# $I^2C$ Bus 与 SMBus 间技术差异

前言：

关于$I^2C$与SMBus，许多人很少去谈论与了解两者的细节差异，包括很多国外的简报，文章也经常将两者混写、交杂描述、交替运用。

确实，在一般运用下，$I^2C$ Bus与SMBus没有太大的差别，从实际接线上看也几乎无差异，甚至两者直接相连多半也能相安无误地正确互通并运作。不过若真要仔细探究，其实还是有诸多不同，如果电子设计工程师不能明辨两者的真实差异，那么在日后的开发设计的验证纠错阶段必然会产生困扰，为此本文将从各层面来说明$I^2C$ Bus与SMBus的细微区别，期望能为各位带来些许帮助。

我写这篇文章，可以理解为郭長祐先生博客中相关文章的读书笔记，我可没有那么高的造诣，关于$I^2C$ Bus的基础，可参考先生之前的「I2C界面之线路实务」，网址为：

http://www.digitimes.com.tw/n/article.asp?id=304799064272FED148256FDC00481D68

当然也可以去参考Philips半导体网站的$I^2C$官方规格：

http://www.semiconductors.philips.com/acrobat/literature/9398/39340011.pdf

运用背景、版本演进之别

首先从规格的制订背景开始，$I^2C$是在设计电视应用时所研发的界面，首版于1992年发表；而SMBus（System Management Bus）则是Intel与Duracell（金顶电池）共同制订笔记本电脑所用的智能型电池（Smart Battery）时所研发的接口，首版于1995年发表，不过SMBus文件中也提及，SMBus确实是参考自$I^2C$，并以$I^2C$为基础所衍生成。

$I^2C$起源于电视设计，但之后朝通用路线发展，各种电子设计都有机会用到$I^2C$；而SMBus则在之后为PC所制订的先进组态与电源管理接口（Advanced Configuration & Power Interface；ACPI）规范中成为基础的管理讯息传递接口、控制传递接口。

虽然$I^2C$与SMBus先后制订时间不同，但都在2000年左右进入成熟化改版，$I^2C$的过程改版以加速为主要诉求，而SMBus以更切合Smart Battery及ACPI的需求为多。

$I^2C$三次主要改版：

1992年 v1.0

1998年 v2.0

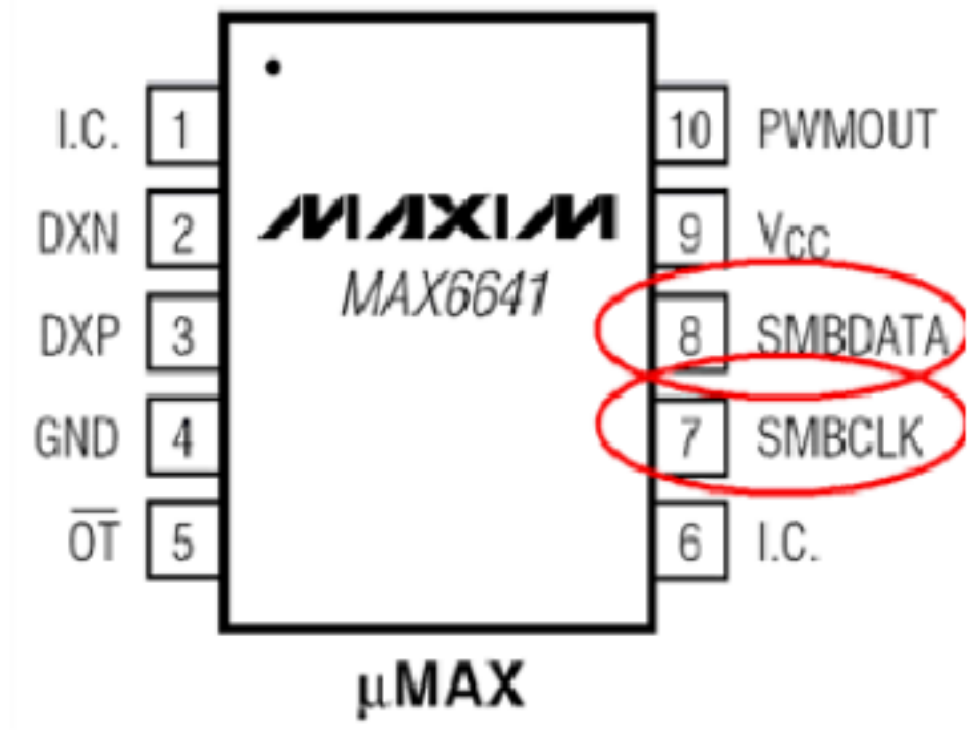
2000年 v2.1

SMBus三次主要改版：

1995年 v1.0

1998年 v1.1

2000年 v2.0




▲图例：MAXIM 公司的 MAX6641 芯片，具有温度监督及风扇控制功能（用 PWM 脉宽调变方式控制风扇转速），图中脚位 7、8即是 SMBus （圈处），其他装置可透过 SMBus 与此芯片沟通，取得温度及相关信息，或进行命令操控。（图／ MAXIM-IC.com ）

电气特性差异：逻辑电平定义、限流、相关限制

$I^2C$的Hi/Lo逻辑电平有两种认定法：相对认定与绝对认定，相对认定是依据 Vdd 的电压来决定，Hi为0.7Vdd，Lo为0.3Vdd，绝对认定则与 TTL 准位认定相同，直接指定 Hi/Li 电压，Hi为3.0V，Lo为1.5V。相对的 SMBus 只有绝对认定，且电平与 $I^2C$有异，Hi为2.1V，Lo为0.8V，与 $I^2C$不全然吻合但也算部分交集。

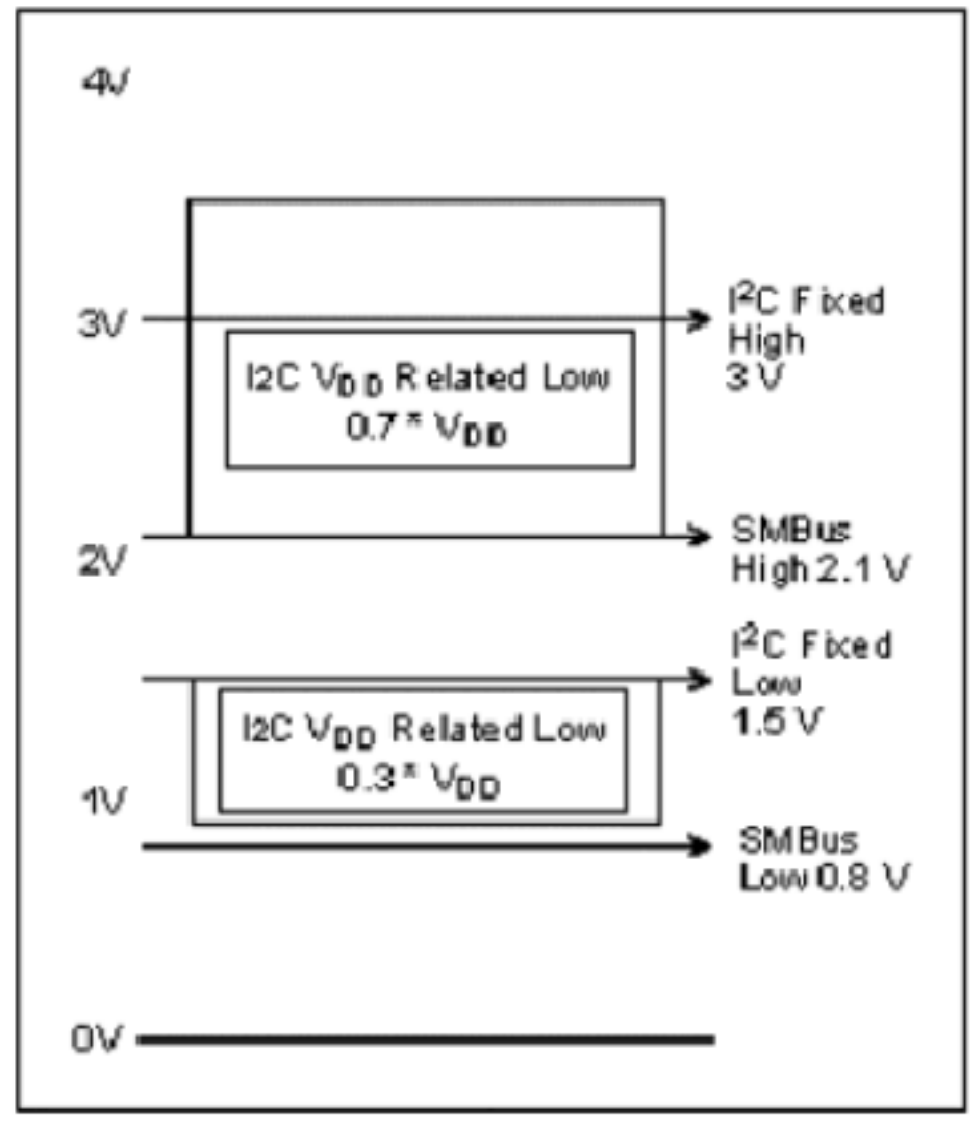
不过，SMBus 后来也增订一套更低电压的电平认定，Hi为1.4V，Lo为0.6V，这是为了让运用 SMBus 的装置能更省成本的作法。

了解电压后再来是电流，由于 SMBus 一开始就是运用在笔记本电脑内，所以省电的表现优于 $I^2C$，只需 100uA 就能维持工作，$I^2C$却要到 3mA 同样的低用电特性也反应在漏电流（ Leakage Current ）的要求上，$I^2C$最大的漏电流为 10uA，SMBus 为1uA，但是 1uA 似乎过度严苛，使运用 SMBus 的装置在验证测试时耗费过多的成本与心力，因此之后的 SMBus 1.1 版放宽了漏电流上限，最高可至 5uA。

再者是相关限制，$I^2C$有线路电容的限制，SMBus 却没有，但也有相类似的配套规范，即是电平下拉时的电流限制，当 SMBus 的集电极开路 Pin 导通而使线路接地时，流经接地的电流不能高于 350uA，另上电流（即相同的集电极开路 Pin 开路时）也一样有规范，最小不低于 100uA，最高也是不破 350uA 的。

既然对电流有限制，那么也可容易地推断对上拉电阻的阻值之范围要求，$I^2C$在5V Vdd 时当大于 1.6kohm，在 3V Vdd 时当大于 1kohm，类似的 SMBus 于5V Vdd 时当大于 14kohm，3V Vdd 时当大于 8.5kohm，不过这个定义并非牢不可破，就一般实务而言，在 SMBus 上也可用 2.4k？ 3.9kohm 范畴的阻值。

附注：$I^2C$的时钟线称 SCK 或SCL，数据线称 SDA。SMBus 的时钟线称 SMBCLK，数据线称 SMBDAT 。




▲图说： I²C与 SMBus 在逻辑位准的电压定义不尽相同，基本上 I²C的定义较为宽裕、弹性，而 SMBus 则更专注在省电方面的要求。（图／ MAXIM-IC.com ）

时序差别与考验

物理层面的空间要求完后，再来就是物理层面的时间，即是时序（ Timing ）方面的差别。

先以运作频率来说， I²C此方面相当宽裕，最低频可至 0Hz（直流状态，等于时间暂停），高可至 100kHz （Standard Mode ）、400kHz （Fast Mode ）、乃至 3.4MHz（High Speed Mode ），相对的 SMBus 就很局限，最慢不慢于 10kHz ，最快不快于 100kHz 。很明显的， I²C与SMBus 的交集运作频率即是 10kHz？100kHz 间。

用于笔记本电脑的电池管理或 PC组态管理、用电管理的 SMBus ，很容易体会不需要更高运作频率的理由， 只要传递小数据量的监督信息、 控制指令本就不用过于高速，而朝向广泛运用的 I²C自然希望用更高的传输以应对各种可能的需求。然而大家可能会疑惑，为何 SMBus 有最低速的要求？何不放宽到与 I²C相同的无最低速限制呢？

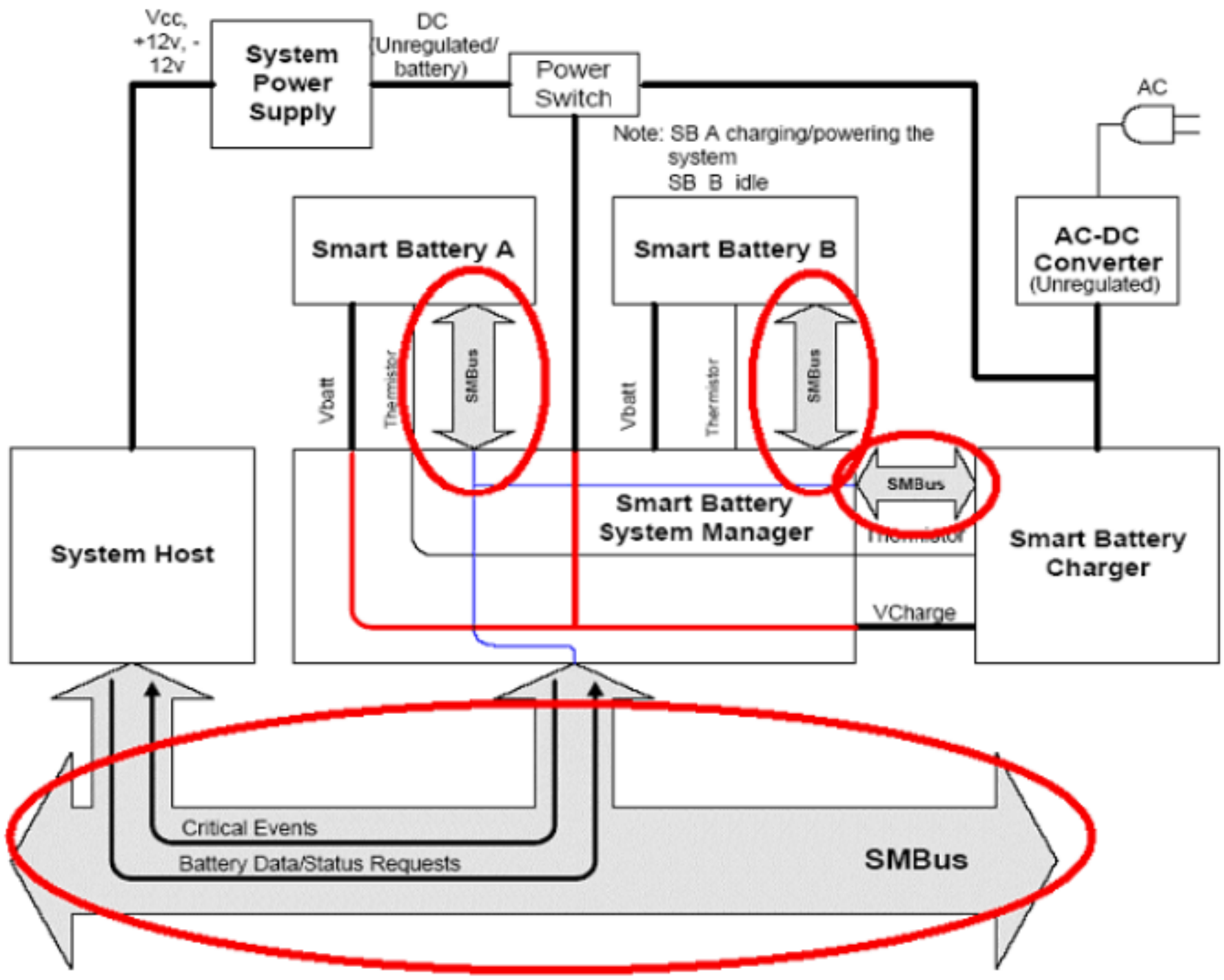
SMBus 一定要维持 10kHz以上的运作频率， 主要也是为了管理监控， 另一个用意是只要在保持一定传速运作的情况下加入参数， 就可轻松获知总线目前是否处于闲置（ Idle）中，省去逐一侦测传输过程中的停断（ STOP）信号，或持续保有停断侦测并辅以额外参数侦测，如此对总线闲置后的再取用会更有效快速。

传速要求之后还有数据保持时间（ Data Hold Time ）的要求， SMBus 规定 SMBCLK 线路的电平下降后， SMBDAT 上的数据必须持续保留 300nS ，但I²C 却没有对此有相同的强制要求。

类似的， SMBus 对接口被重置（ Reset）后的恢复时间（ Timeout ）也有要求，一般而言是 35mS ，I²C这方面亦无约束， 可以任意延长时间。 相同的 SMBus 也要求无论是在主控端（ Master ）或受控端（ Slave），其频率处于 Lo电平时的最长持续时间不得超越限制，以免因为长时间处在 Lo准位，而致收发两端时序

脱轨（失去同步，造成后续误动作）。

还有，$I^2C$与SMBus 在信号的上升时间、下降时间等也有不同的细节要求，此点必要时也必须进行确认，或在验证过程中稍加留意。




▲图例： Smart Battery 或ACPI 的实现、监督、与操控，最底层都需要 SMBus （圈处）作为后援，图为简易的多组式智能型电池系统，图中有 Smart Battery A 、B 两组电池。（图／SBS-Forum.org ）

「已妥」与「未妥」机制的强制性差别

不单是电气、时序有别，更深层次的协议机制也有不同。在 $I^2C$中，主控端发送端 (主控端 )要与 接收端 (受控端 )通讯前，会在总线上广播受控端的地址信息，每个接收端都会接收到地址信息， 但只有与该地址信息相切合的接收端会在地址信息发布完后发出「已妥」的回应（ Acknowledge ；ACK），让发送端知道对应的接收端确实已经备妥，可以进行通讯。

但是，$I^2C$并没有强制规定接收端非要做出响应不可，也可以默不作声，即便默不作声， 发送端还是会继续工作， 开始进行数据传递及下达读／写指令， 如此的机制在一般运用中还是可行， 但若是在一些实时 （Real Time ）性的应用上，任何的动作与机制都有一定的时限要求，这种可有可无式的响应法就会产生问题，可能会导致受控端无法接收信息。

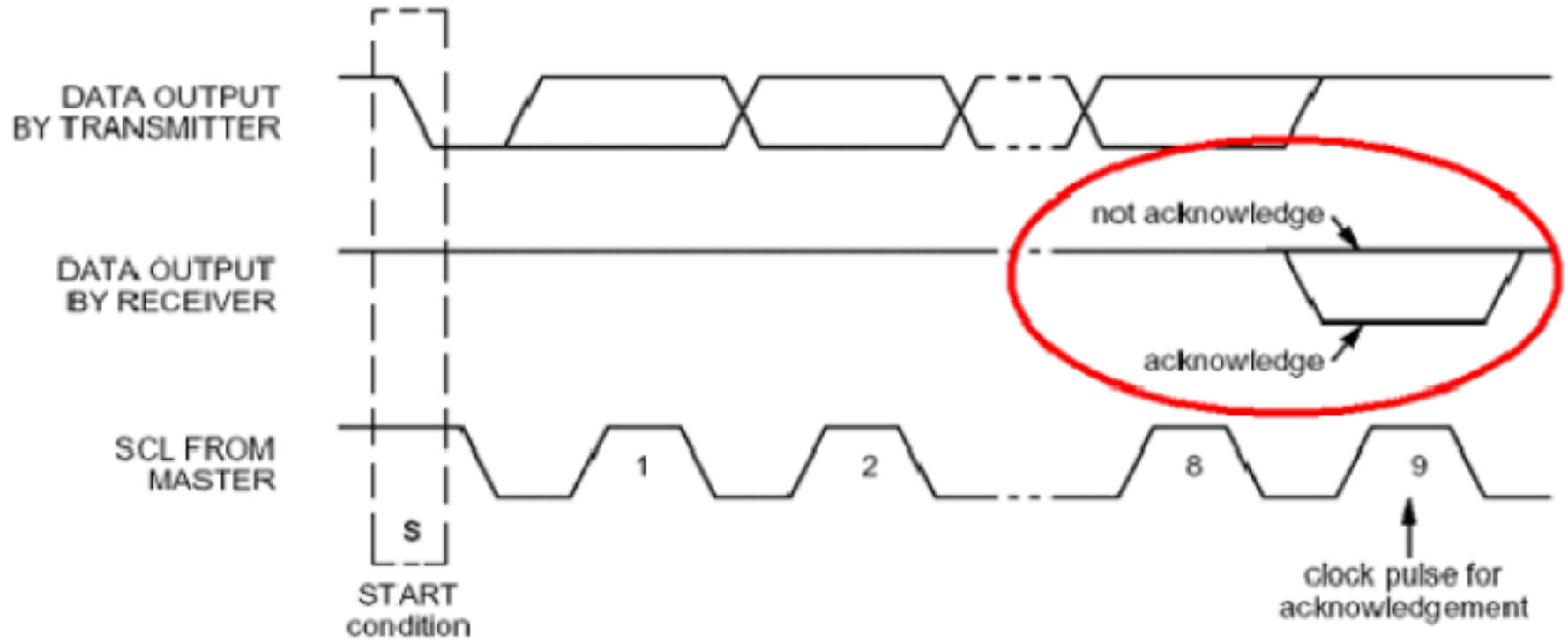
相同的情形，在 SMBus 上是不允许接收端在接收地址信息后却不发出回应，每次都要回应，为何要强制回应？其实与 SMBus 的应用息息相关， SMBus 上所连接的受控装置有时是动态加入、 动态移除的， 例如换装一颗新电池， 或笔记本电脑接上 DOCK PORT 等，如果接入的装置已经改变却没有回应， 则主控端的程

序所掌握的并非是整体系统的最新组态，就会造成误动作。

类似的情形也适用于 ACPI，PC机内机外经常有一些装置可动态增入、 移除，如机内风扇、外接打印机等，这些也一样该强制对主控端群发 (广播)的地址信息作出完整响应。

地址动作方面有异，数据传输方面也有异。在 $I^2C$方面， Slave 虽然对 Master 所发出的地址作出响应， 但在后续的数据传递中， 可能因某些事务必须先行处理、因应而无法持续原有的传输， 这时候 Slave 就要对 Master 发出「未妥」的回应（Not Acknowledge ；NACK），向 Master 表示Slave 正为他务忙碌中。

而SMBus 方面，与$I^2C$相同的，会以 NACK 的回讯向 Master 表达Slave 尚未收妥传递的信息，但是 SMBus 的Slave 会在后续的每个 Byte 传输中都发出 NACK 回信，这样设计的原因是因为 SMBus 没有其他可向 Master 要求重发（ Resend ）的表示法。更直接说就是： NACK 机制是 SMBus 标准中的强制必备， 任何的讯息传递都很重要，不允许有漏失。




▲图说： $I^2C$在完成一段地址或数据信息的传输后，受接端可发出讯息收妥（ ACK ）、未妥（NACK 的响应， SMBus 也具相同的机制，但由于应用之故有更强制的回显请求。（图／Semiconductors.Philips.com ）

传输协议的子集、超集

互动知会机制上有强制与否的差别，协议方面也是。 SMBus 的通讯协议与协议中所用的讯息格式， 其实只是取自 $I^2C$ 规范中， 对于数据传输格式定义中的子集合（Subset ）而已。所以，如果将 $I^2C$与SMBus 交混连接，则 $I^2C$装置在存取SMBus 装置时，只能使用 SMBus 范畴的协议与格式，若使用 $I^2C$的标准存取方式反而无法正确存取。

另外，$I^2C$规范中有一种称为 「General Call 」的广呼方式， 当发出「0000000 」的地址信息后，所有 $I^2C$上的Slave 装置统统要对此作出反应，此机制适合用在Master 要对所有的 Slave 进行广播性讯息更新与沟通上，是一种总体、批次的运作方式。

SMBus 一样有 General Call 机制，但在此之外 SMBus 还多了一种特用的ALERT （警讯）机制，不过这必须于频率线与数据线外再追加一条线（称为： SMBSUS ）才能实现， ALERT 虽名为警讯但其实是中断（ Interrupt ）的用意，Slave 可以将 SMBSUS 线路的电位拉低（ ALERT# ，#表示低电平有效），这时就等于向 Master 发出一个中断警讯，要求 Master 尽速为某一 Slave 提供传输服务。

Master 要响应这个服务要求，是透过 I2C/SMBus 的频率线与数据线来通讯，

但要如何知道此次的通讯只是 Master 对 Slave 的一般性通讯？还是特别针对 Slave 的中断需求而有的服务响应？

这主要是透过 Master 发出的地址信息来区别， 若为回应中断的服务， 地址信息必然是「 0001100 」， 当 Slave 接收到「 0001100 」的地址信息，就知道这是 Master 特为中断而提供的服务通讯。

因此，软件工程师须留心， 规划时必须让所有的 Slave 都不能占用「0001100 」这个地址，以供 ALERT 机制运用（当然！若现在与未来都不会用上 ALERT 机制则可尽管占用）。事实上各种进阶的规范标准 （如Smart Battery 、ACCESS.bus 、VESA DDC 等）都在 $I^2C$的短寻址中订立了一些为自用而保留的地址， 这在最初设计与定义时就该有所留意，以免因先行占用而导致日后须改写软件的麻烦。

补充提醒的是， SMBSUS 一样是开集电极外加上拉电阻的线路，所以有一个Slave 将电位拉下后，其余 Slave 侦测到电位被拉下，表示已有 Slave 正在与 Master 进行中断需索与响应服务，须等待抢到中断服务权的 Slave 确实被服务完毕，重新将 SMBSUS 释放回高电平后， 才能持续以「看谁能先将线路电平拉低？」的方式来争取中断服务。

最后，若有进一步兴趣的朋友，建议可参考两份数据：

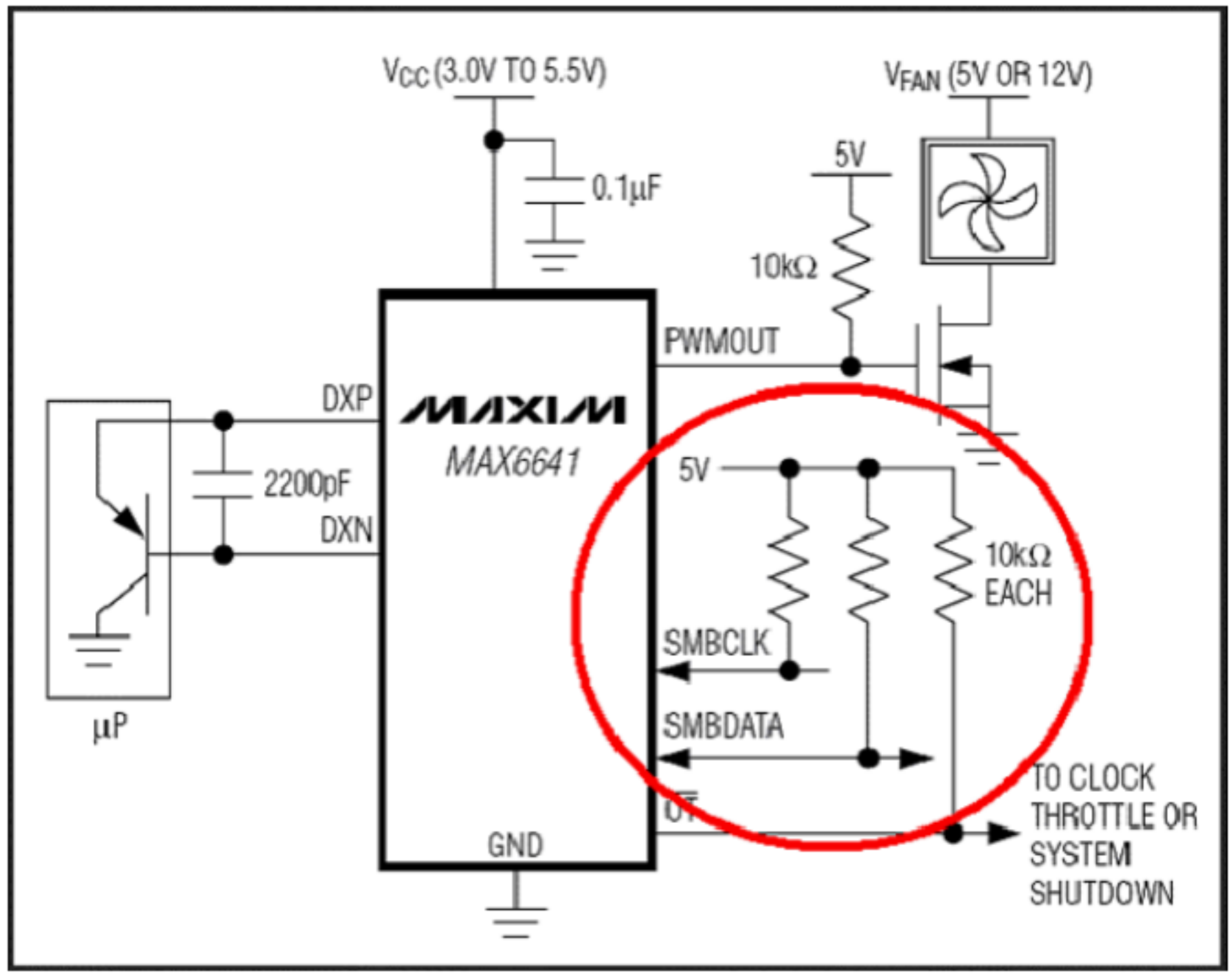
1. SMBus 2.0 版规范（ SMBus 规格网站）
   http://www.smbus.org/specs/smbus20.pdf

2. 比较 $I^2C$与SMBus （MAXIM 公司网站）
   HTML 版：http://www.maxim-ic.com/appnotes.cfm/appnote_number/356
   PDF 版： http://pdfserv.maxim-ic.com/en/an/AN356.pdf

2011-3-10

X

廷敖
作者




▲图例： MAXIM 公司的 MAX6641 芯片之典型应用方式，图左为温度感测电路，图右上为风扇转速控制电路，图右圈处即是 SMBus 接口电路。（图／ MAXIM-IC.com ）