

Interrupting a 2-wire Read

Characteristics of a 2-wire FRAM



Overview

The industry standard 2-wire bus is a robust protocol for a variety of applications. It supports multiple master and slave devices and is intended for harsh environments. Unfortunately, it has one limitation that system developers may encounter. Interrupting a memory read in progress may be difficult.

A read operation involves the master (receiver) issuing clocks and the serial FRAM driving data back to the master. Successive read operations will continue as long as the master acknowledges each byte. That is, the memory device's internal address is automatically incremented and the next data byte is driven on the SDA line. The Acknowledge (ACK) condition is SDA high while SCL is clocked high and low. After the memory has driven the last data bit, it releases the SDA line, allowing it to be pulled high by the external pull-up resistor. At this time, the master can either acknowledge or not acknowledge the data byte. The master is said to ACK (acknowledges) the data byte by driving SDA low, or it can NACK (no-acknowledge) the data byte by leaving SDA high during the next clock.

The preferred method for ending a memory read operation is a NACK following the 8th data bit. The master allows the SDA line to remain high for the 9th clock. The timing for this operation is shown in Figure 1. After a no-acknowledge, the master can assert a STOP, START, or other condition.

In most cases, this protocol is sufficient. The most common mistake made by developers unfamiliar with the protocol is to issue an ACK after the last desired byte. This will cause the serial FRAM to start driving a new data byte on the next clock. Since this is unexpected by the master, the likely result is bus contention. This condition can be corrected simply by giving a 'no-acknowledge' after the last desired byte.

Occasionally, the system may require aborting a read prior to the end of a byte. The 2-wire protocol does not offer a graceful method to accomplish this. This situation is not unique to the serial FRAM devices, but this application note explains the method for clearing the condition.

Methods for Aborting a Read

Certain system requirements may require a read termination as quickly as possible. This could result in stopping the transaction prior to the end of a byte. It is possible to abort a read in two ways. Neither is ideal and the need for these methods usually results from an undesirable system condition.

First, a read may be aborted by forcing a STOP condition. The signal relationship for a STOP is shown in Figure 2. As shown, the STOP requires the master to drive the SDA signal from a low to a high level while SCL is high. Forcing a STOP requires overdriving the output of the memory device that may already be driving low. If the memory is transmitting a '1' value, then the SDA is in a high condition. The master can drive the bus low (for START), then high (for STOP), quite easily. Setup time specifications for START and STOP must be met. Difficulty results when the serial memory is driving the SDA line in a low condition. This means that the data bit value being output is a '0'. The master can use a strong active-high driver to force the state of the pin to logic high. This will generate the rising edge needed to create the STOP condition.

The current drive characteristics of the serial FRAM SDA pin are shown in figure 3 below. It is necessary to pull the SDA line up to $V_{CC} * 0.7$ or nominally 3.5V to create a logic high condition. This will require approximately 67 mA at 3.5V until the new condition is recognized.

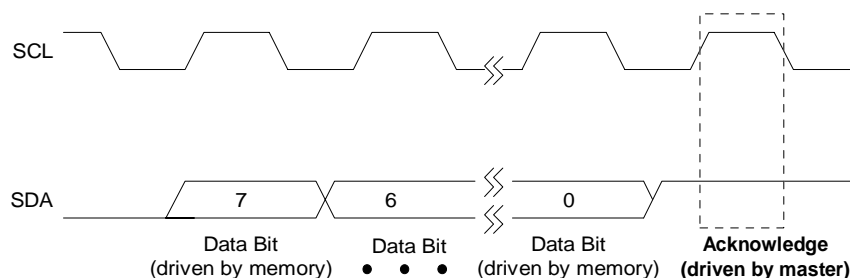


Figure 1. Preferred Method to Terminate a Read Operation

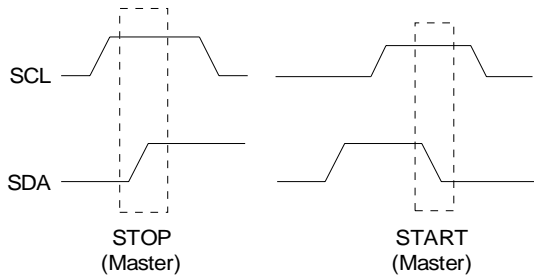


Figure 2. STOP and START Condition Timing

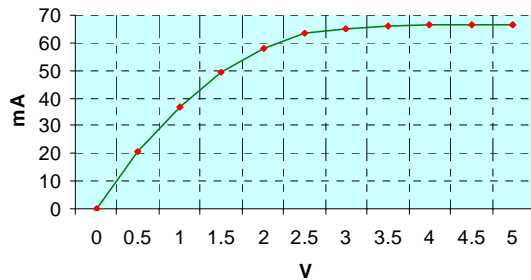


Figure 3. Current Drive of SDA

New serial FRAM devices will have a substantially weaker SDA driver, and consequently they will be easier to forcibly pull up. A system that must initiate this action should be designed with a very strong pull-up transistor on the master(s) SDA pin.

The second way to abort a pending read operation is to issue clocks until a data bit value of '1' occurs. A '1' data bit will cause the serial FRAM to release the SDA line since it only drives low. To make this method work, the master issues an SCL rising edge, then reads the SDA line while SCL is high. If SDA is '0', then another clock must be issued. If SDA is '1', then the master can force SDA low while SCL is high. This is a START condition. A START condition also will cause the memory to end the read operation immediately. Figure 2 shows the timing of a START condition.

This is the fastest method to abort a read operation if the bus cannot be forced. Figure 4 illustrates the concept. Unfortunately, the time to terminate a read is unknown since it is data dependent. At most, nine clocks will be required since this will be enough to read an entire byte and proceed to the acknowledge state.

This procedure also assumes that the master still recognizes its ownership of the 2-wire bus. Unfortunately one of the reasons to abort early is that the device that was the master is reset and is no longer the master. This situation is complex since no device owns the bus. Any device seeking ownership will see that SDA is low, and therefore the bus is not idle. In this case, it is helpful if a host processor recognizes that the bus is hung and implement the early termination protocol as described above. An activity timeout while SDA is low is one method of determining that the bus is hung.

Alternatives to 2-Wire Devices

The 2-wire protocol offers robust communication in multi-master, multi-slave environments. It is also considered cost effective. In order to accomplish these dual missions, the 2-wire bus has a rigid protocol. Hardware devices that strictly follow it can become stranded by system faults unless software that is more flexible can intervene.

An alternative method uses another serial bus, such as SPI. The Serial Peripheral Interface is designed for higher speed communication with a far simpler protocol. Because it operates at a more basic level, users must determine their own multi-master arbitration protocol. However, for early termination of reads, it is far simpler. The memory (slave) devices use chip select, data I/O, and serial clock pins. Each slave device is enabled by an active low chip select. Thus, a system with multiple masters and multiple slaves requires a decoding scheme and more connections. However, any operation can be terminated at any time by deasserting chip enable.

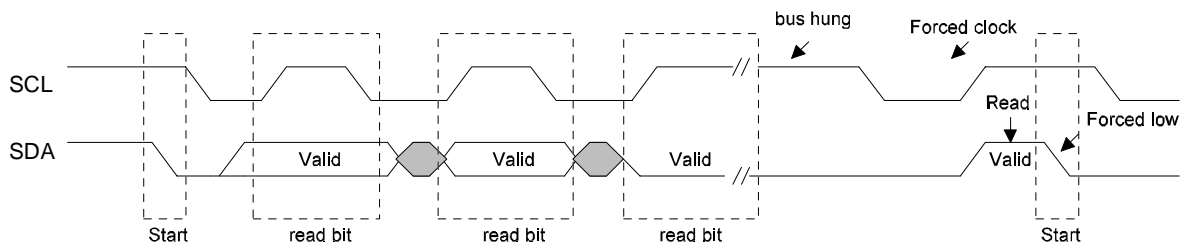


Figure 4. Aborting a Read Operation