# 总结 STM32 的 I2C 的缺陷与使用

——韩大鹏

STM32 的 I2C 的最大缺陷就是有两个有关数据的寄存器：一个是数据寄存器（即 DR），一个是移位寄存器。不止是缺陷，在某种程度上说就是个错误，果然是性价比极高，寄存器都比一般的多啊……相关于这两个寄存器，有两个状态位：RXnE/TxE 和 BTF，RXnE/TxE 由于分别对应于收与发模式所以我把它们算作一个。

## 1.先说发送

发送时首先现将数据写入数据寄存器，如果移位寄存器是空的，DR 中的数据被写入移位寄存器，移位寄存器负责将数据一位一位的发送出去。

那么相关的标志位在什么时候置位呢？当把 DR 中的数据写到移位寄存器的时候，数据寄存器空了，TxE 被置位，此时你可以写入新的字节。

就这样，整个 I2C 的发送过程就是：移位寄存器发送时序，发完之后数据寄存器将数据写到移位寄存器，你判断 TxE 后将字节写到数据寄存器。我们正常的操作就应该是这样的。如果能一直这样进行下去就好了，那么便不会有这么多迷惘预抱怨与失望了。

想一下，如果由于某些原因（说的保守了，其实就是中断），在 TxE 被置位后，你没能及时将数据写入 DR 中，会发生什么情况？移位寄存器在把其中的字节发完后也会变空，注意了，此时 BTF 被置位！在发送中 BTF 被置位就标志着两个寄存器都空了，会怎么样？会发生“时钟延展”，这个我们不必关注，我只是提一下。我们关注的是什么？是事件！在 I2C 的固件库中，查询方式的 EV8 中是包含 BTF 的，不知道是哪位神级的工程师，直接帮我们把 BUG 考虑进去了，我们用起来不会出错，只是速度会变慢，本来无需等两个寄存器都空才写新的数据的，只要数据寄存器空就可以写了。但是中断方式的固件库中就比较诡异了，它有两个 EV8 相关事件，就是包含 BTF 和不包含 BTF 的，它采用的就是判断不带 BTF 的事件，这样，你就不能有任何的停顿（中断），一停顿就变成带 BTF 的（当然也不全是，是中断超过十个时钟周期，也有可能不到十个，这边对时间就不说的很严格了）。为什么 ST 要求中断优先级最高？就是这样。被其他中断一打断就完了~~按照他的固件库，你的事件错了，结果操作就错了！

其实发送这也不算什么毛病了，你就查查 BTF 呗，速度慢一点就慢一点呗，还是可以正常使用的，中断也不用什么最高优先级了。致命的缺陷其实在就在接收中，各位看官请继续瞧吧。

## 2.再说接收

看完了发送相信各位看官对其 I2C 的工作方式有一定了解了吧。读的情况与写倒过来。

接收时，首先将总线上的数据一位一位的收到移位寄存器中。如果 DR 是空的，则移位寄存器中的数据转移到 DR 中，此时 RxNE 置位，此时你可以读出 DR 的数据。

所以，同发送类似，接收的正常过程就是：移位寄存器接收时序，接收之后移位寄存器中的数据转移到数据寄存器，你判断 RxNE 后将数据寄存器的字节读出。

那么，还是同于发送，由于中断，在 RxNE 后你没能将 DR 中数据及时读走，那么过一会移位寄存器也满了，注意，此时 BTF 被置位了。同样，发生“时钟延展”，当然同样无需关注。关注的还是事件，带 BTF 的和不带 BTF 的。这个时候，I2C 的固件库达到了统一，查询方式和中断方式的均采取了不带 BTF 的。但是，若有中断，便会出错。你只能使用最高优先级中断。

前面发送那边不是说了吗，有什么大问题？查查 BTF 位呗，慢就慢点呗！但是，请注意，到接收这边，行不通了。使用过 I2C 的都知道，在接受最后一字节之前要配置 NACK 位以使 I2C 接收最后一字节后产生 NACK 以通知从机传输完毕使从机释放总线。但是 BUG 由此产生，在倒数第二字节的时候若发生中断呢？倒数第二字节进入 DR，倒数第一字节（也即最后字节）进入移位寄存器。NACK 没有配置，没有发 NACK，从机不释放总线，而主机却以为完事了……（从机：开始了吗？主机：已经完事啦~~）总线由此错误，天下大乱了。

我一直在思考，为什么 ST 要用两个有关数据的寄存器呢？在我对 ST 两个有关数据的寄存器的用法产生质疑后，我就上网搜索大量带 I2C 或 TWI 的芯片的硬件数据寄存器实现，清一色的“数据移位保持寄存器”，只有一个，每次只能收发一字节！中断优先级不需要最高。也没听说发生过什么难用的事。可见 STM32 的 I2C 是多么大的一个败笔啊!! 当然，STM32 这个芯片的性价比据说还是相当高的。其它模块我也没用过，不知性能如何，暂不做评论。

## 下面说一下在忍受内心想狂骂的煎熬下的使用：

1.若你使用的系统只是一个死循环，没有中断。那么恭喜你，你可以随意使用固件库任何方式，ST 童叟无欺，绝对好用。

2.查询方式：　发　采用带 BTF 的事件。可以保证写的正确。

　　收　（1）若接受的字节大于两个，查询带 BTF 事件。在最后两字节前的

时候配置 NACK，并且屏蔽中断，倒数第二字节查询 BTF 事件，最后字节查询不带 BTF 事件。

（2）若小于等于两字节，屏蔽中断，在最后字节前的时候配置 NACK，皆查询不带 BTF 事件。

3.中断方式：

（1）最简单的，使用最高优先级，并且得是抢占式的，并且其他函数中不能有任何屏蔽中断的操作，总之一句话，就是给 I2C 绝对的最高优先级，不能耽误一刻。（其实有时候可以耽误在几个 i2c 时钟周期内，但是如何保证？得看运气!）

（2）不想使用绝对最高优先级的情况下：

发送　判断 BTF 事件，也能正常使用。

接收　也是判断 BTF 事件，在倒数第二字节配置 NACK，最后字节判断不带 BTF 事件。但是，在中断中不像查询中，你可以想判断哪个就是那个。都是 CASE，怎么控制呢？一般的时候都直接不带 BTF 事件就过了，而这样的话，你在倒数第二字节配置 NACK 便是错误的。不好搞啊，不给力啊。也好，就这样吧，咳咳……

4.DMA 方式：本人没试过 DMA 方式。应该可以使用，因为其没有开 BUF 中断。判断机制不一样。

5.使用 IO 模拟方式，简单易用，可移植性强，等等。缺点就是浪费了那么快的硬件电路。