AN1471
Application note

What happens to the M24xxx I²C EEPROM if the I²C bus communication is stopped?

This application note describes what can be attempted to set an M24xxx memory back to a known state if it has been suddenly stopped before completion of the current I²C instruction.

The method presented here will work regardless of whether the device was stopped during an incoming or an outgoing byte transfer. It is used to resynchronize the memory device whenever an undefined state has been detected on the I²C bus.

1 Resynchronizing the M24xxx's internal logic

If the bus Master (the microcontroller or processor) or other components on the I²C bus have failed, with clock and data lines being improperly driven, the internal state of the M24xxx might reach an unknown state. The M24xxx internal logic must be resynchronized. The analysis of this situation can be structured under the following sub-headings:

- *The interrupted transmission was an incoming data byte*
- *The interrupted transmission was an outgoing data byte (during a READ instruction)*

1.1 The interrupted transmission was an *incoming data byte*

Issuing a Stop condition is sufficient to abort the transmission and place the device in the Standby mode. However, if the last transmitted instruction was a WRITE, the Stop condition is also able to start the internal Write cycle (if the Stop condition occurs after the 9th clock cycle of each data byte). It is therefore risky to send a single Stop condition.

It is recommended, instead, to issue a Start condition first, followed by a Stop condition. The Start condition aborts the transmission, and leaves the M24xxx waiting for a device select byte; the Stop condition then sets the M24xxx in standby mode.

Caution: Resynchronization does not modify the internal address counter.
In order to define the internal address counter value, the next instruction must be either a Byte Random Read, a Sequential Random Read or a Write (because the current Read or Sequential Read instruction does not modify the internal address counter).

1.2 The interrupted transmission was an outgoing data byte (during a READ instruction)

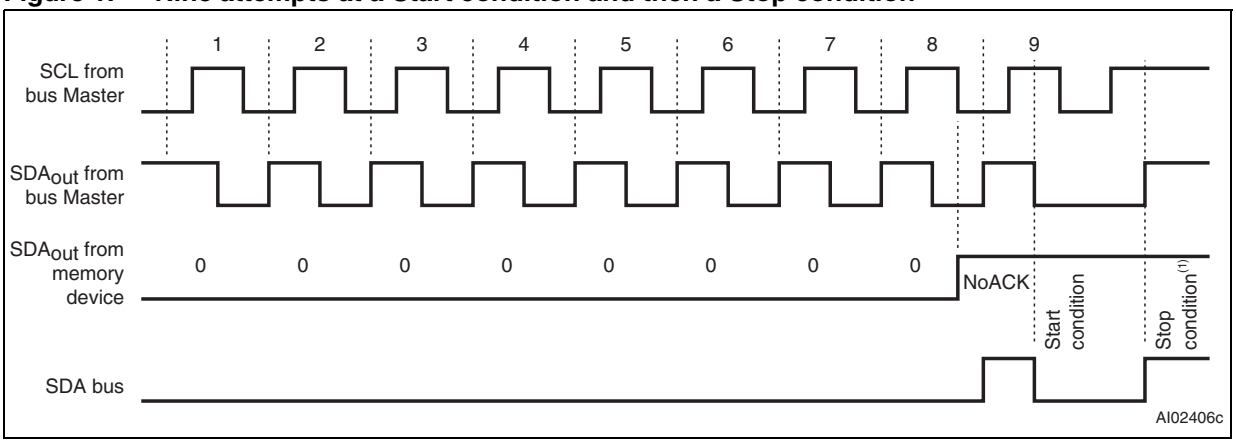
A tricky configuration might be reached if, after loosing the bus communication control, the M24xxx memory is stopped when outputting a 0 on SDA. In such a case, even if the bus Master transmits a Start condition, the M24xxx memory cannot decode it as the SDA bus line is forced to 0. A good way to work around this state is to have the bus Master sending several clock cycles until the M24xxx outputs a 1: once the 1 state has been output, the bus Master sends a Start condition which will be correctly decoded by the M24xxx memory.

This result is always reached using a blind [9 Start + Stop] sequence, with two possible outcomes:

1. **worst case:** when it lost the bus communication control, the M24xxx memory device was starting to clock out eight 0s (the data byte was 00h), as shown in *Figure 1*. As the ninth clock pulse makes the M24xxx output a NoAck state ("SDAout from memory device" is 1), the falling edge of "SDAout from bus Master" correctly drives the SDA bus line and this event is decoded as a Start condition by the M24xxx memory.
2. **standard case:** when it lost the bus communication control, the M24xxx memory device was starting to clock out a byte composed of 0s and 1s; the Start condition is decoded sooner, that is with the first bit of data "SDAout from memory device" =1. In such a case, the M24xxx memory decodes N successive Start conditions (one Start condition for each "SDAout from memory device" =1) which restart the M24xxx memory N times, before decoding the final Stop condition (see *Figure 1*).

Caution: Resynchronization does not modify the internal address counter.
In order to define the internal address counter value, the next instruction must be either a Byte Random Read, a Sequential Random Read or a Write (because the current Read or Sequential Read instruction does not modify the internal address counter).

Figure 1. Nine attempts at a Start condition and then a Stop condition



1. The Stop condition following the Start condition sets the M24xxx in Standby mode.

1.3 Conclusion

The blind [9 Start + Stop] sequence shown in *Figure 1* is specially designed to ensure that the I²C bus Master will, at the end of this sequence, succeed the resynchronization of the M24xxx.

2 Revision history

Table 1. Document revision history

Date	Version	Changes
07-Nov-2001	1.0	First Issue
27-Sep-2002	1.1	Minor revision
25-Aug-2004	2.0	Indication of improvements that might be possible outside the worst cases
28-Aug-2006	3	Application note revised to limit the topic to the case where the I²C bus communication is lost.
09-Oct-2009	4	Small text changes in *Section 1.1: The interrupted transmission was an incoming data byte* and *Section 1.2: The interrupted transmission was an outgoing data byte (during a READ instruction)*. *Section 1.3: Conclusion* added.