4.第二层——数据链路层

4.1 比特传输

SMBus在总线上分别使用固定的电压来表示逻辑 1 和逻辑 0。

4.1.1 . 数据有效性

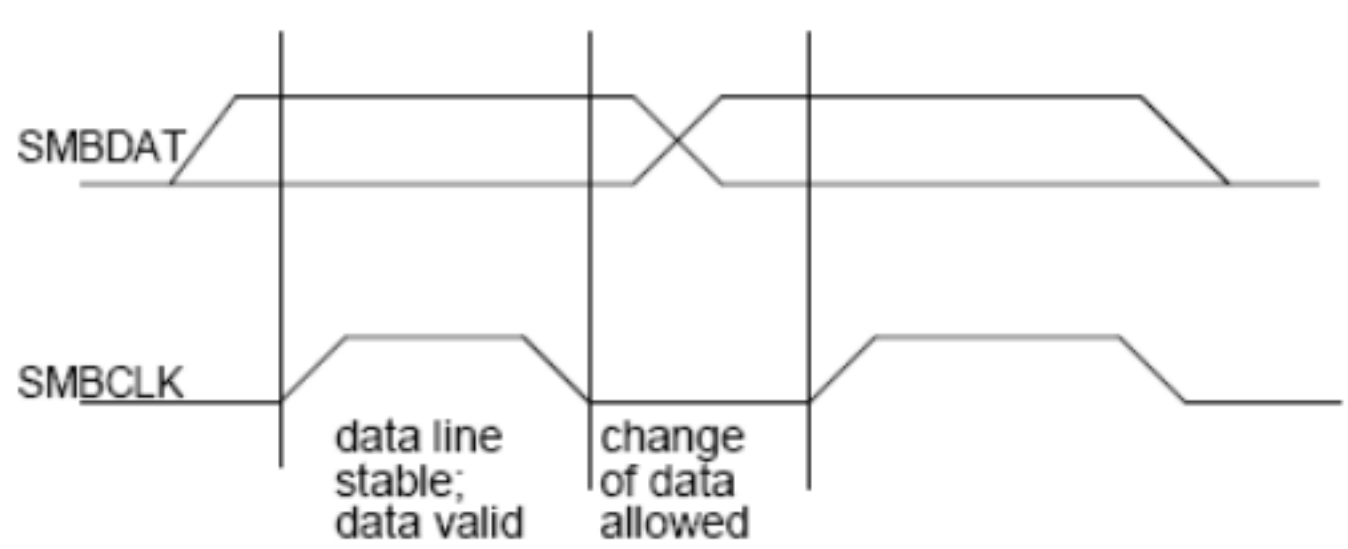

图 4-1 数据有效性

在时钟高电平阶段， SMBDAT上的数据必须保持稳定。当 SMBCLK为低时，数据才可以变化。图 4-1 表明了这种关系。实际的规范请参考图 3-1 和表 1。

4.1.2 开始和结束条件

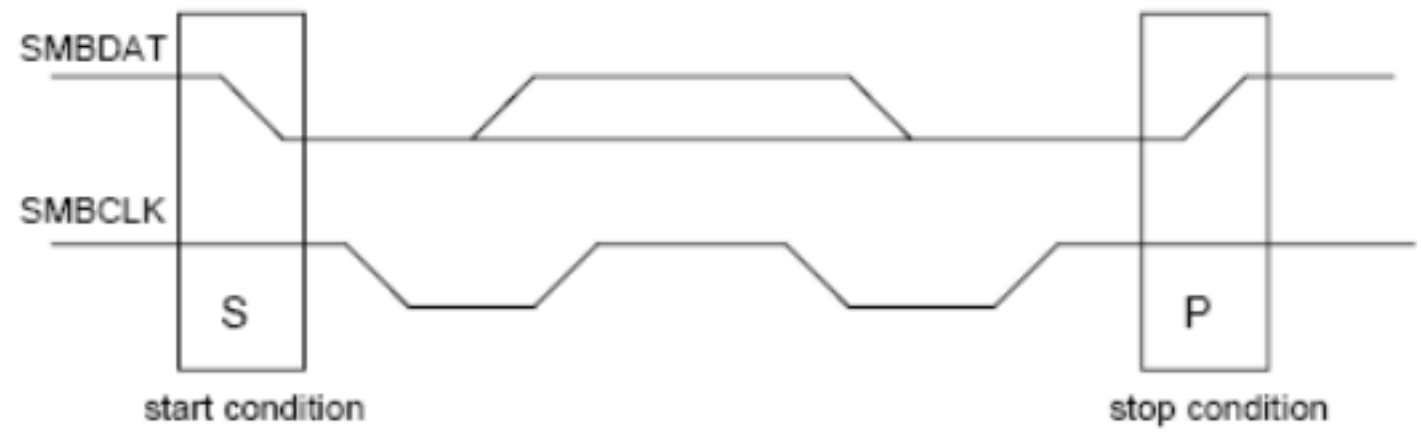

图 4-2 ：开始和结束条件

两种独特的总线状态定义了信息传输的开始和结束条件。

1. 在 SMBCLK为高时， SMBDAT由高到低的跳变表示信息传输的开始条件
2. 在 SMBCLK为高时， SMBDAT由低到高的跳变表示信息传输的开始条件。 开始和结束条件并不总是由总线主器件产生的。在开始条件产生后，总线就被认为处于忙碌状态。当出现结束条件或 SMBCLK和 XMBDAT两条线维持在高电平的时间超过 THIGH:MAX 后，总线处于闲置状态。

4.1.3. 总线闲置条件

总线闲置是一种在以下较小的那个时间内， SMBCLK和 XMBDAT两条线都处于高电平而没有任何状态变化的状态。

检测到上一个结束条件，经过 TBUF （4.7 μ S）以后，或者

$T_{HIGH:MAX}$ （50 μ S）

后一个时间参数考虑了如下的情况：一个主器件被动态地加到总线上，而且可能没有检测到 SMBCLK和 XMBDAT上状态的变化。这时，主器件必须等待足够长的时间来保证当前总线上没有数据传输。

4.2 SMBus 上的数据传输

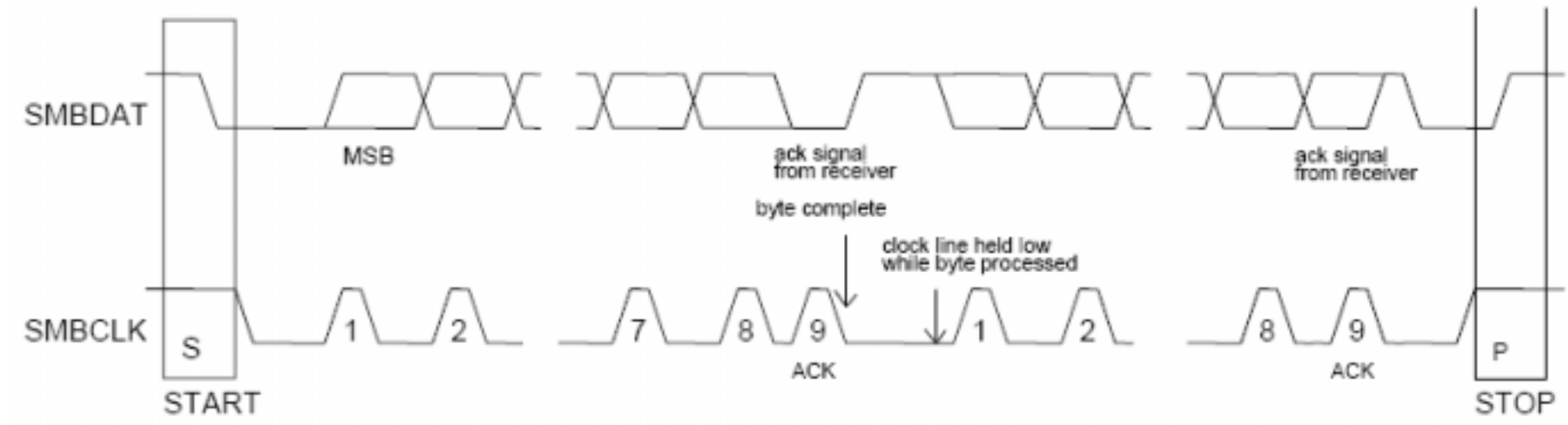

图 4-3 ：SMBus字节格式

每个字节包含八个比特。在总线上每个字节传输完都必须紧跟一个确认比特。字节在传输时都都先传送最高位（ MSB）

下面的图，图 4-4 表明了确认脉冲（ ZCK）和不确认脉冲（ NACK）相对于其他数据的位置。

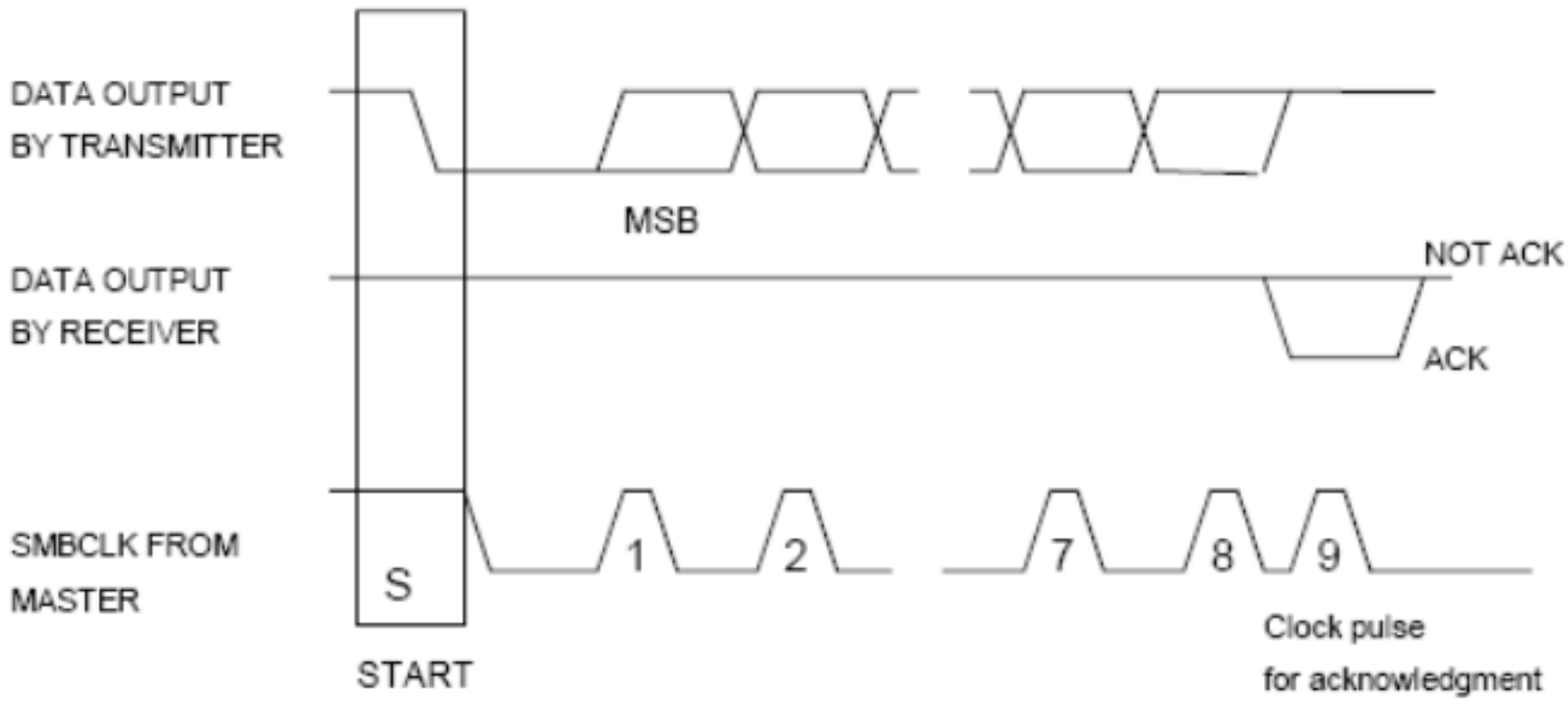

图 4-4 ：SMBus的 ACK和 NACK信号

由主器件产生这个与确认信号有关的时钟。发送端（也就是主器件或者从器件）确认这个时钟周期释放 SMBus数据线（ SMBDAT为高）。为了确认一个字节，接收端必须在时钟脉冲的高电平阶段，根绝 SMBus的时序规范，将 SMBDAT拉到低。

一个 SMBus器件必须始终对它自身地址做出确认。 SMBus利用这个信号来检测总线上可连接的设备是否存在。

一个 SMBus从器件在以下情况下可以不确认一个不是它自身地址的字节：

这个从器件正忙于执行一个实时任务，或者请求的数据不可用。主器件根据检测到的 NACK条件，必须产生一个停止条件来中断传输。注意还有另一种方法：从器件可以在这篇规范的限制内扩展时钟的低电平阶段，从而完成它的任务然后继续传输。

从器件检测到不合理的命令或不合理的数据。在这种情况下，从器件必须部确认这个接收到的字节。主器件根据检测到的 NACK条件必须产生一个停止条件，然后重新尝试通信。

如果一个主器件接收端正在通信中，它必须在从器件时钟下的最后一个字节产生一个

NACK，通过这种方法来像从器件示意数据传输的结束。从器件发送端必须释放数据线来允许主器件产生一个停止条件。

这最后一个机制是器件检测从器件发送端是否执行数据包错误检测的一种方法。具体内容参考 5.4 节的数据包错误检测。

4.3 时钟产生与仲裁

4.3.1. 同步

当不止一个主器件同时尝试在总线上放置时钟信号时会出现这种情况。结果总线上的信号是所有这些主器件产生的时钟信号的线与。

在仲裁过程中，为每个参与其中的主器件一比特一比特地对时钟有一个清晰的定义，这对总线的完整性很重要。 在 SMBCLK线上一个高到低的跳变将引起所有关联的期间开始计时它们低电平的长度并且如果这个器件是主器件， 它将驱使 SMBCLK为低。只要一个器件完成它对低电平的计时，它将释放 SMBCLK线。然而，如果其他主器件的低电平阶段较长的话将使 SMBCLK保持为低，导致实际 SMBCLK上的信号可能不变到高。在这种情况下，释放 SMBCLK的主器件将进入 SMBCLK高电平等待阶段。 当所有的器件都完成了对它们低电平的计时， SMBCLK线将被释放并变为高。此时，所有有关联的器件将开始计时它们的高电平阶段。第一个完成对它的高电平计时的器件将把 SMBCLK拉到低，然后下一个周期开始。

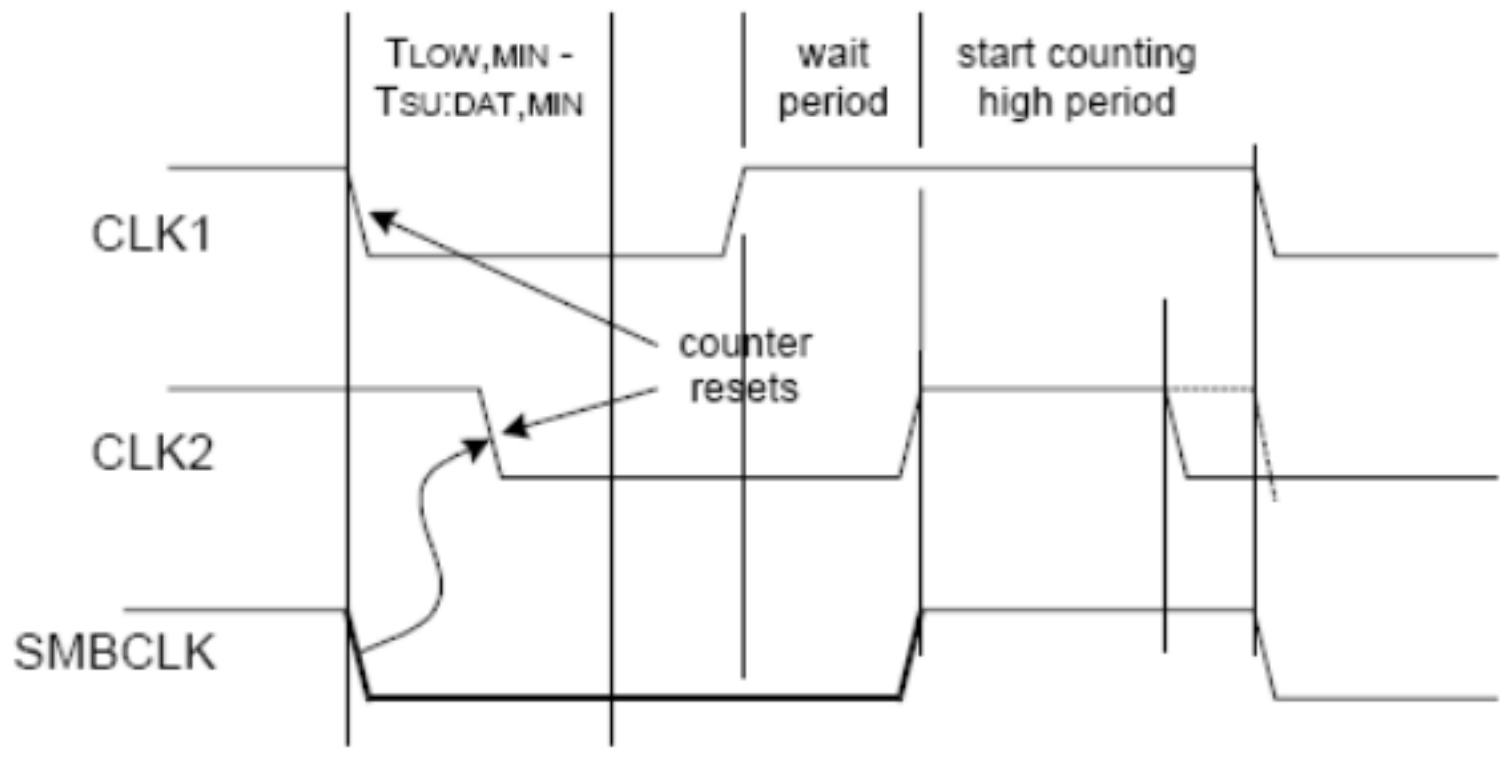

图 4-5 ：SMBus时钟同步

在图 4-5 中， CLK1和 CLK2的第一个下电平之间必须小于（ TLOWMIN-TSU:DAT ）。这样，所有器件就有了一个同步的时钟， SMBCLK低电平取决于最慢的那个器件，而 SMBCLK高电平取决于最快的器件。

4.3.2. 仲裁

仅当总线闲置时，主器件才可以开始传输数据。在最小保持时间 THOLD:STA 内可能有一个或多个器件产生开始条件，导致在总线上产生一个定义的开始条件。

因为当器件产生开始条件时可能并不知道其他主器件也在竞争总线。在 SMBCLK为高时，SMBDAT线上会发生仲裁。如果某个器件在 SMBDAT上传送高电平，而其他器件在 SMBDAT上传送低电平，则这个器件将在仲裁周期失去对总线的控制。

在仲裁中失去对总线控制的主器件可能会继续提供时钟脉冲，直到这个仲裁周期结束。在两个主器件尝试与同一个器件通信的情况下，在地址字节之后，冲裁将继续。在这种情况下，在传输剩余的数据时，冲裁依然存在。在两个主器件之间需要做出仲裁，其中一个要在总线上

放置重复开始信号，而另一个主器件要在总线上放置“ 0 ”数据比特时，前一个主器件必须认识到，它不能开始而将输掉仲裁。如果第一个主器件要在总线上放置一个重复开始信号，而第二个要早总线上放置 “ 1 ”数据比特， 第二个主器件会知道总线处于开始状态， 而它自身输掉仲裁。如果两个主器件都在总线上相同的比特位放置重复开始信号，在每个数据比特都将持续发生仲裁。

这种机制需要参与到仲裁周期的 SMBus主器件在仲裁期间监控 SMBDAT线的实际状态。

如果这个主器件同时具有从器件的功能并且在仲裁周期地址阶段失去对总线的控制，它必须检查放在总线上的实际地址， , 从而判断是否有另一个主器件在尝试与它通信。 在这种情况下，在仲裁中失败的主器件必须立刻切换到它的从器件接收端模式，以便接受到剩下的信息。

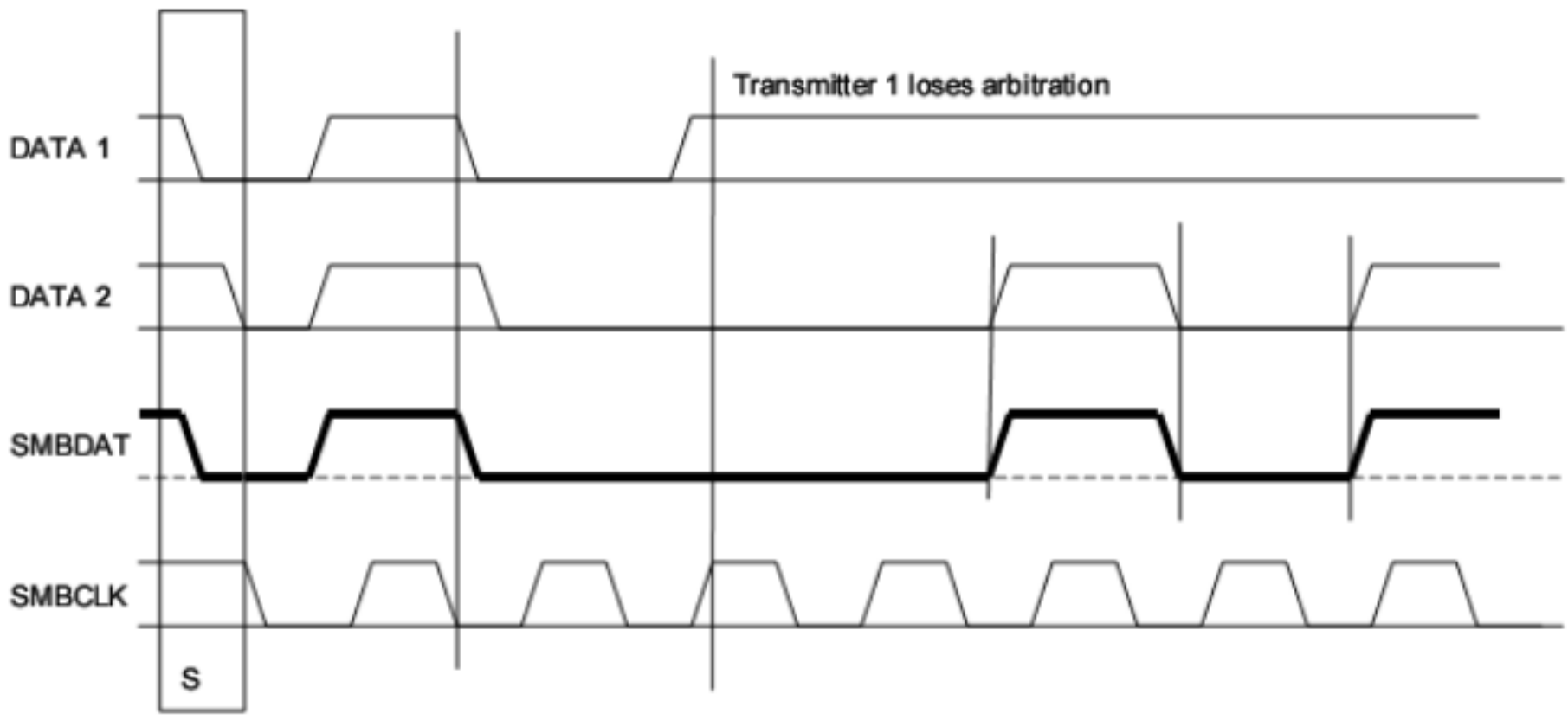

图 4-6 ： SMBus仲裁

在每一次总线通信时，主器件按需要能够识别总线上的重复开始条件。如果一个器件检测到一个不是它自己产生的重复开始条件，它必须退出传输。

一旦一个主器件赢得了仲裁，则在总线闲置之前，不再有仲裁。

4.3.3. 时钟低电平扩展

SMBus 利用时钟同步机制使得不同速度的器件可以在总线上共存。时钟同步机制除了用于总线仲裁过程中，还可以用于一比特或一字节的传输，以便让速度较慢的从器件与酥度较快的主器件配合。

在比特的层次上，器件可以通过周期性地扩展时钟的低电平阶段来减慢总线速度。

器件在传输大小范围为一个至这篇文档所描述的 AC规范的限制之间信息时， 它们可以延展时钟。然而，为了保持 SMBus 的带宽，这些可以周期性延展时钟的器件必须保持 10KHz 的 fSMB,MIN 频率。

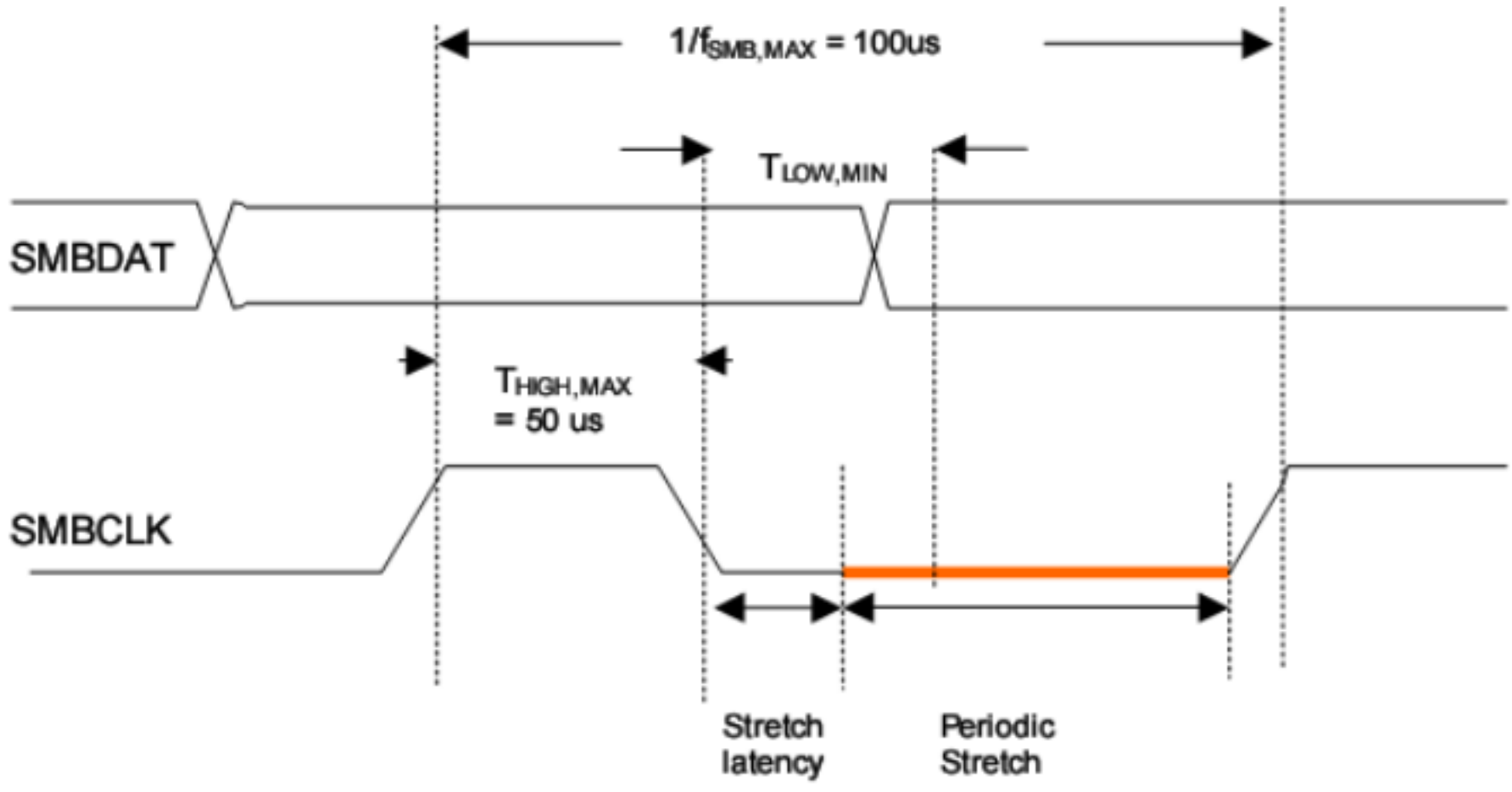

图 4-7 ：SMBus从器件的周期性时钟延展

如果需要时钟低电平扩展，必须在 SMBCLK下变平之后 TLOW:MIN-TSU:DAT时间段内开始。被设计成可以扩展时钟低电平的器件在每一个比特传输时必须保持总线最小频率 fSMB,MIN，也就是 10KHz。一个从器件为了进行一个实时任务或者检查一个字节的有效性，可能会选择在某个特定的比特传输时扩展时钟。在这种情况下，这个从器件必须满足 TTIMEOUT 和 TLOW:SEXT 的规范。时钟低电平扩展在任意比特传输时都会发生，包括比 ACK时钟脉冲优先级更高的时钟。

从器件可能会在总线上的字节传输之间扩展时钟低电平，以便处理接收到的数据或者准备待传输的数据。此时，从器件会在接收完数据或者对一个字节确认后将时钟保持为低。同样，从器件必须不违背 SMBus对于 TLOW:SEXT 的规范。

在总线工作期间，主器件同样可以选择扩展时钟低电平，不论是在在传输字节之间或者在传输字节的任一时刻，包括字节传输之后和确认时钟之前的时钟低电平阶段。为了处理出具或者完成一个实时任务，主器件可能需要选择性地扩展时钟低电平阶段。同样，这个主器件必须不违背 TLOW:SEXT 的规范。

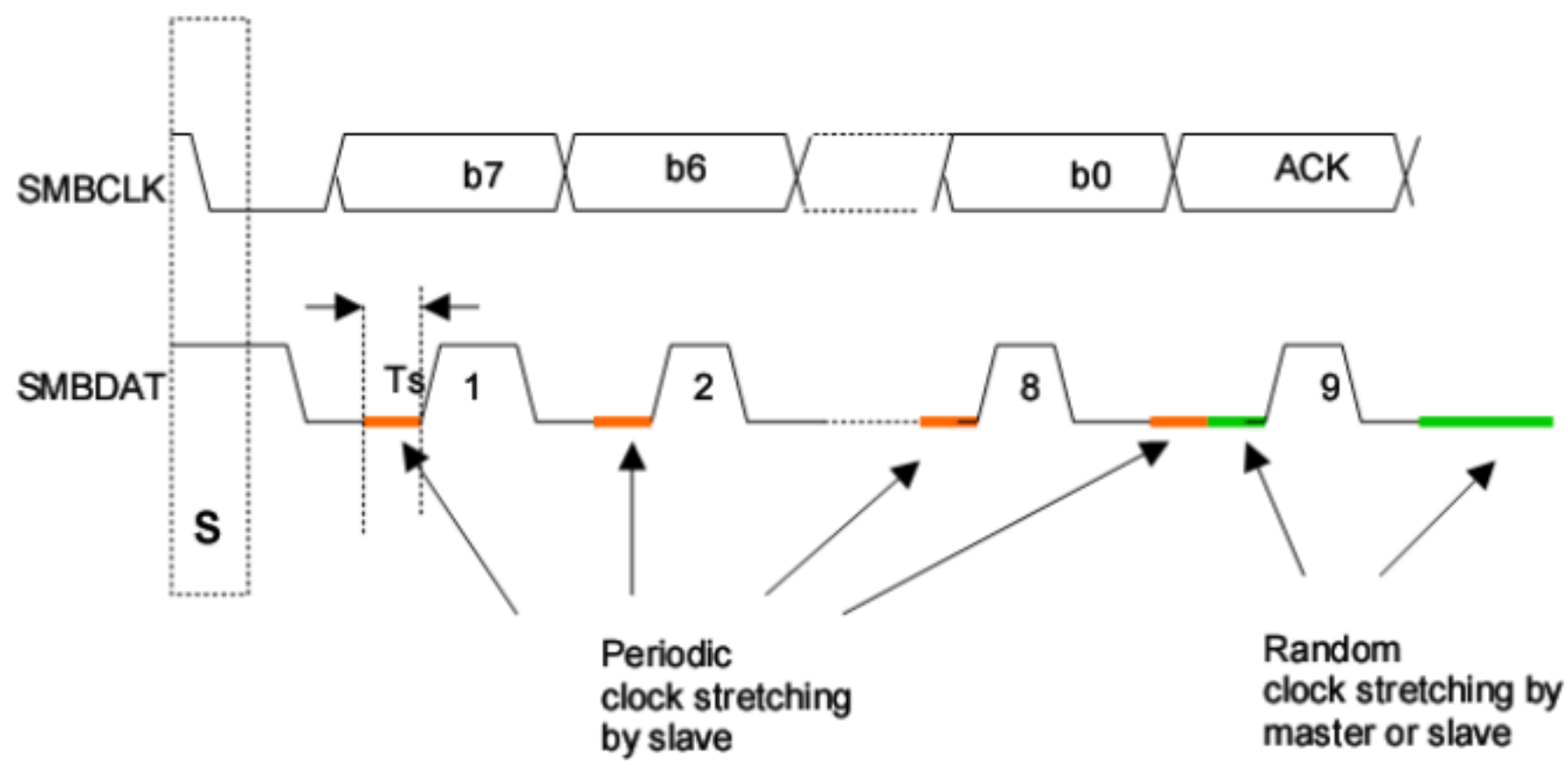

图 4-8 ：周期的和随机的时钟扩展

为了保持总线带宽以及可以从恶劣的总线条件中恢复，主器件和从器件都必须满足 SMBusTTIMEOUT 的规范。

4.4. 数据传输格式

SMBus数据传输满足如下图所示的格式。

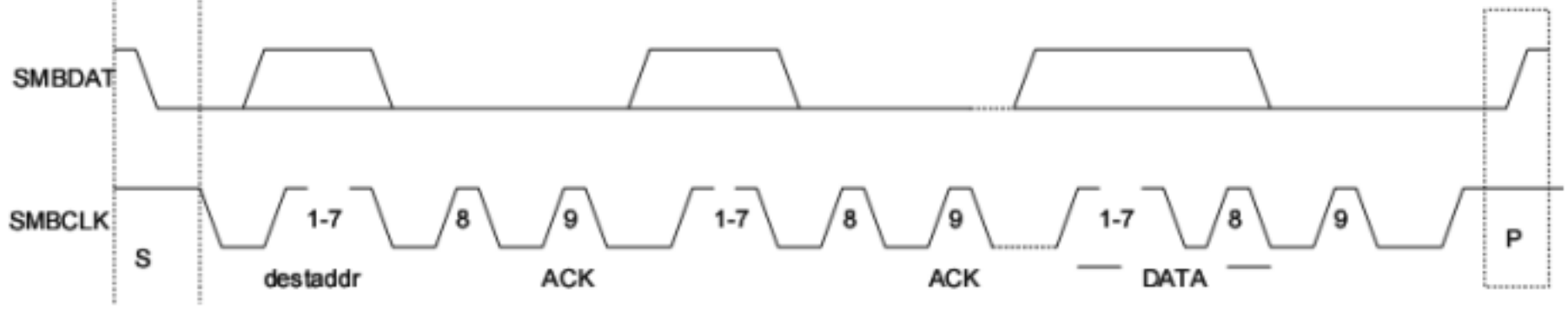

图 4-9 ：SMBus上的数据传输

在开始条件 S 之后，主器件在总线上放置它想与之通信的从器件的七比特地址。这个地址长度为七比特， 紧接着的第八比特 表示数据传输的方向 （R/W#）：一个 0 信号表示数据发送 （写）而 1 信号表示数据请求（读） 。一次数据传输总是由主器件产生的停止条件终止。

规范的 SMBus协议需要主器件产生一个重复开始条件，紧接着是从器件的地址，而不是首先产生一个停止条件。

5. 第三层——网络层

5.1. 用途模型

系统管理总线规范涉及到三种器件。从器件是接收数据或者对命令响应的器件。主器件是发送命令，产生时钟，终止传输的器件。主机是一个特殊用途的主器件，它提供与系统 CPU之间的主体接口。主机必须主从器件，而且必须支持 SMBus主机通知协议。在一个系统中最多有一个主机。一个无主机系统的例子是一个简单的电池充电器。这个充电器可能塞在墙上，等待为电池充电。

一个器件可能会被设计成始终是从器件而不会是主器件。一个器件大多数时候表现为从器件，但在一些特殊情况下，它可能变成主器件。同样，器件也能被设计成只是主器件。系统主机是一个大多数情况下表现为主机而偶尔作为从器件的例子。

5.2. 设备识别——从器件地址

任意一个在系统管理总线上作为从器件的设备都有一个独一无二的地址， 叫做从器件地址。作为参考，下面的地址是保留的，除非是这篇规范中详细讲到的其他情况，不然它们不可被 SMBus从器件使用或者分配给从器件。

Slave Address Bits 7-1	R/W# bit Bit 0	Comment
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Address reserved for different bus format
0000 011	X	Reserved for future use
0000 1XX	X	Reserved for future use
0101 000	X	Reserved for ACCESS.bus host
0110 111	X	Reserved for ACCESS.bus default address
1111 0XX	X	10-bit slave addressing
1111 1XX	X	Reserved for future use
0001 000	X	SMBus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

表 4：保留的 SMBus地址

所有其他其他地址都可以被动态地址设备以及 / 或混合设备用来分配地址。 混合设备地址将在 5.2.1.4 中讨论。

SMBus警报响应地址（ 0001 100 ）等价于设备的主器件性能。详见附录。

SMBus设备默认地址是被 SMBus地址解决协议保留的，它允许动态地分配地址。详细内容请看 5.6 节。

这里没有说明的，以及在附录中的地址都是留待将来使用的。所有的十比特从器件地址都被留作将来使用，因而不再此规范讨论的范围之内。

主机的地址长度最短，所以发送给主机的信息对于总线仲裁有最高的优先级。

5.2.1 SMBus 地址类型

在实际系统中， 几个 SMBus设备可以同时使用。 在设备地址竞争的情况下， 设计者可在 SMBus设备中使用可编程特征来解决这些竞争，或者使用在同一个系统中的多个 SMBus分支来分散这些使用相同地址的设备。

5.2.1.1. 保留地址

正如在表 4 中定义的， SMBus、 Access 总线以及 I^2C都为特殊的总线功能保留若干地址。

5.2.1.2. 目标分配地址

这些地址是由 SMBus工作组分配，用来区别设备的类型。分配了地址的每一种器件类型都必须有一个与之相关的 SMBus规范。一些使用 SMBus的系统认为假如一个设备是在一个功能分配地址上，那么它服从这个地址相关的规范。比如，智能电池中 SMBus的应用假定，智能电池设备以及控制器都在它们的功能分配地址上。这样，那些不满足智能电池设备的功能分配地址规范的器件不可以在智能电池应用中被使用。

其他 SMBus应用不仅仅依赖设备地址去识别设备的功能。在这些应用中，设备可能使用与功能分配地址重复的地址。

器件制造商。总之，除非希望设备满足相应地址的规范，否则应当避免功能分配地址。设备生产商应当咨询 SMBusWG来获得关于功能分配地址的最新信息， 以此作为判断是否他们的地

址分配在某些 SMBus应用中不允许的依据。

5.2.1.3. 原型地址

Slave Address	Description
1001 0XX	Prototype Addresses

表 5：原型地址

原型地址（ 1001 0XX）是为那些利用功能分配地址的试验中的设备保留的。它们不可以被分配到任一个器件。

5.2.1.4. 混合设备地址

为了特殊的系统目的，生产商已经生产了并且可能继续生产兼容 SMBus的设备，这些设备不需要执行所有的 SMBus规范，或者他们不需要从操作系统得到直接的支持。这些设备可能是端口扩展器， D/A 电路等等。 SMBus 网站 (http://www.smbus.org) 有一页，在这页面上， SBS-IF 成员公司可以分享这些设备的地址信息。这个网页帮助生产商避免可能在同一个 SMBus上共存的器件之间发生地址冲突。

5.2.1.5. 动态分配地址

2.0 版本引进了动态分配地址的概念。这将在 5.6 节里详细讨论。

5.3. 使用设备

一个典型的 SMBus设备有一系列命令，通过这些命令数据可以被读或者被写。所有的命令长度都为八比特 （一字节）。命令以及返回参数的长度不是固定的。 访问一个不存在或者系统不支持的命令可能导致错误。为了与这篇规范一致，最高位首先被传送。

对任意一个设备，我们有十一个可能的指令协议。从器件可能使用这十一个协议中的任意一个或者所有去通信。协议都是快指令，发送字节，接收字节，写字节，写字，读字节，读字，程序调用，块读，块写以及块读 - 块写程序调用。

5.4. 数据包错误检测

SMBus1.1 版本引进了数据包错误检测机制来提升可靠性以及通信的鲁棒性。 SMBus设备执行数据包错误检测对于 SMBus设备是可选的，但是，当且仅当在 ARP过程中，它是必须的。执行数据包错误检测的 SMBus设备必须能够与不执行数据包错误检测机制的主机以及其他设备通信。

数据包错误检测在执行时是在每个信息传输结束是天界数据包错误码 （PEC）。每个协议（除了后面一节中描述的快指令主机通知协议）有两种类型：一种有数据包错误码（ PEC）字节，另一种没有。 PEC是一个计算所有信息字节（包括地址以及读写比特）之后得到的 CRC-8 错误校验字节。 PEC是由提供最后一个数据字节的设备添加在信息的尾部。

5.4.1. 执行数据包错误检测

SMBus 必须能够兼容支持以及不支持数据包错误检测的设备的任意组合。表现为从器件并且支持 PEC的设备必须始终准备好作为从器件进行有 PEC或者没有 PEC的数据传输， 如果有 PEC 的话验证它的正确性，并且仅当 PEC 正确时处理信息。当 PEC 存在但是错误的时候需要执行 NACK。

5.4.1.1. ACK/NACK 以及数据包错误检测

在 SMBus字节中 ACK/NACK比特和其他 SMBus数据包里的任一比特一样易受破坏。 因此，在PEC之后的 ACK并不保证 PEC是正确的。一个主器件发送端在写结束时接收 ACK尽管很有可能表示这次写数据在从器件接收端被成功执行，但也并不必然是这样。

主器件发送端在 PEC之后接收到的 NACK信号表明从器件接收端的数据链路层及时发现数据传输的错误， 并在 PEC字节之后提供一个 NACK信号。这可能是因为 PEC错误或者任意的其他错误。如果在 NACK出现时， 设备能够足够快地发现并指出错误， 数据链路层上的错误也是可以由NACK信号指出的。

主器件发送端接收到 ACK信号表示从器件接收端没有及时确定数据链路层上的错误，所以没有提供 NACK。这可能是因为接收端不能实时地检查 PEC的有效性。

如果主器件发送端希望肯定一次写操作在目标设备上被成功执行，它必须使用一些更高层的机制来证明。这可能是采取带有 PEC读数据的形式、接收一个正确的 PEC可以可靠地表示原始的写操作没有发生错误。

当一个主器件接收端从一个从器件发送端读取数据时，由主器件接收端在通信结束时提供的 ACK/NACK信号除了标志最后一字节的结束外没有任何意义。 从器件发送端提供数据， 如果从器件发送端提供的 PEC是正确的，主器件接收端可能认为数据与发送端发送的一样。如果 PEC是错误的，主器件接收端将采取合适的补救措施。

5.4.1.2. 主器件执行

主器件在任意的通信中都可以使用 PEC。主器件需要预先知道目标从器件是否支持 PEC，或者要有判断目标从器件是否支持 PEC的方法。 PEC的使用是通过下面这些控制的： 上层协议（比如设备驱动）， SMBus规范（比如 SMBusARP协议的要求）或者给定一类器件的检测算法（比如智能电池）。

5.4.1.3. 从器件执行

执行数据包错误检测的从器件必须准备好接收、发送那些有或者没有 PEC的数据。在从器件接收端数据传输期间，设备被识别之后，协议和指令必须接收并且检查这个附加的 PEC，以便检查有效性。

在从器件接收端数据传输期间，从器件接收端必须在最后一个字节传输完后对另外的时钟做出响应，并且向请求数据的主器件接收端提供一个 PEC。