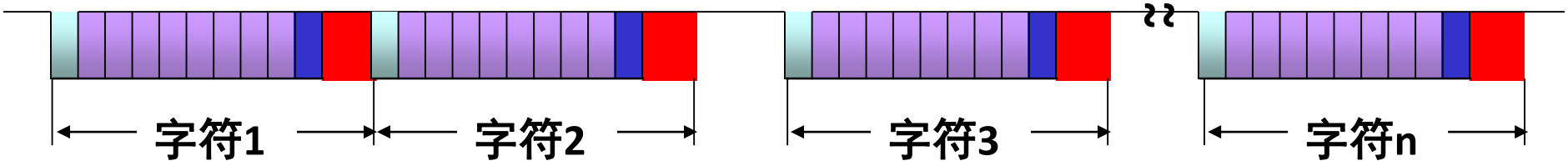


异步串行通信特点

- 异步串行通信以帧为信息单位进行传送，1帧包含1个字符
- 在数据格式中设置起始位和停止位来协调“同步”
- 异步串行通信中字符与字符之间通信没有严格的定时要求，每个字符作为一个独立的单位，可以随机出现在数据流中，即每个字符在数据流中出现的时间是任意的
- 异步串行通信中位与位之间有严格的定时

  一旦传送开始，收/发双方则按预先约定的传输速率，

  发/收双方在各自时钟的作用下，传送/接收字符中的 每1位
- 异步串行通信适合于发送数据不连续、传送数据量较少，或对传输率要求不高的场合。



数据块的传输

8. 接收/发送时钟和波特率因子

接收/发送时钟

在串行通信中，无论发送或接收，

都必须有时钟脉冲信号对传送的数据进行同步和定位控制，

这就需要有接收/发送时钟。

同步，就是确认通信什么时候开始

定位，就是确认每一个的数据位

波特率因子

为了提高串行通信的抗干扰能力，

往往用多个时钟调制1位二进制数据，

调制1位二进制数据所用的收/发时钟个数称为波特率因子，

用n表示波特率因子，则：

收/发时钟频率 = 波特率 × 波特率因子n

例如 波特率因子为16，则16个时钟脉冲发/收1位

n可以是1, 16，32，64等

以n=16为例，

接收器以数据波特率16倍的时钟对所接收的数据进行检测:

首先正确地检测到起始位，然后逐位确定各个数据位。

具体过程如下：

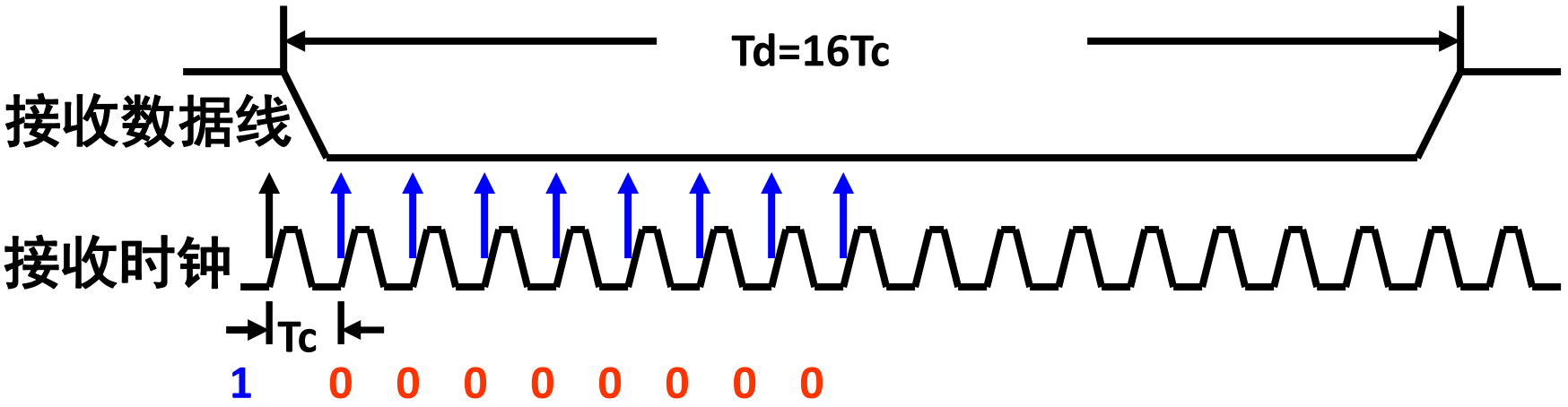
接收器在每个接收时钟的上升沿采样接收数据线，

当发现接收数据线出现低电平时，就认为是起始位的开始，

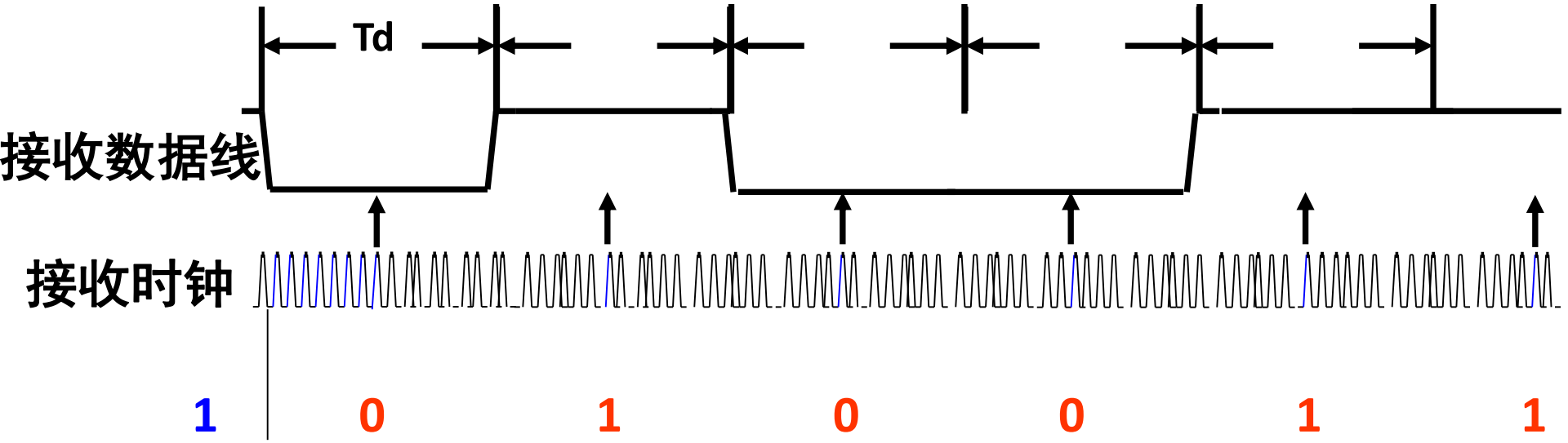
若在连续的8个时钟周期内检测到接收数据线仍保持低电平，

则确认它为起始位，而不是干扰信号。

起始位的确定



数据位的采样



通过这种方法，

不仅能够排除接收线上的噪声干扰，识别假起始位，

而且能够相当精确地确定起始位的中间点，

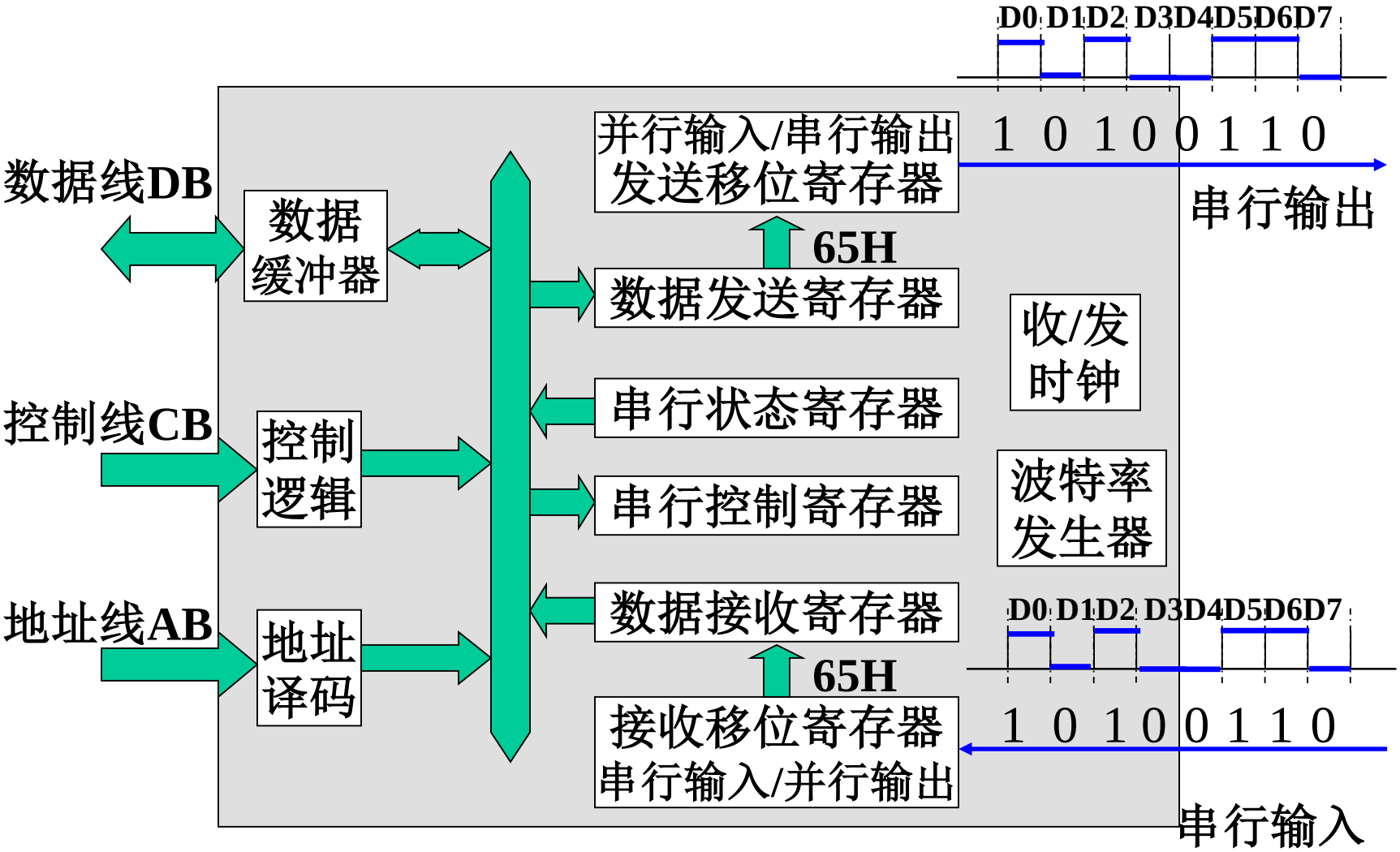
从而提供一个准确的时间基准，

从这个基准算起，每隔16个Tc采样一次数据线，作为输入数据。

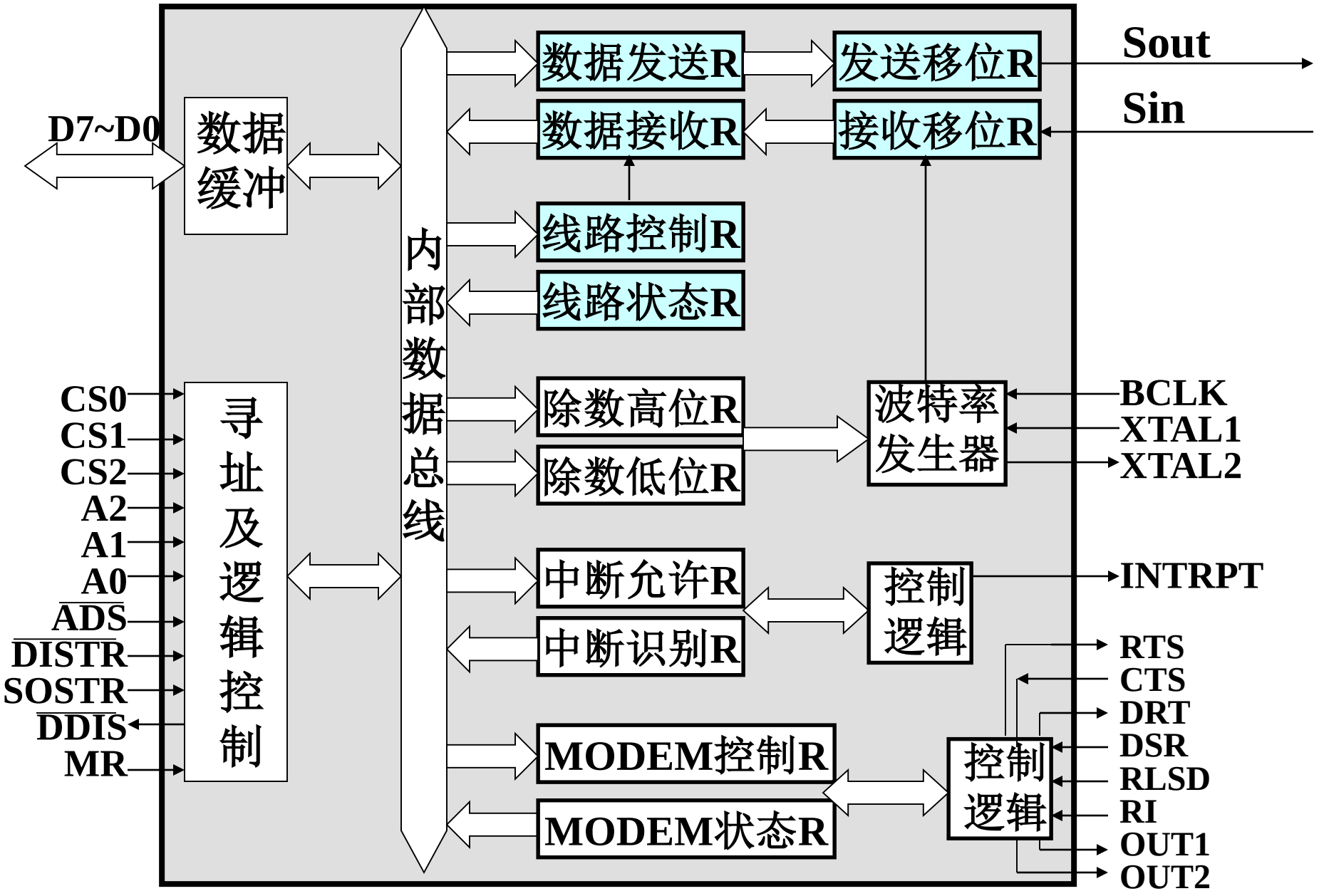
二、典型异步串行通信接口简介

1. 串行通信接口典型结构图
2. 串行通信有关寄存器
3. 串行发送数据过程
4. 串行接收数据过程

1. 串行通信接口典型结构图



例　PC机异步通信接口INS8250结构图



2. 串行通信有关寄存器

- 发送数据寄存器
- 发送移位寄存器
- 接收移位寄存器
- 接收数据寄存器
- 串行控制寄存器
- 串行状态寄存器

数据发送寄存器

用于存放要发送的并行数据,

如检测到发送移位寄存器为空,

自动将保存的数据传送到发送移位寄存器.

发送移位寄存器(并行输入/串行输出)

用于将并行数据转换成串行数据,

并通过串行输出管脚发送数据.

CPU不能直接对发送移位寄存器存取操作.

接收移位寄存器(串行输入/并行输出)

用于接收从串行输入管脚传入的数据，并转换成并行数据.

当接收到1字符的串行数据，就自动传送给接收数据寄存器.

CPU不能直接对接收移位寄存器存取操作.

接收数据寄存器

用于存放由接收移位寄存器传送来的数据,

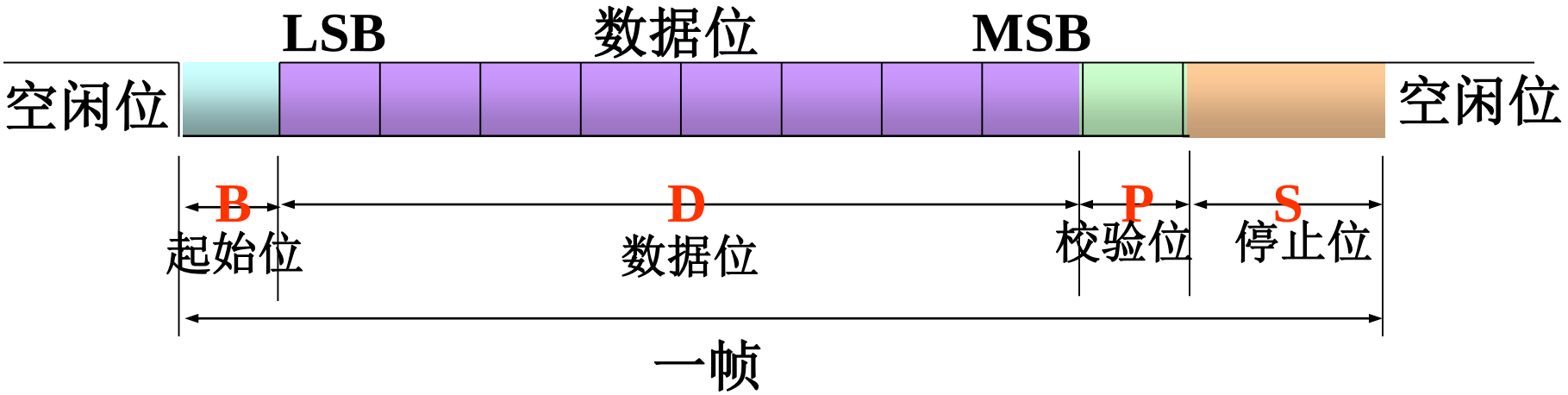
CPU通过对接收数据寄存器进行读操作，获取传入的数据.

串行控制寄存器

设置串行通信的数据格式，

包括波特率、停止位长度、数据位长度、奇偶校验方式的设置

异步串行数据格式为：



例如　某串行通信接口控制寄存器的含义

D7	D6	D5	D4	D3	D2	D1	D0

- **D7~D5: 波特率设置**

  000 /110bps　001/150bps　010/300bps　011/600bps

  100 /1200bps　101/2400bps　110/4800bps　111/9600bps

- **D4~D3: 奇偶校验设置**

  00/无校验　　01/奇校验　　10/无校验　　11/偶校验

- **D2　　: 停止位长度设置**

  0:　1位　　1:2位

- **D1~D0: 数据位长度设置**

  10: 7位　　11:8位

串行状态寄存器

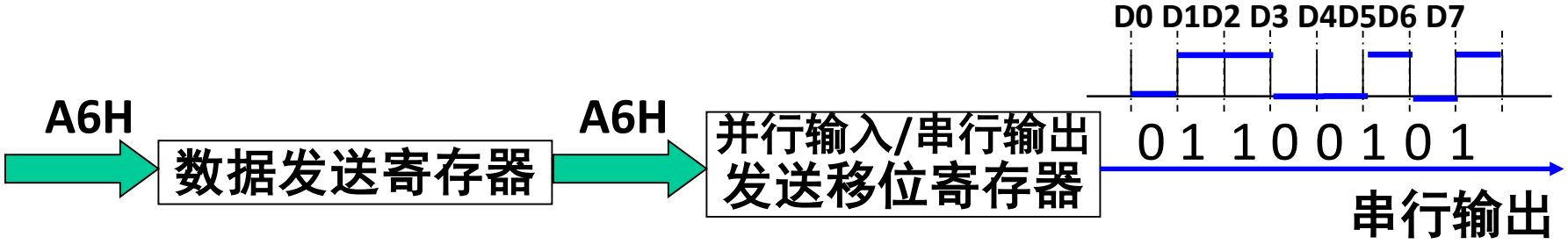
存放串行通信的状态，

包括发送数据寄存器，接收数据寄存器，通信过程的状态等.

通信口状态参数含义

D7	D6	D5	D4	D3	D2	D1	D0

- D6 : 1=发送数据寄存器空
- D3 : 1=帧格式错误
- D2 : 1=奇偶校验错误
- D1 : 1=溢出错误
- D0 : 1=接收数据寄存器满



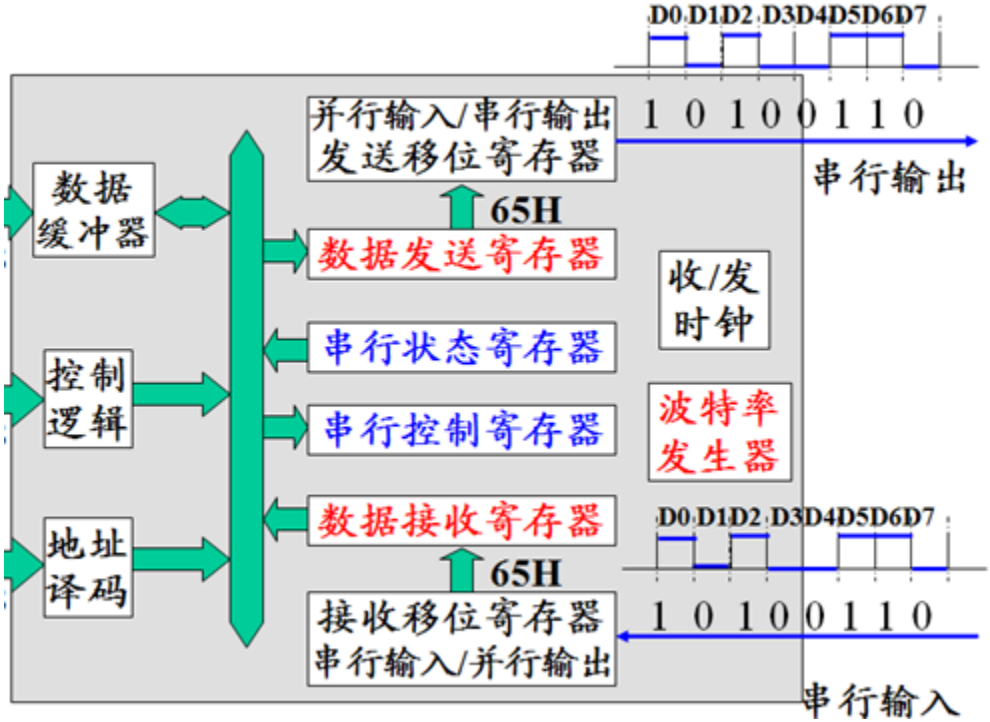
串行状态寄存器

D7	D6	D5	D4	D3	D2	D1	D0

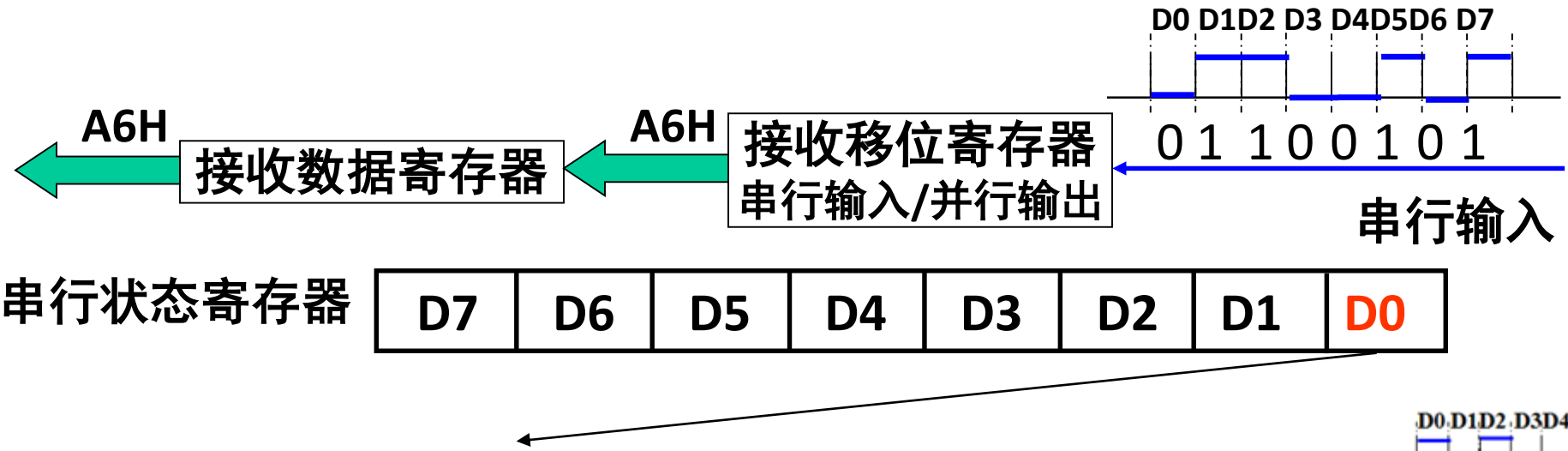
发送数据寄存器的状态

满: 当发送数据寄存器的数据未传送到移位寄存器处于满状态；

空: 当发送数据寄存器的数据传送到移位寄存器，新的数据还未写入发送数据寄存器时，处于空状态



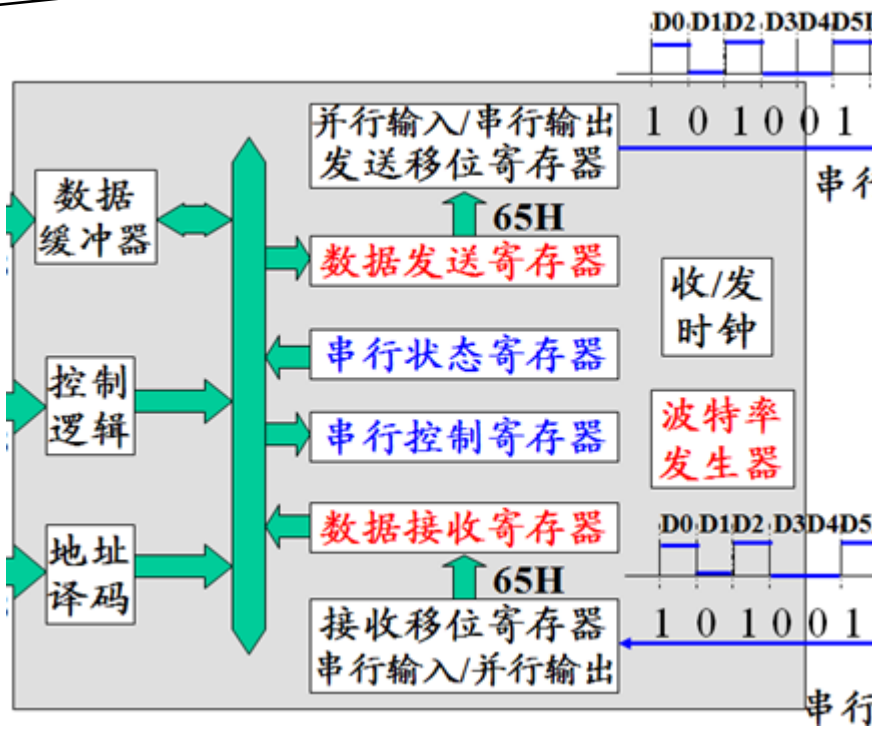
CPU可通过检测发送数据寄存器是否为空状态，确定是否可以发送新数据.



接收数据寄存器的状态

满: 当接收移位寄存器从串行输入管脚接收一个字节的数据，并转换成并行数据自动存放到接收数据寄存器后，接收数据寄存器处于满状态.

空: 当接收数据寄存器的数据被取走，但新的数据未传入，数据接收寄存器处于空状态，表示没有新的数据到来.



CPU可通过检测接收数据寄存器是否为满状态，确定是否可以接收新数据.

接收/发送通信过程状态

D7	D6	D5	D4	D3	D2	D1	D0

串行状态寄存器

奇偶校验错误

通信线上的噪音引起某些数据位的改变，
使接收数据位的奇偶数与设定的奇偶校验方式不一致，
从而产生奇偶校验错误。

例 传送方发送8位数据65H(01100101B)，奇校验，1个停止位，
数据格式为 01010011011 (含校验位在内1的个数5为奇数)

如果接收方按同样的通讯协议接收数据，
假如收到的数据为 01011011011
由于收到1的个数为6, 为偶数，产生奇偶校验错误。

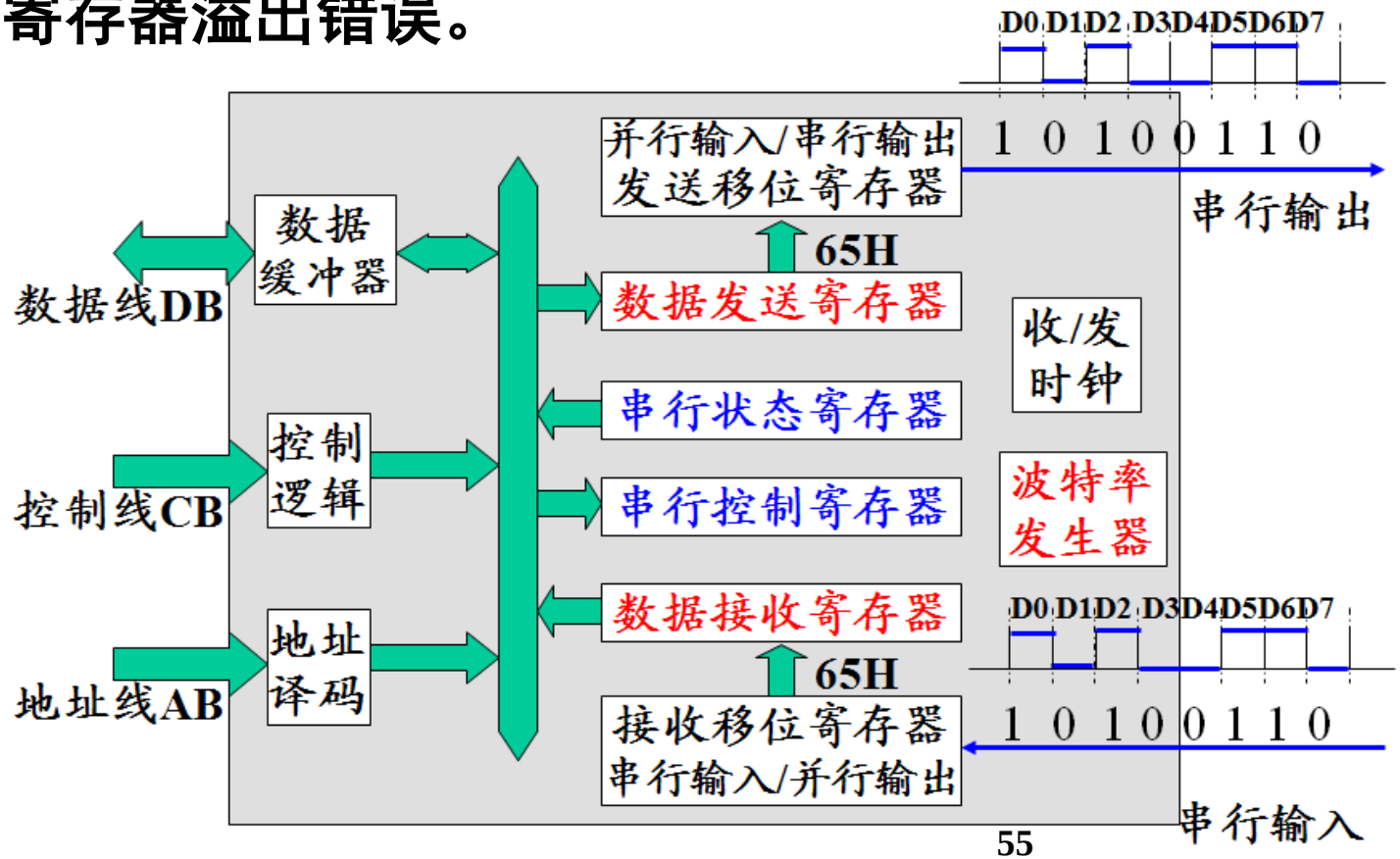
(注意最后1位“1”是停止位，不在检验数据之内)

D7	D6	D5	D4	D3	D2	D1	D0

串行状态寄存器

溢出错误

当上一个存在接收数据寄存器的数据还没有被CPU取走，
又有字符传送到接收数据寄存器时，
产生接收数据寄存器溢出错误。



D7	D6	D5	D4	D3	D2	D1	D0

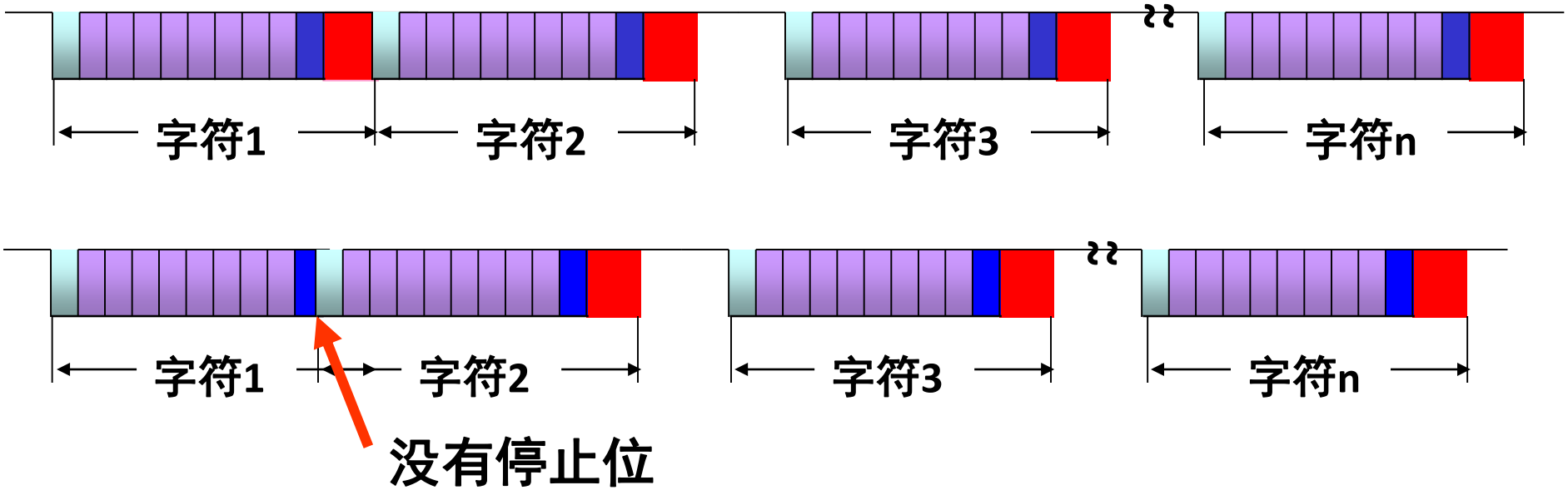
串行状态寄存器

帧格式错误

当接收的数据没有停止位时，产生帧格式错误，

这种错误可能是由于通信线上的噪音引起停止位的丢失，

或是由于接收方和发送方通信协议初始化不匹配引起。



3. 串行发送数据过程

CPU采用查询方式检测到发送数据寄存器空状态后，

将待发送的数据写入发送数据寄存器中；

当发送移位寄存器空时，

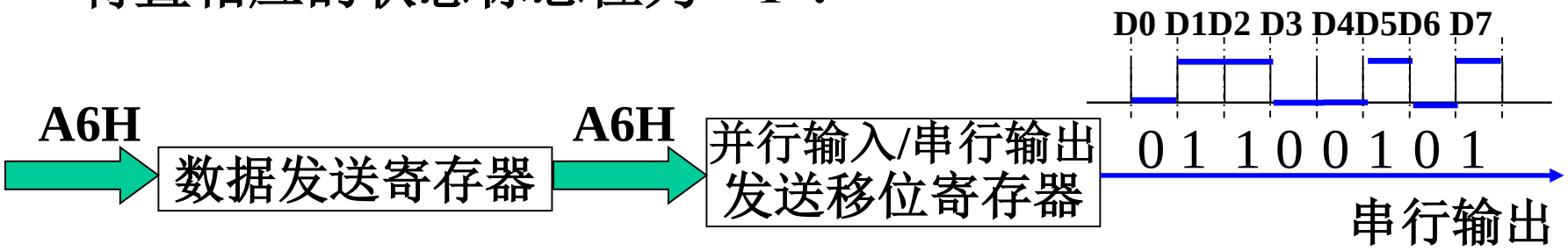
数据发送寄存器会自动将数据传送到移位寄存器，

发送控制逻辑将自动按规定的数据格式构成信息帧，

然后逐位由串行输出管脚输出；

当发送数据寄存器空时，

将置相应的状态标志位为“1”.



4. 串行接收数据过程

串行的位信息由串行输入管脚输入，

接收侧监测通信线路, 如果检测出一个起始位, 就进行内部同步，

开始在接收时钟控制下采样串行输入管脚上的数据，

移位寄存器将接收的串行数据，转换成并行数据，

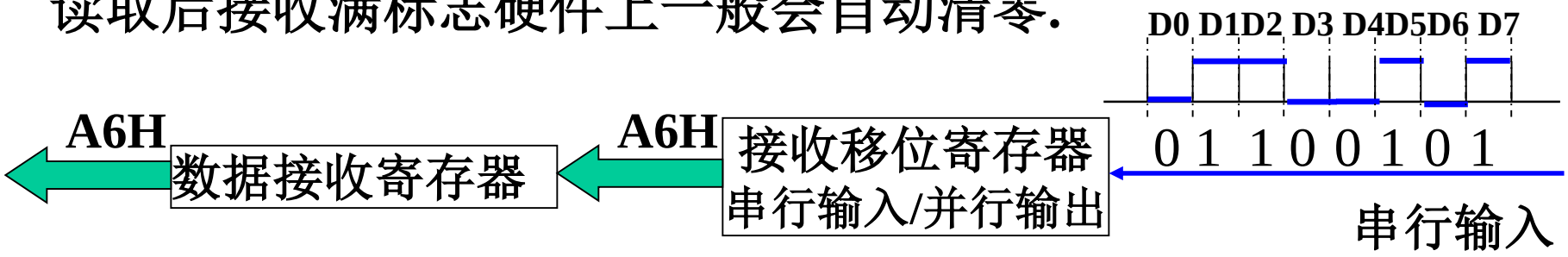
自动存放在接收数据寄存器中，

并置数据接收寄存器满标志位为“1”;

CPU通过查询方式检测到数据接收寄存器满状态后，

可读取接收数据寄存器中的数据，将数据取走;

读取后接收满标志硬件上一般会自动清零.



三、PC机的标准串口

1. 早期的PC机一般有1或2个异步串行通信接口，
   组装在主板上，称为COM1和COM2
2. 使用的串行通信接口芯片是INS8250,
   或与之兼容的串口芯片(如16550等).
   8250是一种可编程的异步串行通信接口芯片，
   支持异步通信协议，可通过编程改变传送数据的波特率，
   提供与MODEM连接的联络信号，实现远程通信。
3. 两串口的结构相同，只是占用的系统资源不同，
   COM1：3F8~3FFh, 中断IRQ4
   COM2：2F8~2FFh, 中断IRQ3

串口1　　并口

4. 现在的PC机(特别是笔记本电脑)一般配置有USB口，未配置串口，可利用USB/串口转换模块，转换出一个串口。

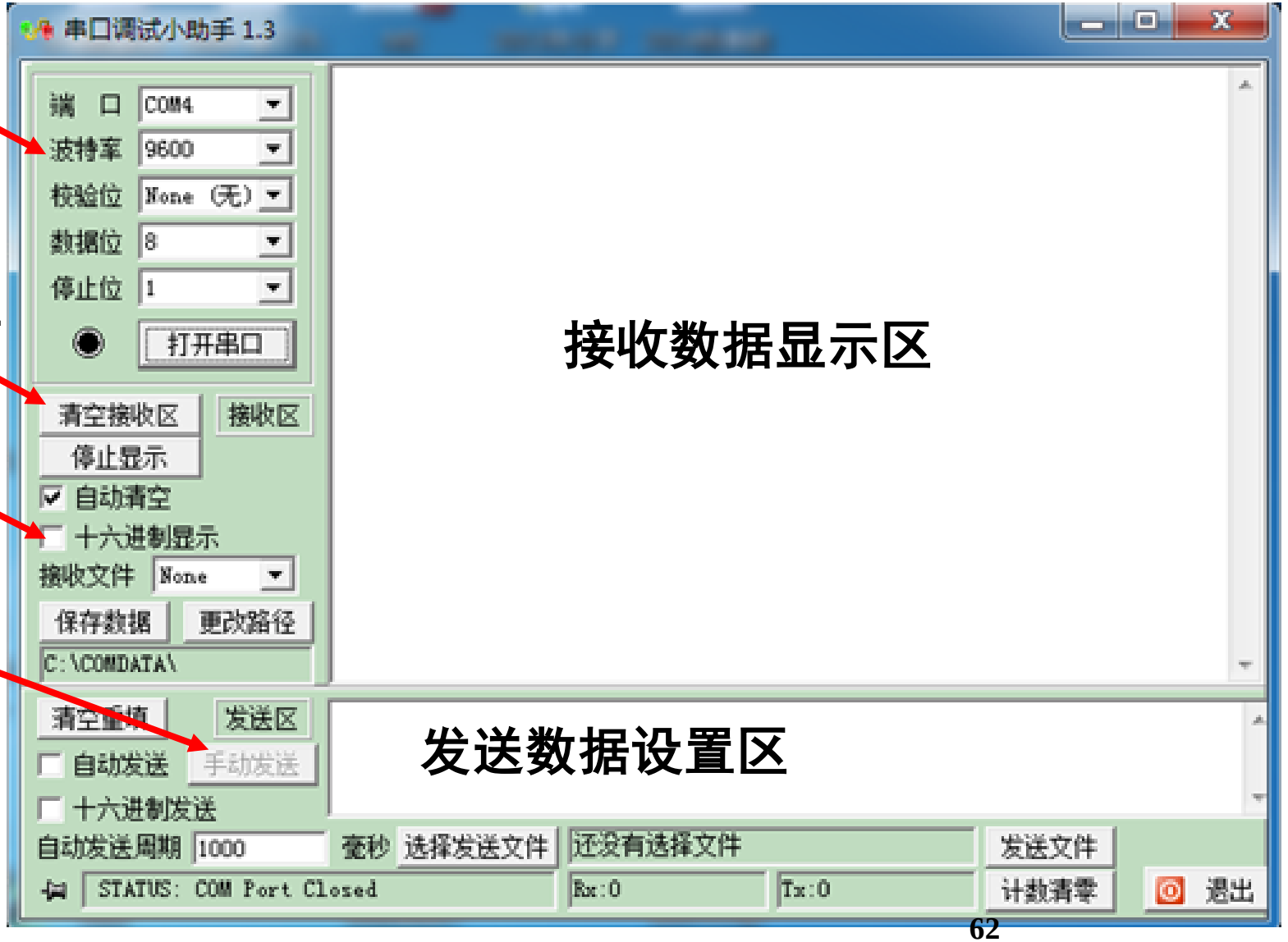
USB/串口转换电路模块

实验室有这样的模块，
做项目时可单独借

电脑利用USB接口通过
USB/串口模块转换出一个串口

可用串口调试助手一类的软件对PC机上的串口进行控制

串口调试助手界面



4. 串行通信接口标准RS-232

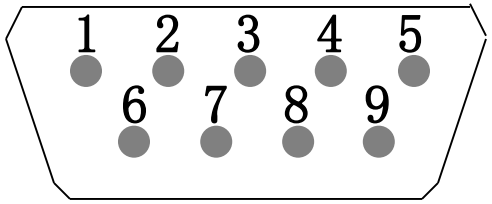
- 在串行通信中，设备之间的连接要符合接口标准
- 计算机通信中使用最普遍的是：

  美国电子工业协会(EIA)推荐使用的RS-232C标准

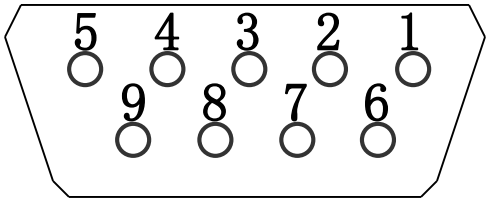
  EIA——Eliectronic Industries Association
  美国电子工业协会

  RS——Recommend Standard　推荐标准

  C表示第三版本，之前有过A、B版
- PC机上的COM1、COM2连接器，符合RS-232C接口



针型DB-9M



孔型DB-9F

引脚号	符号	信号名称	方向
1	DCD	载波检测	输入
2	RxD	接收数据	输入
3	TxD	发送数据	输出
4	DTR	数据终端就绪	输出
5	GND	信号地	
6	DSR	数据设备就绪	输入
7	RTS	请求发送	输出
8	CTS	允许发送	输入
9	RI	振铃指示	输出

RS-232C采用负逻辑，TTL采用正逻辑，

RS-232C信号电平与TTL不兼容

	RS-232C标准	TTL标准
逻辑0	+5 ~ +15V	0.3 ~ 1.0V
逻辑1	-5 ~ -15V	2.5 ~ 5.0V

一般串行接口芯片如8250、8251、

单片机内（如msp430、串行接口均使用TTL电平，

使用电平转换电路才能与RS-232C连接器连接。

第2节　MSP430通用串行通信模块 USCI

(*Universal Serial Communication Interface*)

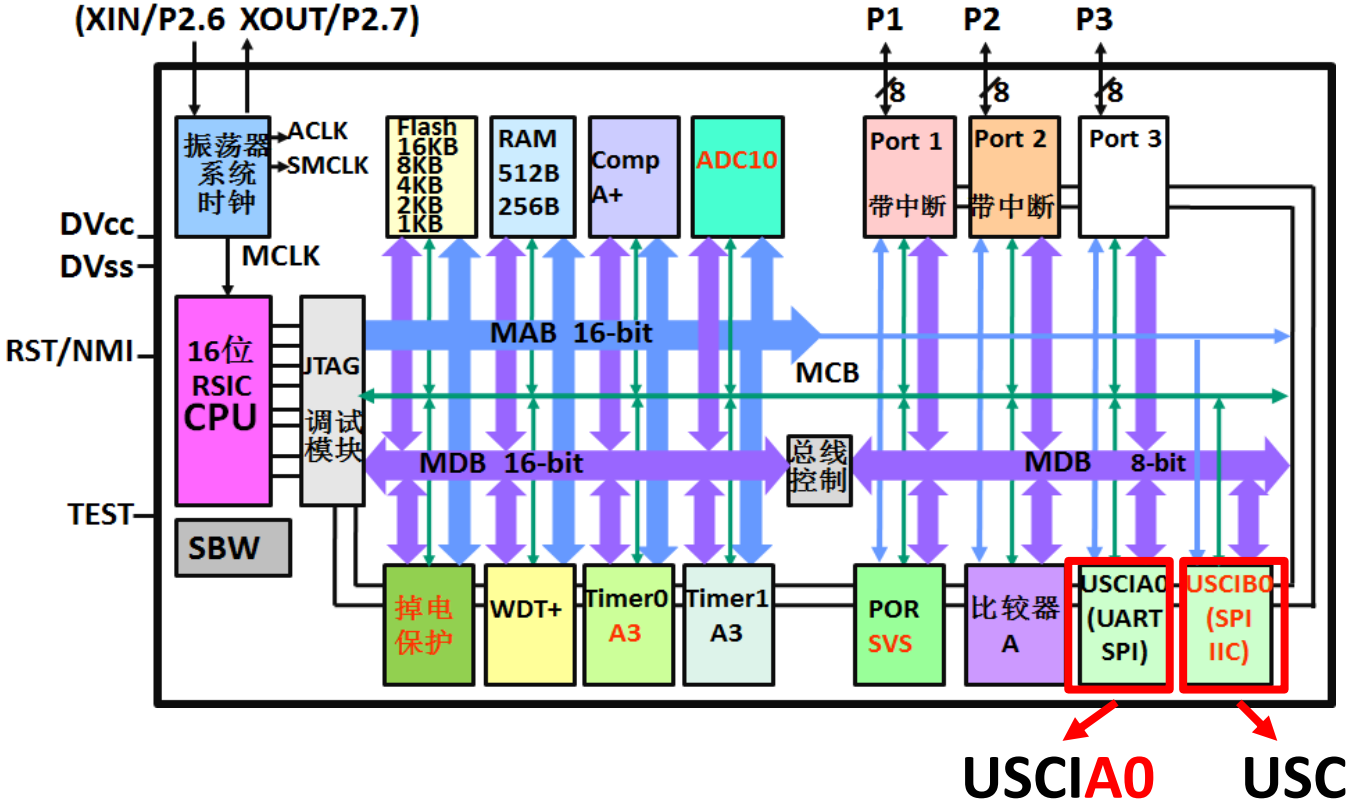
一、USCI 模块简介

二、USCI模块异步串行方式编程结构

三、利用USCI模块 异步串行方式通信

一、USCI模块简介

MSP430G2553内有两个通用串行通信接口USCI



通过MCU芯片的数据手册，如msp430G255.pdf，可了解芯片内包含的哪些具体USCI模块。

- MSP430的USCI模块支持多种串行通信模式：
  如 异步、同步、I2C等
- 不同USCI模块支持的具体通信方式不同，用字母区分：
  如 USCI_A 、USCI_B
- 若MCU内有多个相同的USCI模块，用数字区分
  如USCI_A0、USCI_A1

- msp430G2553 包含USCI_A0和USCI_B0 两个USCI模块
- USCI_Ax 模块支持：
  - ✓ 异步串行通信 UART mode
  - ✓ 同步串行通信 SPI mode
  - ✓ 支持自适应波特率检测(不介绍，不要求)
  - ✓ 红外通信 Pulse shaping for IrDA communications(不介绍，不要求)
- USCI_Bx模块支持:
  - ✓ SPI mode
  - ✓ I2C 模式

USCI通用串行通信模块的异步和同步串行通信

异步串行通信方式 UART Mode

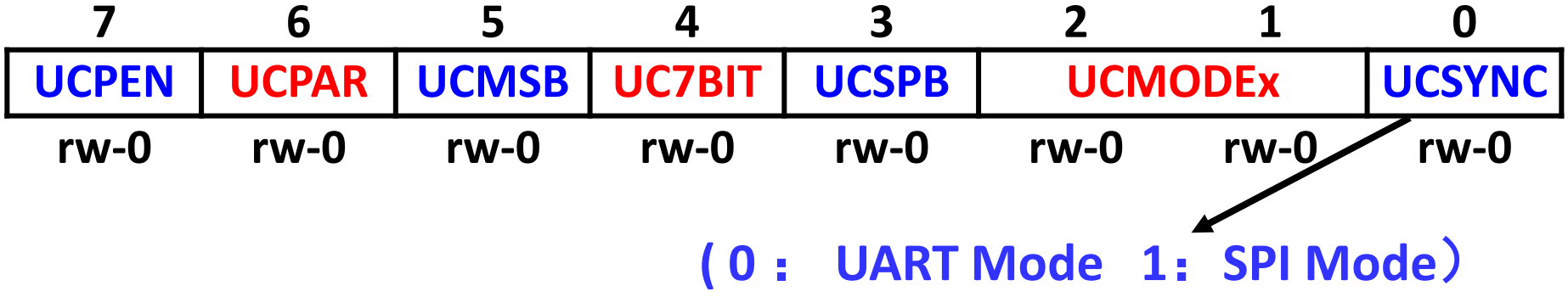
同步串行通信方式 SPI Mode

UART: Universal Asynchronous Receive/Transmit

SPI: Synchronous peripheral interface

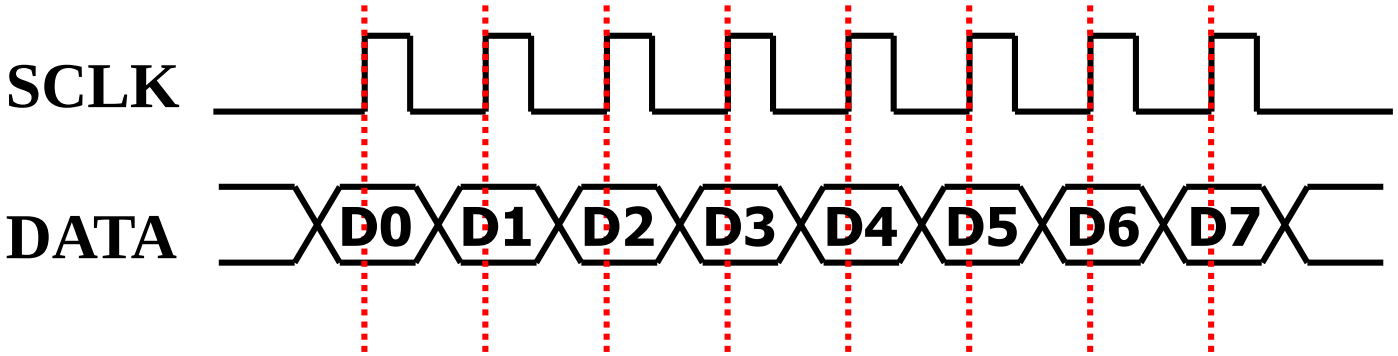
由USART 控制寄存器UxCTL中的SYNC位决定工作方式

USCI_A0串口控制寄存器0（UCA0CTL0）

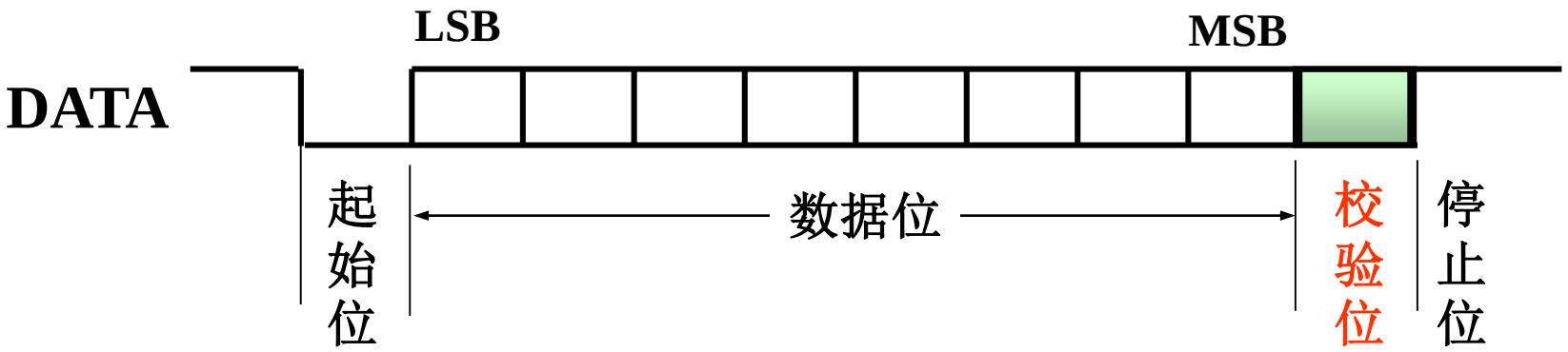


异步串行通信与同步串行通信

同步串行：使用独立的同步时钟信号线来实现位同步



异步串行：不使用独立的同步时钟信号线，
位同步靠起始位、停止位等实现



USCI 异步串行通信特性

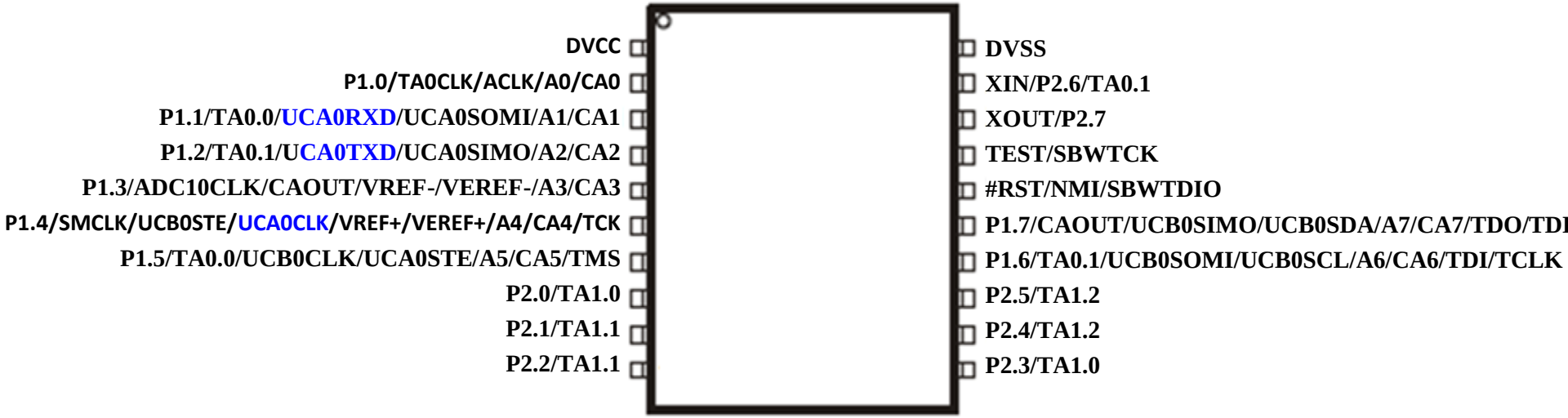
- 数据可7位或8位，可选择 奇校验、偶校验或无校验
- 可选择数据先从低位(LSB)还是高位 (MSB)发送
- 独立的发送中断和接收中断
- 支持带地址位的多机串行通信(不介绍、不要求)

■ 与USCI_A0异步串行通信有关引脚

P1.4/UCA0CLK：外部时钟输入

P1.1/UCA0RXD：USCI_A0异步串行数据接收

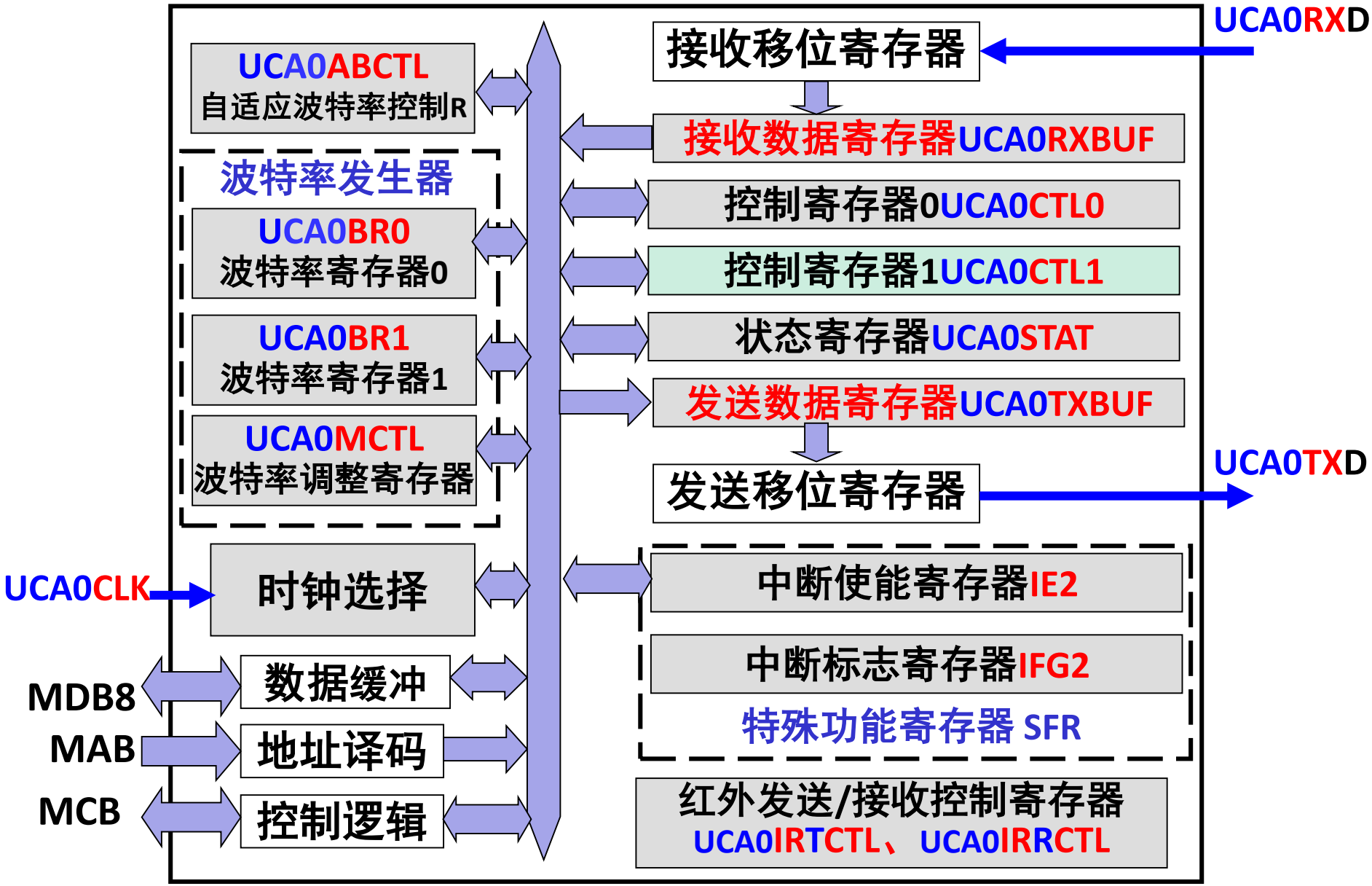
P1.2/UCA0TXD：USCI_A0异步串行数据发送



引脚名	x	功能	相关控制位的设置				
			P1DIR.x	P1SEL.x	P1SEL2.x	ADC10AE.x INCH.x=1	CAPD.y
P1.1/ UCA0RXD/	1	UCA0RXD	X (From USCI)	1	1	0	0
P1.2/ UCA0TXD/	2	UCA0TXD	X (From USCI)	1	1	0	0

```
P1SEL |= BIT1+BIT2;   //置P1.1、P1.2为USCI_A0的收/发引脚
P1SEL2 |= BIT1+BIT2;
```

二、 MSP430 USCI_A0异步串行方式的编程结构



USCI_A0串行通信编程方法

1）管脚功能选择　　PxSEL, PxSEL2

2）异步串行数据格式设置　　UCA0CTL0 控制寄存器0

UCA0CTL1 控制寄存器1

3）波特率设置　　UCA0BR0 波特率寄存器0

UCA0BR1 波特率寄存器1

UCA0MCTL 波特率调整寄存器

4）数据收/发方式　　UCA0RXBUF 数据接收寄存器

UCA0TXBUF 数据发送寄存器

IFG2 中断标志寄存器2

IE2 中断使能寄存器2

USCI_A0串行通信编程方法

2）异步串行数据格式设置

UCA0CTL0 控制寄存器0

UCA0CTL1 控制寄存器1

■ UCA0CTL0 USCI_A0串口控制寄存器0

7	6	5	4	3	2	1	0
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB			UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

校验允许位
0: 禁止
1:允许

奇偶校验位
0: 奇校验
1: 偶校验

发送顺序
0:先发低位
1:先发高位

数据位长度
0: 8位
1: 7位

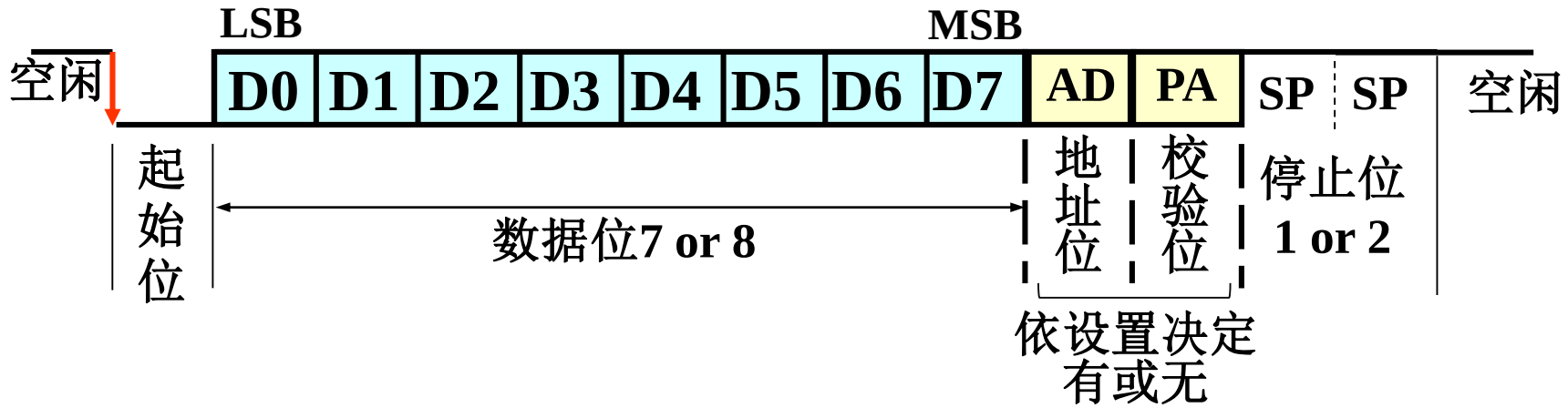
停止位长度
0:1位
1:2位

异步模式控制
00: 常规异步串行方式

只介绍常规异步串行方式

串行方式
0: 异步
1: 同步

异步串行通信数据格式



msp430g2553.h中UCA0CTL0 各个位的符号定义

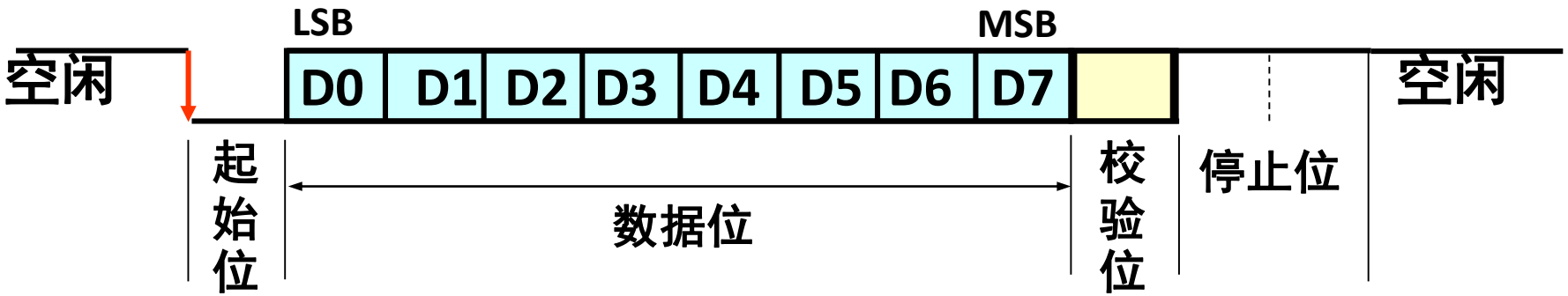
```
#define  UCPEN          (0x80)      /* Async. Mode: Parity enable */
#define  UCPAR          (0x40)      /* Async. Mode: Parity     0:odd / 1:even */
#define  UCMSB          (0x20)      /* Async. Mode: MSB first  0:LSB / 1:MSB */
#define  UC7BIT         (0x10)      /* Async. Mode: Data Bits  0:8-bits / 1:7-bits */
#define  UCSPB          (0x08)      /* Async. Mode: Stop Bits  0:one / 1: two */
#define  UCSYNC         (0x01)      /* Sync-Mode  0:UART-Mode / 1:SPI-Mode */
```

UCA0CTL0 USCI_A0串口控制寄存器0

7	6	5	4	3	2	1	0
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB			UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

例 编程设置USCI_A0异步串行数据格式为：

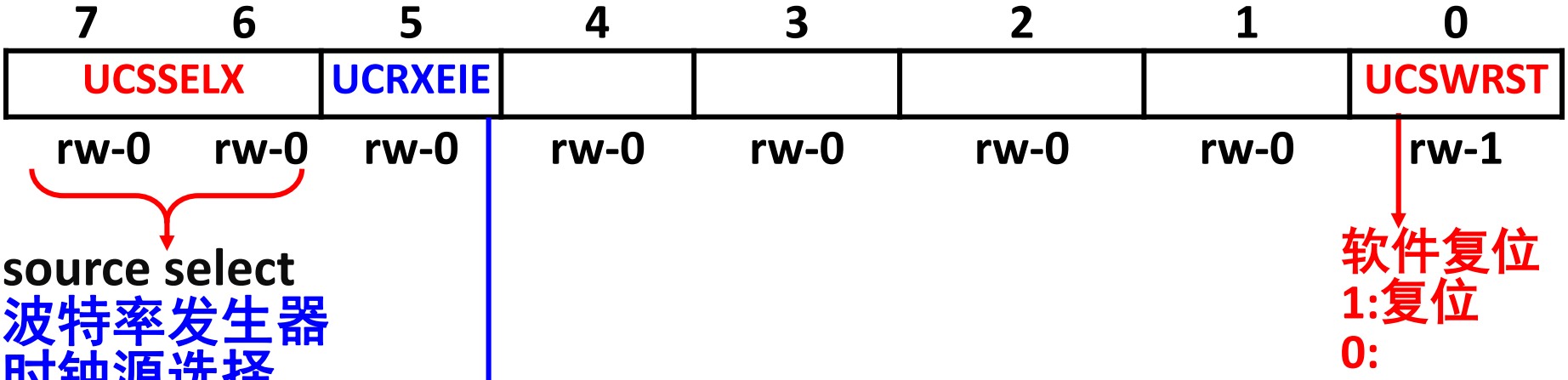
常规异步串行方式、 数据8位、先低位、偶校验、 2位停止位



	7	6	5	4	3	2	1	0
UCA0CTL0	UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB			UCSYNC
复位值	0	0	0	0	0	0	0	0
欲设置值	1	1	0	0	1	0	0	0

```
UCA0CTL0 |= UCPENA+UCPAR+UCSPB;
```

UCA0CTL1 USCI_A0串口控制寄存器1

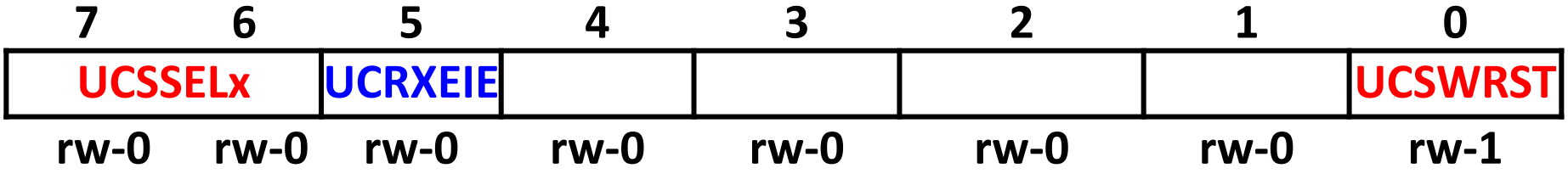


source select
波特率发生器
时钟源选择
0: UCLKI (外部提供时钟信号，注意不是晶振)
1: ACLK
2: SMCLK
3: SMCLK

软件复位
1:复位
0:

URXEIE=
- 0: 若接收到错误字符，则不存入 UxRXBUF，并且不置位 URXIFGx
- 1: 接收到的字符无论错对，都存入 UxRXBUF，并置位 URXIFGx

■ UCA0CTL1 USCI_A0串口控制寄存器1



msp430g2553.h中UCA0CTL1相关各个位和位域值的符号定义

```
#define UCSSEL1      (0x80)    /* USCI 0 Clock Source Select 1 */
#define UCSSEL0      (0x40)    /* USCI 0 Clock Source Select 0 */
#define UCRXEIE      (0x20)    /* RX Error interrupt enable */
#define UCSWRST      (0x01)    /* USCI Software Reset */
#define UCSSEL_0     (0x00)    /* USCI 0 Clock Source: 0 */
#define UCSSEL_1     (0x40)    /* USCI 0 Clock Source: 1 */
#define UCSSEL_2     (0x80)    /* USCI 0 Clock Source: 2 */
#define UCSSEL_3     (0xC0)    /* USCI 0 Clock Source: 3 */
```

例 编程设置USCI_A0波特率时钟源为SMCLK，
字符不论对错均存入U0RXBUF中,并设置URXIFG0

```
UCA0CTL1 |= UCSSEL1+UCRXEIE;
```

或

```
UCA0CTL1 |= UCSSEL_2+UCRXEIE;
```

例 编程设置USCI_A0波特率时钟源为ACLK，
字符不论对错均存入U0RXBUF中,并设置URXIFG0

```
UCA0CTL1 |= UCSSEL0+UCRXEIE;
```

或

```
UCA0CTL1 |= UCSSEL_1+UCRXEIE;
```

source select
波特率发生器
时钟源选择
0: UCLKI (外部时钟)
1: ACLK
2: SMCLK
3: SMCLK

```
#define  UCSSEL1      (0x80)
#define  UCSSEL0      (0x40)
#define  UCSSEL_0     (0x00)
#define  UCSSEL_1     (0x40)
#define  UCSSEL_2     (0x80)
#define  UCSSEL_3     (0xC0)
```

■ UCA0CTL1 USCI_A0串口控制寄存器1

7	6	5	4	3	2	1	0
UCSSELx		UCRXEIE					UCSWRST
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-1

USCI_A0串行通信编程方法

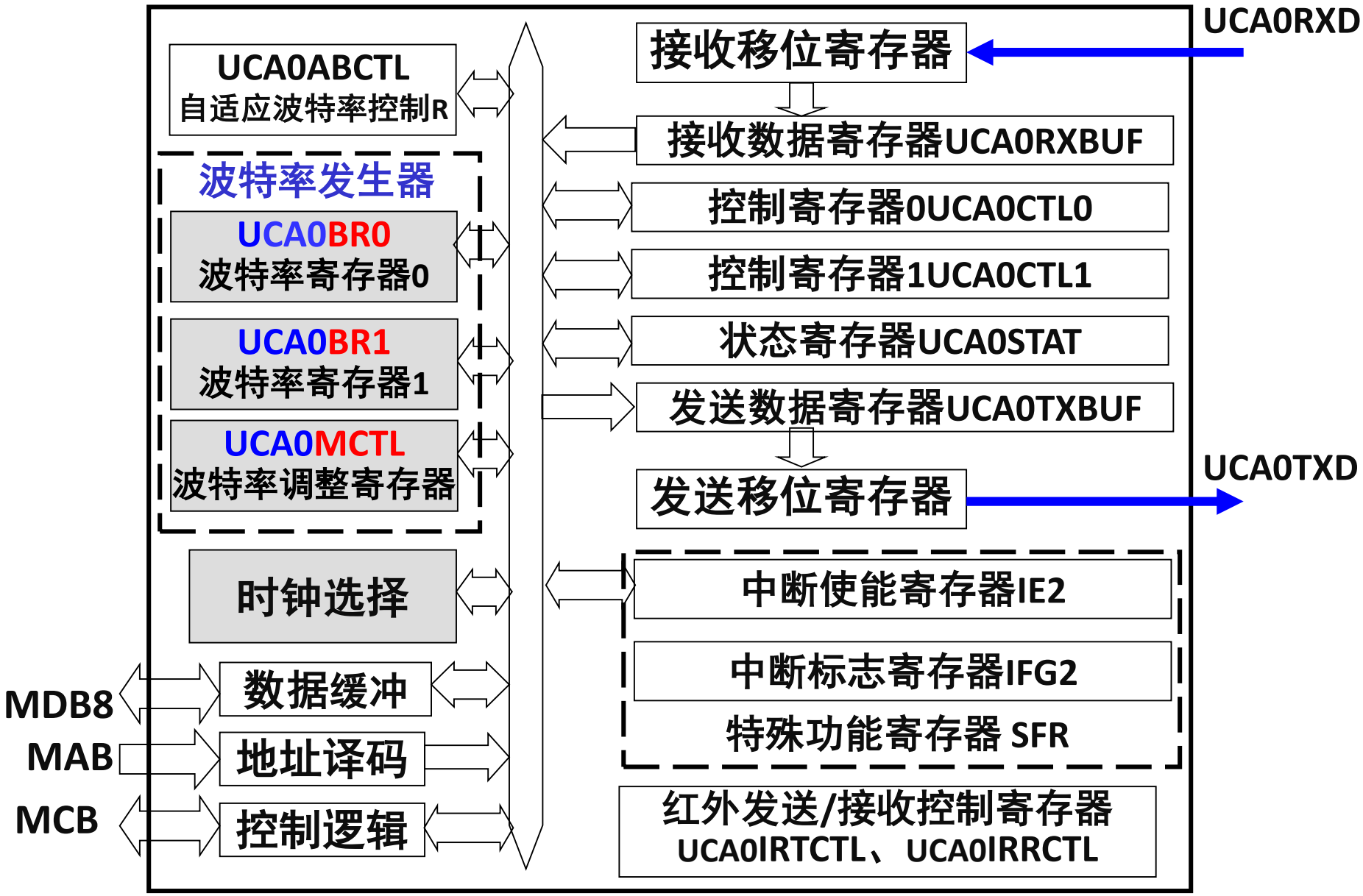
3）波特率设置

UCA0BR0 波特率寄存器0

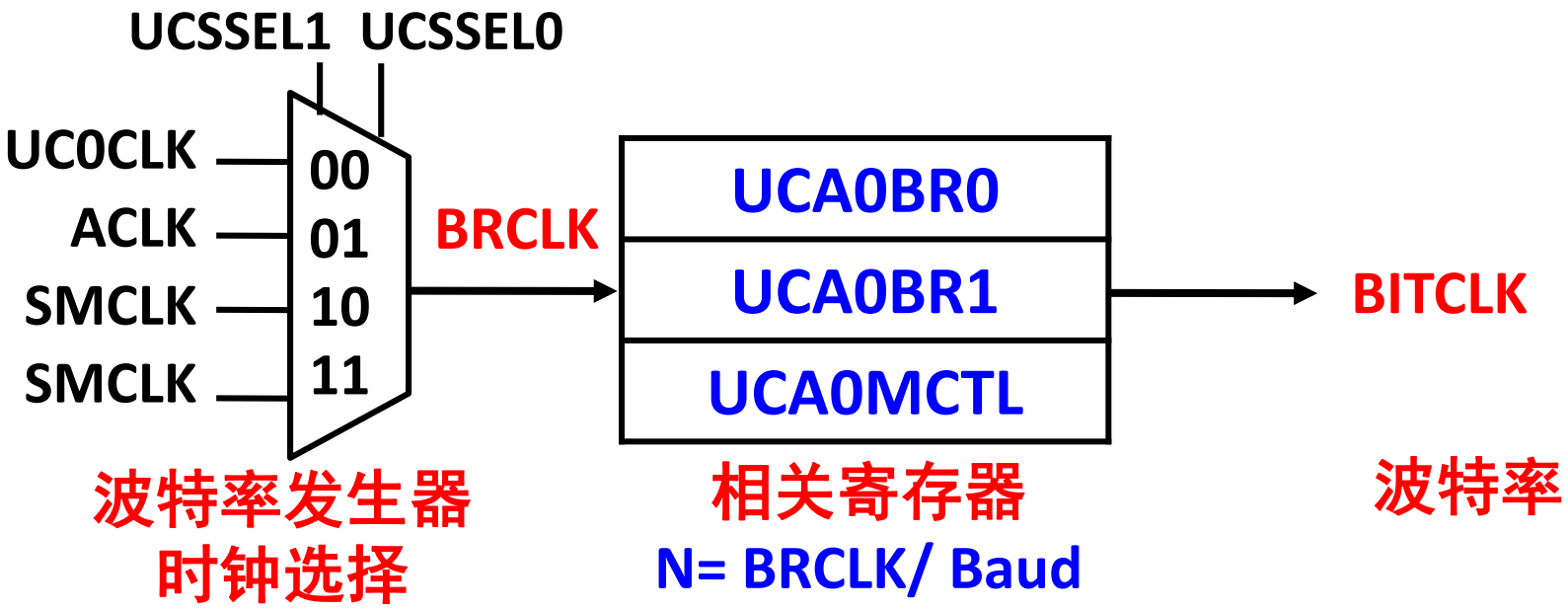
UCA0BR1 波特率寄存器1

UCA0MCTL 波特率调整寄存器

USCI 串口波特率发生器



波特率发生器编程结构示意图



波特率因子N = BRCLK/波特率，其中 3 ≤ N < 0xFFFF

如 SMCLK=1MHz，波特率= 9600， N=1000000/9600=104.167

如 SMCLK=32.768kHz，波特率= 9600， N=32768/9600=3.41

波特率发生器两种工作模式

- **低频率波特率模式（Low-Frequency Baud Rate Mode）**
  - ✓ 通常用于使用低频时钟源（如32.768KHz）产生波特率；
  - ✓ 通常在波特率因子N较小时采用（例如N<16时）
  - ✓ 可以最大可设的波特率是所选输入时钟源的1/3；
  - ✓ 使用UCA0BR1、UCA0BR0和 UCBRSx 设置波特率
  - ✓ 设置UCOS16 = 0 ,
- **过采样波特率模式（Oversampling Baud Rate Mode）**
  - ✓ 使用相对高频时钟源（≥16倍波特率）产生波特率
  - ✓ 采用此模式时，波特率因子需要大于16

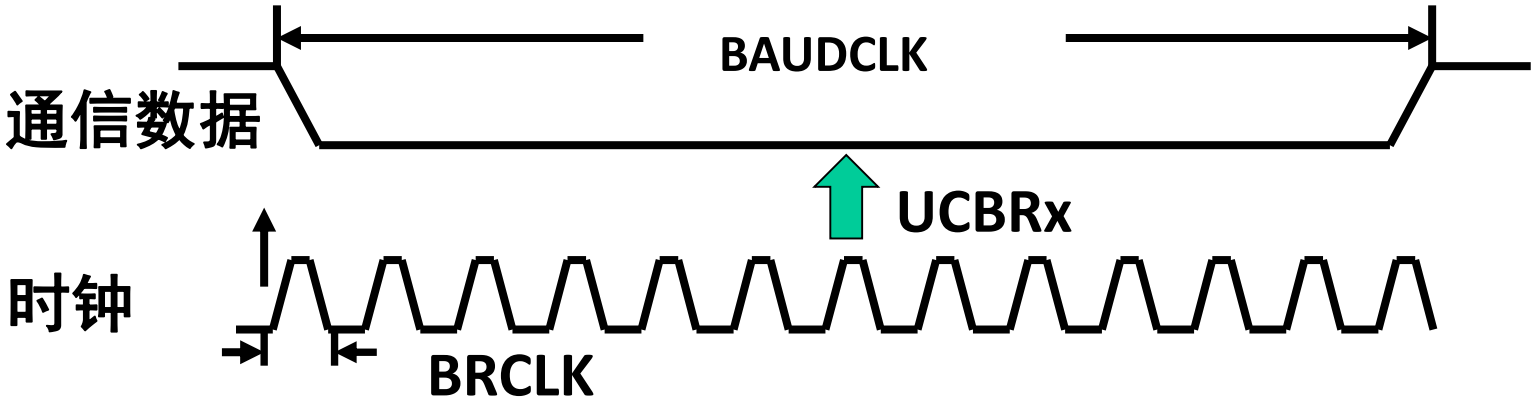
    例如：产生9600波特率，时钟源应大于 9600x16=153.6KHz
  - ✓ 使用UCA0BR1、UCA0BR0和 UCBRFx 设置波特率
  - ✓ 设置UCOS16 = 1

UCA0MCTL 波特率调整寄存器

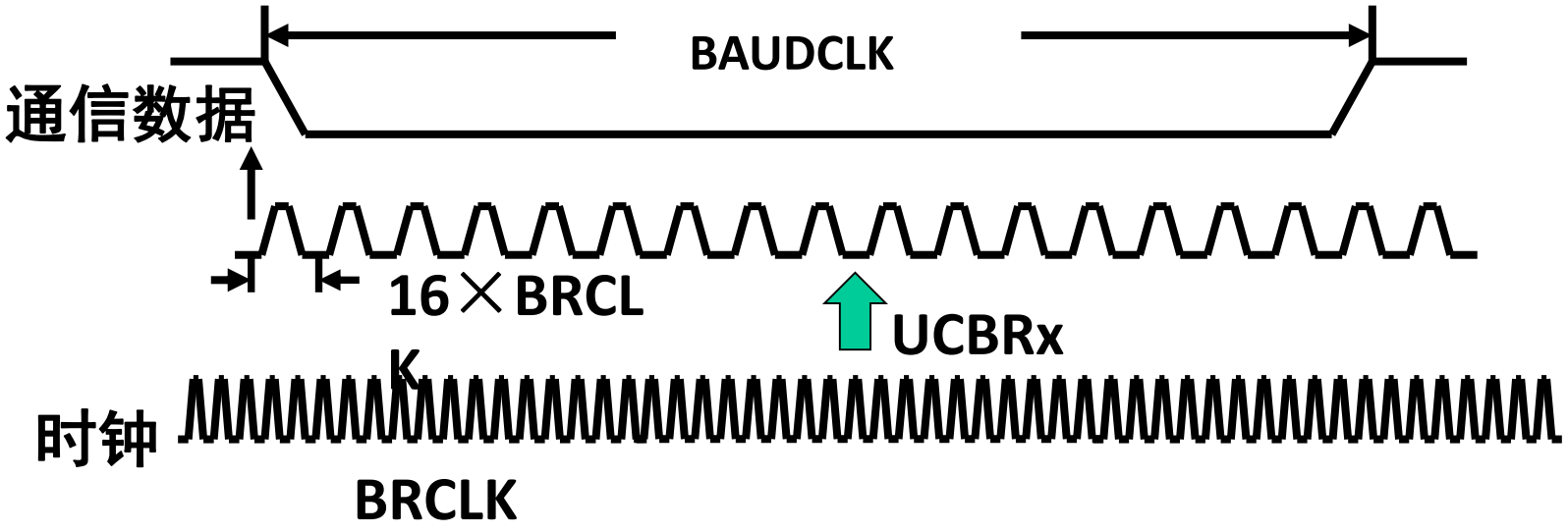
7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

86

- 低频率波特率模式（Low-Frequency Baud Rate Mode）



- 过采样波特率模式（Oversampling Baud Rate Mode）



UCA0BR1

USCI_A0波特率控制寄存器1

7	6	5	4	3	2	1	0
UCBRx							

UCA0BR0

USCI_A0波特率控制寄存器0

7	6	5	4	3	2	1	0
UCBRx							

- ✓ 用于存放波特率发生器的分频因子的整数部分
- ✓ UCA0BR1存放高8位， UCA0BR0存放低8位。

低频率波特率模式：$N = f_{BRCLK} / f_{Baud} = n.x$

过采样波特率模式：$K=N/16=m.y$

例： SMCLK=1MHz，波特率= 9600

低频率波特率模式时 N=1000000/9600=104.167

UCA0BR1=0　　UCA0BR0=104

过采样波特率模式时 K=N/16=104.167/16=6.51

UCA0BR1=0　　UCA0BR0=6

分频因子只用整数就可以了吗？

小数部分有什么影响？

小数部分怎么处理？

■ UCA0MCTL 波特率调整寄存器

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

■ **UCOS16:** 波特率控制模式选择

■ **UCBRFx, UCBRSx：** 波特率调整参数（由小数点位决定）

✓ UCBRFx：过采样波特率调整参数（0~15，仅在过采样波特率模式下使用）

✓ UCBRSx：低频波特率调整参数（0~7，仅在低频率波特率模式下使用）

对于给定的BRCLK时钟，所需的波特率决定了分频因子N：

$$N = \frac{f_{BRCLK}}{\text{Baud rate}}$$

■低频率波特率模式的设定方法

UCBRx = INT(N) //取整

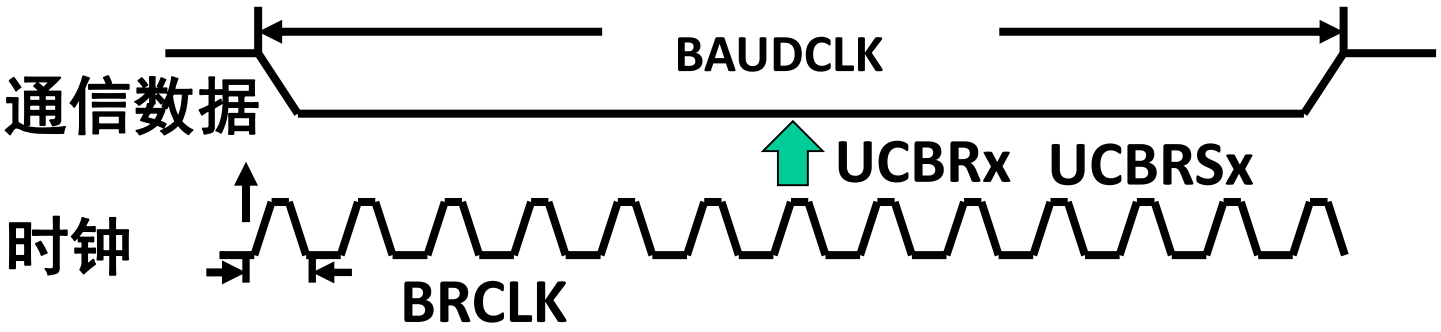
UCBRSx = round((N – INT(N)) × 8) //四舍五入取整

例如： N=1000000/9600=104.167

UCBRx=104， UCBRSx=0.167×8=1.33≈1

调整参数起什么作用？

$$\frac{1000000}{104}\times\frac{7}{8}+\frac{1000000}{105}\times\frac{1}{8}=\ ?$$ *9604*



对于给定的BRCLK时钟，所需的波特率决定了分频因子N：

$$N = \frac{f_{BRCLK}}{\text{Baud rate}}$$

■过采样波特率模式的设定方法

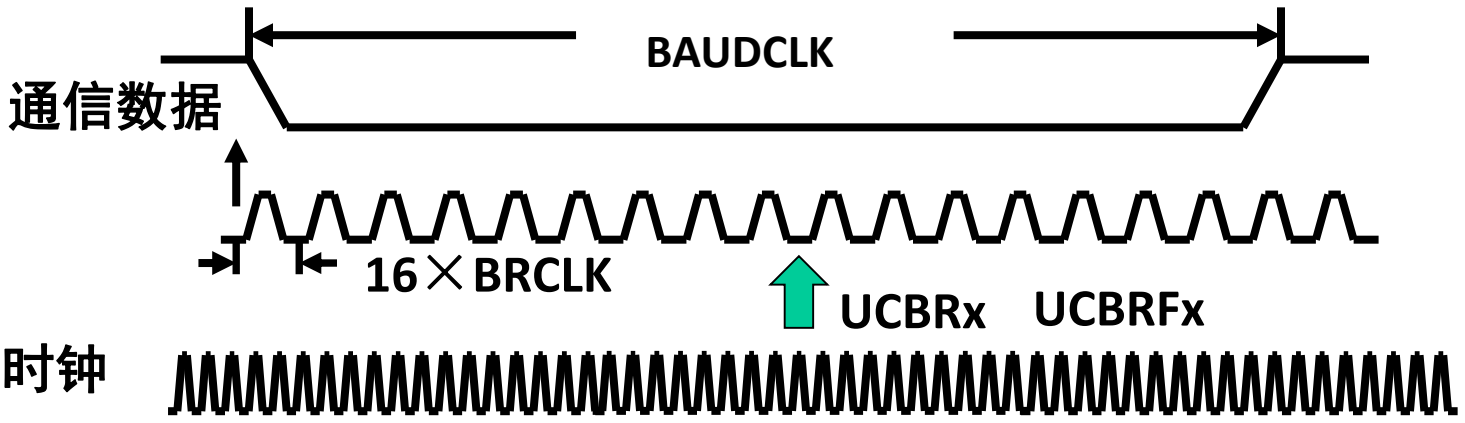
UCBRx = INT(N/16)　　//取整

UCBRFx = round((N/16 – INT(N/16)) × 16)　　//四舍五入取整

例如： K=N/16=1000000/9600/16=104.167/16=6.51

UCBRx=6　UCBRFx=0.51 ×16=8.16≈8　　*调整参数起什么作用？*

$$\frac{1000000/16}{6}\times\frac{8}{16}+\frac{1000000/16}{7}\times\frac{8}{16}=$$ ？ 9672



USCI_A0异步串行通信的波特率的设置

1. 根据选择时钟源和预设的波特率，计算分频因子N

N = BRCLK/Baud = n.x

其中BRCLK为输入波特率发生器的时钟频率

Baud为所需要的波特率

2. 根据N值，设置UCA0BR1、 UCA0BR0 、UCA0MCTL

1) 如果两者之比N < 16，可采用低频波特率模式

置UCOS16 = 0;

(UCA0BR1、UCA0BR0) = n; //取N的整数部分n

UCBRSx = 0.x×8； //N的小数部分0.x乘8, 四舍五入后取整

2) 如果两者之比 N ≥16，可采用过采样波特率模式

K = BRCLK/Baud/16 =N/16= m.y

置UCOS16 = 1;

(UCA0BR1、UCA0BR0) = m; //取K的整数部分=m

UCBRFx = 0.y×16; //K的小数部分0.y乘16, 四舍五入后取整

USCI_A0异步串行通信的波特率的设置例1

假设　时钟源选 SMCLK=32768Hz，波特率为 9600

计算：

N = BRCLK/Baud = 32768/9600=3.41 < 16

//只能采用低频波特率方式

n = 0x0003　　//取N的整数部分n

0.x×8= 0.41×8 ≈ 3　　//小数部分0.x乘8, 四舍五入后取整

所以：

```
UCA0BR1 = 0;
UCA0BR0 = 3;
UCA0MCTL = UCBRS_3;    //置UCBRSx=3,UCBRFx=0,UCOS16 = 0
```

UCA0MCTL

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

USCI_A0异步串行通信的波特率的设置例2

假设 时钟源选 SMCLK=1.0MHz， 波特率为 9600

采用低频率波特率方式：

计算:

N = BRCLK/Baud = 1000000/9600=104.167

(UCA0BR1、UCA0BR0)=104=0x0068

UCBRSx = 0.167×8 ≈ 1； UCOS16 = 0;

所以：

UCA0BR1 = 0;

UCA0BR0 =104; //等同于 UCA0BR0 =0x68;

UCA0MCTL = UCBRS_1; //置 UCBRSx=1,UCBRFx=0,UCOS16 = 0

UCA0MCTL

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

USCI_A0异步串行通信的波特率的设置例2(续)

假设　时钟源选 SMCLK=1.0MHz, 波特率为 9600

采用过采样波特率方式：

计算:

K= BRCLK/Baud /16 = 104.167/16 =6.51

(UCA0BR1, UCA0BR0)=0x0006;

UCBRFx=0.51×16 ≈ 8, UCBRS0, UCOS16 = 1.

所以：

UCA0BR1 = 0;

UCA0BR0 = 6;

UCA0MCTL = UCBRF_8 + UCOS16; //置 UCBRFx=8,UCBRSx=0,UCOS16 = 1

	7	6	5	4	3	2	1	0
UCA0MCTL	UCBRFx				UCBRSx			UCOS16
	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

USCI_A0异步串行通信的波特率的设置例3

假设　时钟源选 SMCLK=12.0MHz, 波特率为 9600

采用低频率波特率方式：

计算：

N = BRCLK/Baud = 12000000/9600=1250.0

(UCA0BR1, UCA0BR0) =1250=0x04E2

UCBRSx=0，　UCOS16 = 0;

所以；

UCA0BR1=0x04;

UCA0BR0=0xE2;

UCA0MCTL = 0;

UCA0MCTL

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

USCI_A0异步串行通信的波特率的设置例3(续)

假设　时钟源选 SMCLK=12.0MHz, 波特率为 9600

采用过采样波特率方式：

计算：

K = BRCLK/Baud/16 = 12000000/9600/16 =78.125

(UCA0BR1, UCA0BR0) =78

UCBRFx=0.125×16 =2; UCOS16 = 1;

所以；

UCA0BR1=0;

UCA0BR0=78;

UCA0MCTL = UCBRF_2 + UCOS16;

UCA0MCTL

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

USCI_A0异步串行通信的波特率的设置例4

假设　时钟源选 SMCLK=12.0MHz, 波特率为 115200

采用低频率波特率方式：

计算：

N = BRCLK/Baud = 12000000/115200=104.167

(UCA0BR1, UCA0BR0) =104

UCBRSx = 0.167×8 ≈ 1, UCOS16 = 0

所以：

UCA0BR1=0;

UCA0BR0=104;

UCA0MCTL = UCBRS_1

UCA0MCTL

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

USCI_A0异步串行通信的波特率的设置例4(续)

假设　时钟源选 SMCLK=12.0MHz, 波特率为 115200

采用过采样波特率方式：

计算：

K = BRCLK/Baud/16 = 12000000/115200/16 =6.51

(UCA0BR1, UCA0BR0)=0x0006

UCBRFx=0.51×16 ≈ 8, UCOS16 = 1

所以：

UCA0BR1=0;

UCA0BR0=6;

UCA0MCTL = UCBRF_8+ UCOS16;

UCA0MCTL

7	6	5	4	3	2	1	0
UCBRFx				UCBRSx			UCOS16
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

USCI_A0串行通信编程方法

4）数据收/发方式

数据收发寄存器： UCA0RXBUF 数据接收寄存器

UCA0TXBUF 数据发送寄存器

查询方式收发： IFG2 中断标志寄存器2

发送：数据发送寄存器空时可以发数据

接收：数据接收寄存器满时可以取数据

中断方式收发： IFG2 中断标志寄存器2

IE2 中断使能寄存器2

发送寄存器空/接收寄存器满时产生中断

■ UCA0TXBUF 发送数据寄存器

用于存放要发送的并行数据，

如检测到发送移位寄存器为空，

自动将保存在UCA0TXBUF中的数据传送到发送移位寄存器，

并通过UCA0TXD引脚按设定的异步串行数据帧格式发出.

例 编程将字节类型变量Tdata的内容通过UCA0TXD发送出去

```
UCA0TXBUF=Tdata;
```

■ UCA0RXBUF 接收数据寄存器

用于存放接收移位寄存器接收UCA0RXD引脚传送来的数据，

CPU通过对接收数据寄存器进行读操作，获取传入的数据.

例 编程将UCA0RXBUF接收的数据保存到字节类型变量Rdata中

```
Rdata=UCA0RXBUF;
```

■ IF2 中断标志寄存器2 (Interrupt Flag 2)

UCA0TXIFG：USCI_A0发送中断标志, 发送空时置位

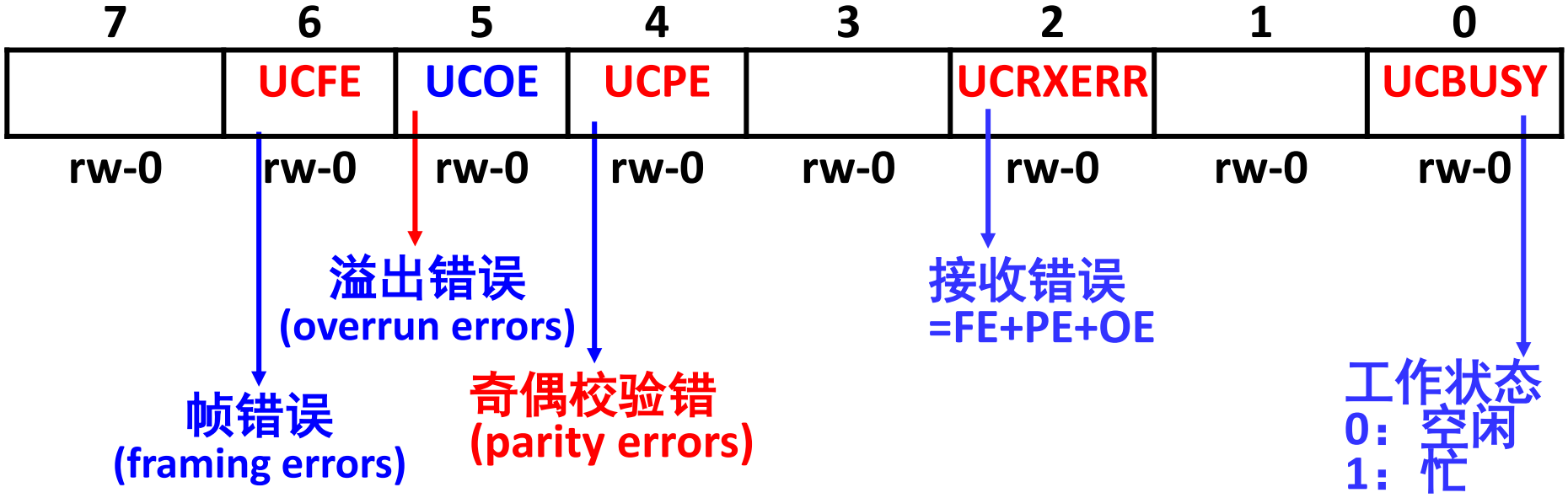
UCA0RXIFG：USCI_A0接收中断标志, 接收满时置位

■ IE2 中断允许寄存器2 (Interrupt enble 2)

UCA0TXIE：USCI_A0发送中断允许

UCA0RXIE：USCI_A0接收中断允许

■ UCA0STAT USCI_A0串口接收状态寄存器



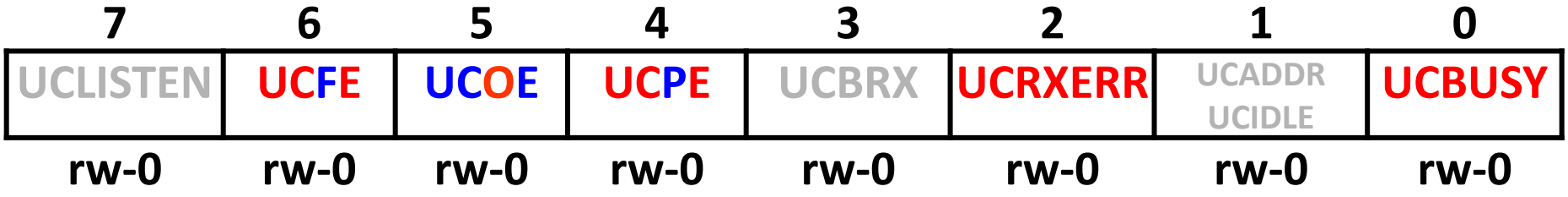
状态信息中 0: 表示正确 ； 1: 表示有错

读UxRXBUF操作将清零FE,PE,OE,RXERR等标志位
因此错误标志位的判断应在读UxRXBUF操作前进行。
错误标志位也可由软件清零。

msp430g2553.h中UCA0STAT 相关各个位的符号定义

```
#define  UCFE          (0x40)     /* USCI Frame Error Flag */
#define  UCOE          (0x20)     /* USCI Overrun Error Flag */
#define  UCPE          (0x10)     /* USCI Parity Error Flag */
#define  UCRXERR       (0x04)     /* USCI RX Error Flag */
#define  UCBUSY        (0x01)     /* USCI Busy Flag */
```

UCA0STAT



例 查询方式发送数据程序段

假设USCI_A0已经初始化，编程发送从string开始的26个字节

```
....
char    string[26]="Hello, this is MSP430G2553";
unsigned int  i;
...
for (i=0; i<26; i++)
{  while( (IFG2&UCA0TXIFG)==0 ){ };      //检测发送缓冲是否空
    UCA0TXBUF= string[i];                  //取一个字符发送
};
...
```

■ IE2中断标志寄存器2 (Interrupt Flag 2)

7	6	5	4	3	2	1	0
				UCB0TXIFG	UCB0RXIFG	UCA0TXIFG	UCA0RXIFG
				rw-1	rw-0	rw-1	rw-0

例 查询方式接收程序段

接收32个字节并存入buffer 开始的存储器中

假设USCI_A0已初始化，接收到的字符无论对错，都存入UCA0RXBUF中。

```
....
char    buffer[32];
unsigned  int  j;
...
for (j=0; j<32; j++)
{while( (IFG2&UCA0RXIFG) == 0 );    //检测接收缓冲器是否满
          buffer[j]= UCA0RXBUF;     //接收一个字符并保存
 };
 ...
```

■ IE2中断标志寄存器2 (Interrupt Flag 2)

7	6	5	4	3	2	1	0
				UCB0TXIFG	UCB0RXIFG	UCA0TXIFG	UCA0RXIFG
				rw-1	rw-0	rw-1	rw-0

三、利用USCI_A0异步串行方式通信

1. 查询方式

2. 中断方式(自学)

3. 简单通信协议举例

4. 基于异步串行接口控制的模块例

查询方式USCI_A0 初始化

1). 设置UCA0CTL1中 UCSWRST=1

2). 设置串行通信相关引脚

将相关引脚设置为USCI_A0模块功能；

3). 设置USCI_A0的2个控制寄存器和3个波特率寄存器

控制寄存器0：UCA0CTL0

控制寄存器1：UCA0CTL1

波特率控制寄存器: UCA0BR0，UCA0BR1，UCA0MCTL

4). 设置UCA0CTL1中 UCSWRST=0

允许USART运行

注： PUC(上电清零)时自动置位 SWRST=1

	7	6	5	4	3	2	1	0
UCA0CTL1	UCSSELx		UCRXEIE	UCBRKIE	UCDORM	UCTXADDR	UCTXBRK	UCSWRST

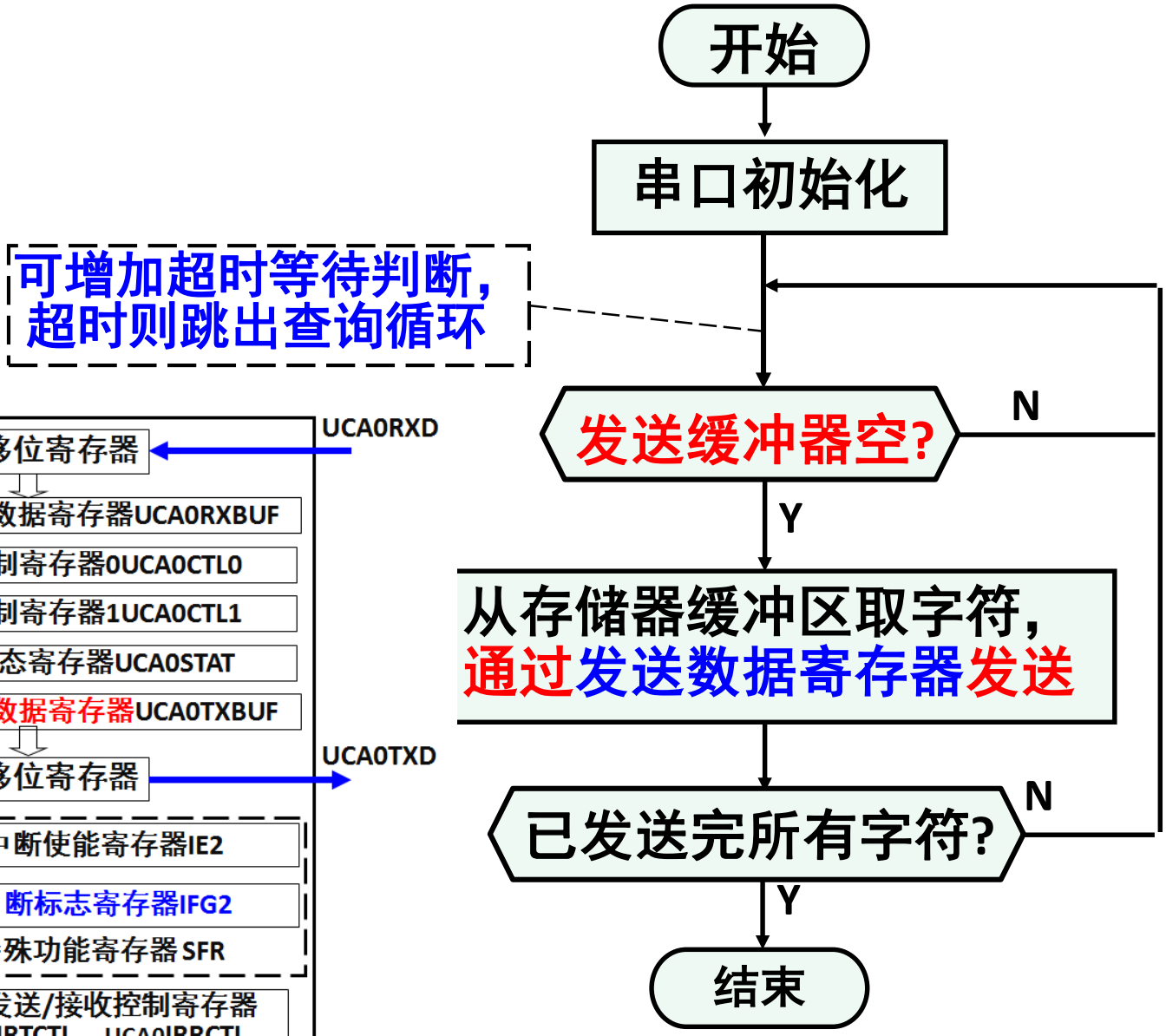
例 USCI_A0初始化子程序

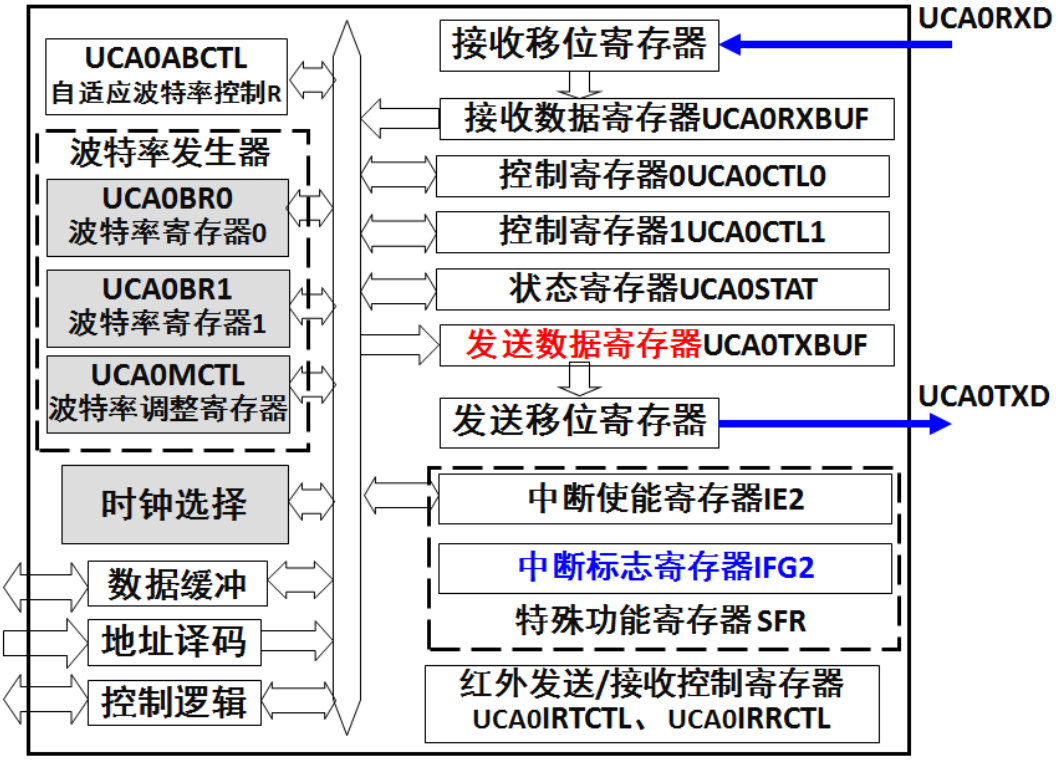
```
void  USCIA0_init( )
{     UCA0CTL1 |= UCSWRST;                    //swrst=1
      //置P1.1P1.2为USCI_A0的收/发引脚
      P1SEL |= BIT1+BIT2;          //P1.1串行接收，P1.2串行发送
      P1SEL2 |= BIT1+BIT2;
      //UCA0CTL0使用上电复位设置：
      //异步串行模式、设置数据格式数据8位、无校验、 1位停止位
      //时钟选择SCLK, 不论数据对均收:
      UCA0CTL1  | = UCSSEL1+UCRXEIE ;
      //假设SMCLK=1Mhz,设置波特率9600相关波特率寄存器:
      UCA0BR1=0;
      UCA0BR0=104;
      UCA0MCTL = UCBRF_0+ UCBRS_1;  //低频率波特率模式
      UCA0CTL1 &=~UCSWRST ;                   //swrst=0
}
```

UCA0CTL0寄存器

7	6	5	4	3	2	1	0
UCPEN	UCPAR	UCMSB	UC7BIT	UCSPB			UCSYNC
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

串口查询方式发送数据流程





7	6	5	4	3	2	1	0
				UCB0TXIFG	UCB0RXIFG	UCA0TXIFG	UCA0RXIFG

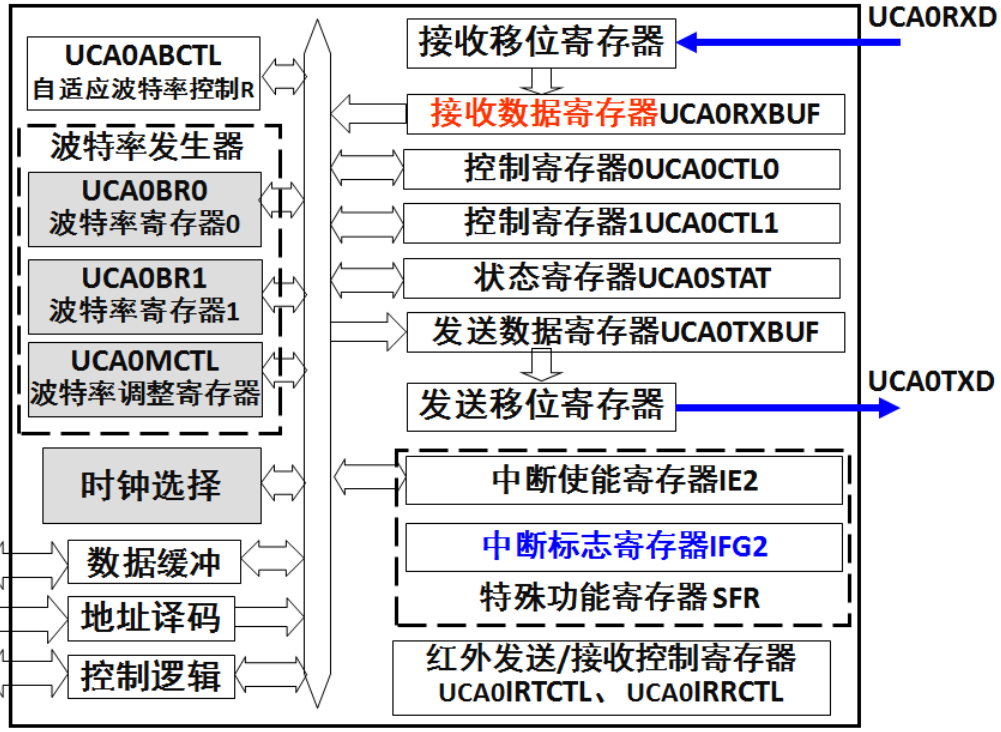
查询方式发送程序举例

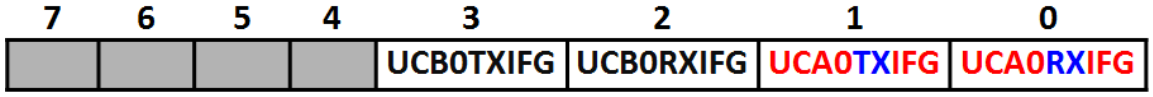
从USCIA0发送从string 开始的26字节

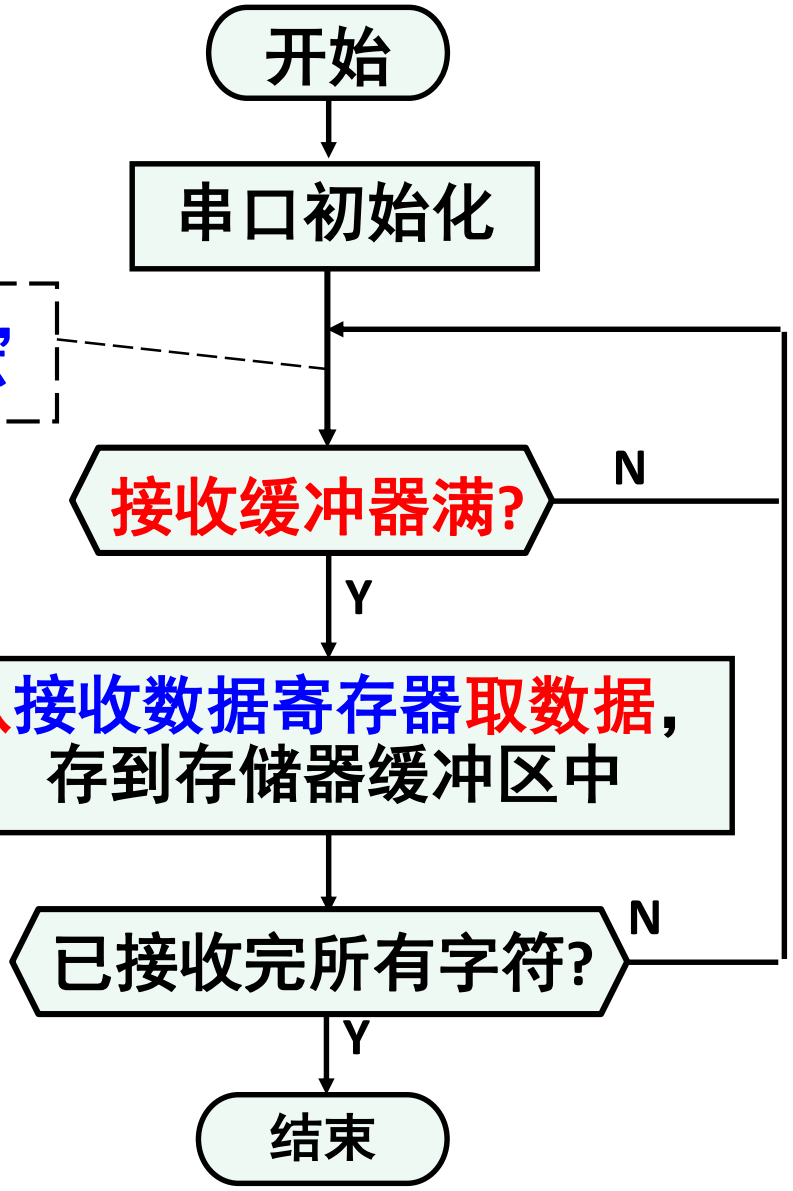
```
……
USCIA0_init();
…
for (i=0;i<26;i++)
{  while( (IFG2&UCA0TXIFG)==0 ){ };      //检测发送缓冲是否空
UCA0TXBUF= string[j];                    //取一个字符发送
};
…
```

注意： 当UCA0TXBUF中的数据传送到移位寄存器后，
UCA0TXIFG自动清0，不用软件清0

串口查询方式接收数据流程







查询方式接收程序举例

从USCIA0接收32字节存入buffer 开始的存储器中。
不考虑超时等待

```
    ……
    USCIA0_init();
    ……
for (j=0; j<32; j++)
 {while( (IFG2&UCA0RXIFG) == 0 ){ };   //检测接收缓冲器是否满
            string[j]= UCA0RXBUF;      //接收一个字符并保存
 };
  …
```

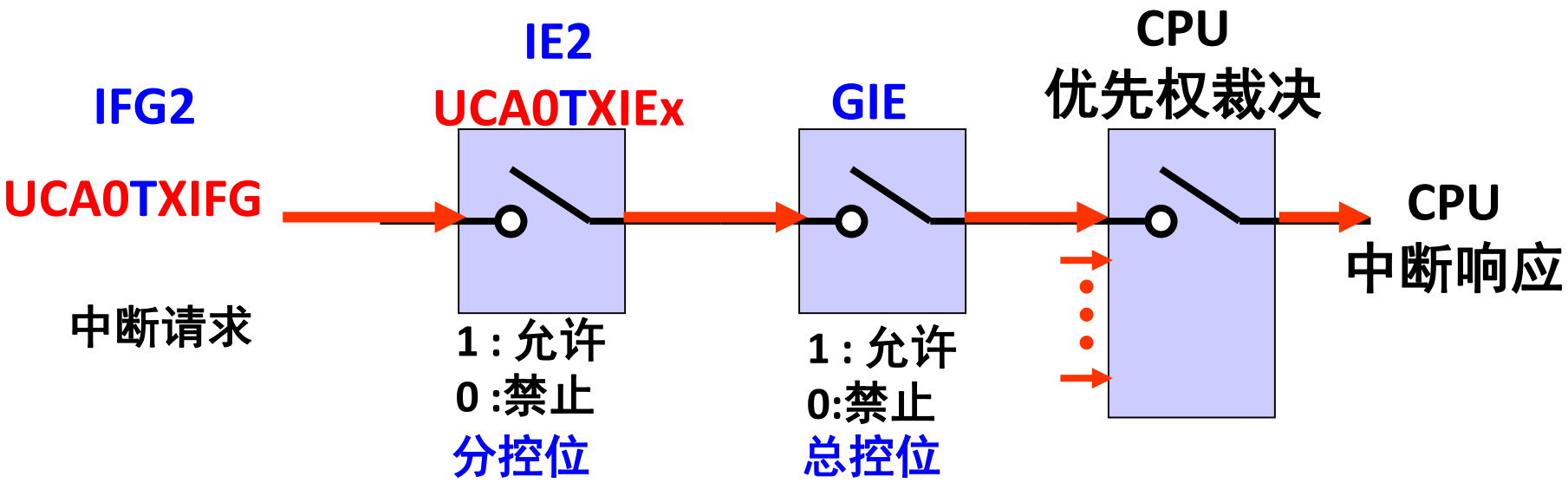
注意：从UCA0RXBUF读取数据后，UCA0RXIFG自动清0，
不用软件清0

2. 中 断 方 式

- 串口发送、接收中断控制
- 串口发送、接收中断向量
- 串口中断方式初始化
- 串口中断方式程序流程举例
- 串口中断方式程序清单例

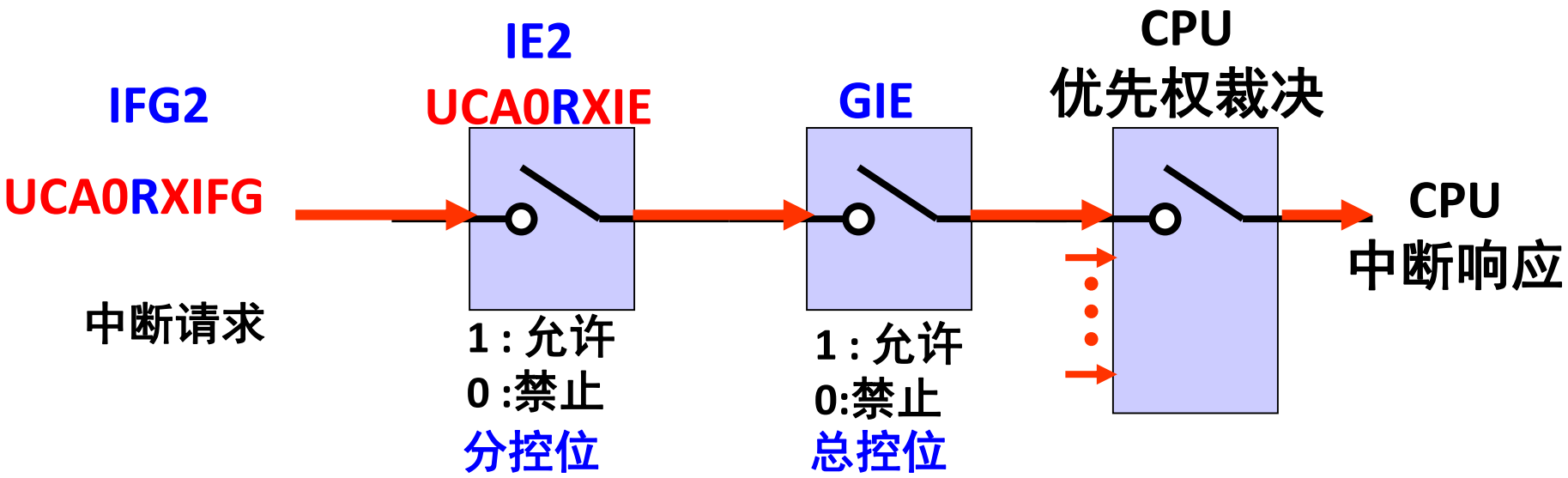
USART 发送中断控制

- 当被发送的字符从发送数据寄存器UCA0TXBUF移入发送移位寄存器,发送数据寄存器变空,置IFG2中的发送寄存器空标志UCA0TXIFG=1，发中断申请,相当于主动告诉CPU可以发送新数据了,如果IE2中的分中断允许位UCA0TXIE=1, 且CPU的 GIE=1,则CPU将响应该中断，转去执行相应类型的中断程序



USART 接收中断控制

- 当串行接口检测到引脚UCA0RXD上有异步串行格式数据，通过移位寄存器转化成并行数据存放到接收数据寄存器时，置IFG2中的接收寄存器满标志UCA0RXIFG=1，发中断申请，相当于主动告诉CPU有新数据到了，
  如果IE2中的分中断允许位UCA0RXIE=1，且CPU的 GIE=1，则CPU将响应该中断，转去执行相应类型的中断程序



MSP430串口通信接收和发送具有各自的中断向量

中断源	中断标志	向量地址	优先级,类型号
串行通信接口 A0/ B0	UCA0RXIFG, UCB0RXIFG	0xFFEE	7
串行通信接口 A0/ B0	UCA0TXIFG, UCB0TXIFG	0xFFEC	6

msp430G2553.h中的定义：

```
//Interrupt Vectors (offset from 0xFFE0)
......
#define USCIAB0TX_VECTOR    (6 * 2u)  /* 0xFFEC USCI A0/B0 Transmit */
#define USCIAB0RX_VECTOR    (7 * 2u)  /* 0xFFEE USCI A0/B0 Receive */
......
```

USCIA0异步串行中断方式 初始化

1. 设置UCA0CTL1中 UCSWRST=1
2. 设置串行通信相关引脚

   将相关引脚设置为USCI_A0模块功能；
3. 设置USCI_A0的2个控制寄存器和3个波特率寄存器

   控制寄存器0: UCA0CTL0

   控制寄存器: UCA0CTL1

   波特率控制寄存器: UCA0BR0，UCA0BR1, UCA0MCTL
4. 设置UCA0CTL1中 UCSWRST=0

   允许USART运行

同查询方式

5. 置收/发中断允许

   在中断允许寄存器IE2中，置UCA0RXIE=1或 UCA0TXIE=1
6. 开总中断允许

   置GIE=1

例： 编程设置允许USCI_A0 的接收中断

```
IE2 |= UCA0RXIE;
```

思考：如何打开USCI_A0的发送中断?

0: disable
1: enable

IE2中断标志寄存器2 (Interrupt Flag 2)

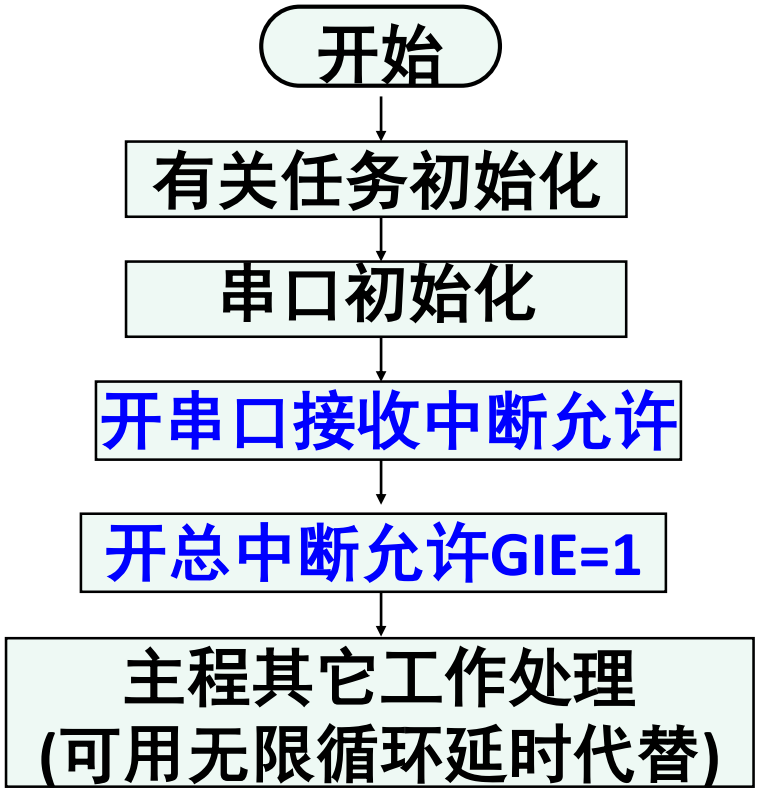
7	6	5	4	3	2	1	0
				UCB0TXIFG	UCB0RXIFG	UCA0TXIFG	UCA0RXIFG
				rw-1	rw-0	rw-1	rw-0

IE2中断允许寄存器2 (Interrupt enble 2)

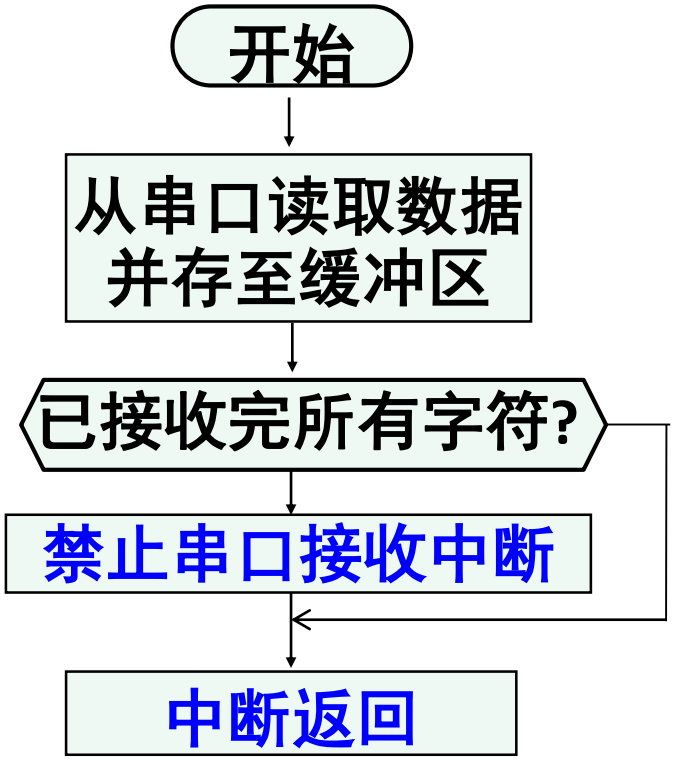
7	6	5	4	3	2	1	0
				UCB0TXIFG	UCB0RXIFG	UCA0TXIE	UCA0RXIE
				rw-0	rw-0	rw-0	rw-0

例 采用中断方式接收数据程序流程

1. 主程序



2. 中断子程



3. 设置中断向量

根据中断源在
中断向量表相应位置
设置中断向量

例 采用中断方式从USCIA0接收数据存到buffer开始的缓冲区

```
#include “msp430.h"
void  USCIA0_init( );
char     buffer[32];
 unsigned int  j;

int  main( void )
{   WDTCTL = WDTPW + WDTHOLD;
    Clock_int();                    //时钟初始化
    USCIA0_init(  );                //串口初始化
    IE2 |=UCA0RXIE;                 //开串口接收中断允许
    _EINT();                        //开总中断
    while(1);
}

#pragma  vector=USCIAB0RX_VECTOR
__interrupt  void  UCA0RX_isr( )
{   buffer[ j ]=UCA0RXBUF;              //将接收到的数据存至存储器
    j++;
    if (j==32) IE2 &= ~URXIE0;          //关闭串口0接收中断允许
}

void  USCIA0_init( ) { 略 }
void  Clock_int( )    { 略 }
```

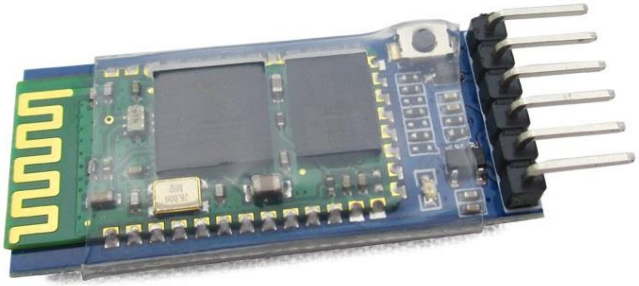
4、蓝牙串口模块的控制

蓝牙技术

- 一种短距离通信的无线电技术，一般在10m内,支持点对点及点对多点通信，工作在全球通用的2.4GHz ISM（即工业、科学、医学）频段。可实现全双工传输。
- 利用蓝牙技术，可简化移动通信终端设备之间的通信。
- 实现蓝牙通信应满足硬件和软件协议,即蓝牙技术联盟的标准。符合这个标准的设备才能以”蓝牙设备”的名义进入市场。
- 每个蓝牙设备均有设备名和设备地址
  设备地址是一个24位的二进制数，设备地址唯一，不可更改；
  设备名可更改。
- 蓝牙两种工作模式
  主模式：查询周围蓝牙从设备，并发起连接，
  建立主、从蓝牙设备间的透明数据传输通道。
  从模式：被动接收连接。

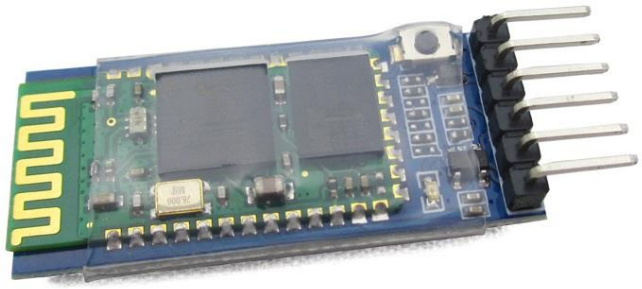
蓝牙串口通信模块

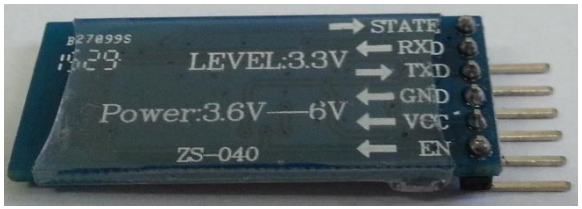
- 把符合蓝牙通信标准的软硬件封装成一个模块，利用一个异步串行通信接口，实现对蓝牙通信的控制；
- 蓝牙设备通信前，作为主模式的设备先选择要通信的蓝牙设备，进行连接，配对成功后，两个蓝牙设备之间方可通信；
- 蓝牙设备断开连接后，可与其他蓝牙设备配对进行通信。
- 数据传输过程中，蓝牙模块自动将来自串口发送信号线的数据，以无线方式发送出去；并把无线接收到的信息，传到串口的接收信号线上；
- 配对成功后，蓝牙模块部分可看成透明的，由串口进行控制。



HC-05蓝牙模块

HC-05蓝牙模块





- 引脚说明

  VCC, GND: 电源正、负极

  TXD:　　模块串口发送

  RXD:　　模块串口接收

  EN：　　AT命令控制端

  STATE: 蓝牙连接状态,

  　　未连接输出低，连接输出高

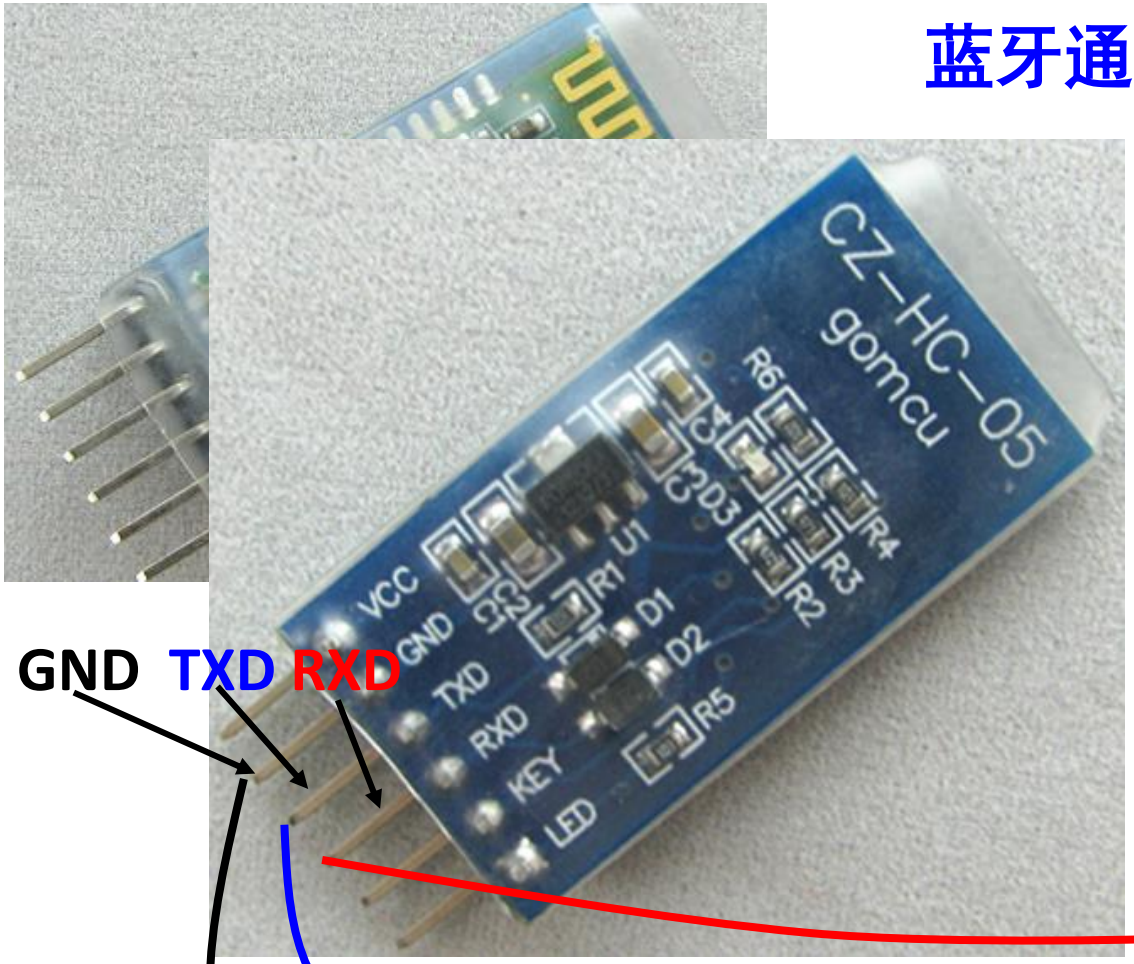
- led指示蓝牙模块工作状态

  快闪：没有蓝牙连接;

  双闪：蓝牙已连接并打开了端口;

  慢闪：进入AT模式;

蓝牙通信模块



GND TXD RXD

MSP430G2553

TXD

RXD

GND

实现无线通信功能

利用蓝牙技术建立无线通信