Communication I2C

# Repeated Start Condition

# A way to claim the bus

During an I2C transfer there is often the need to first send a command and then read back an answer right away. This has to be done without the risk of another (multimaster) device interrupting this atomic operation. The I2C protocol defines a so-called repeated start condition. After having sent the address byte (address and read/write bit) the master may send any number of bytes followed by a stop condition. Instead of sending the stop condition it is also allowed to send another start condition again followed by an address (and of course including a read/write bit) and more data. This is defined recursively allowing any number of start conditions to be sent. The purpose of this is to allow combined write/read operations to one or more devices without releasing the bus and thus with the guarantee that the operation is not interrupted.

![A description...](data:None;base64,)  
Regardless of the number of start conditions sent during one transfer the transfer must be ended by exactly one stop condition.

# Clock Synchronization

A DLN adapter and one or more I2C masters can begin transmitting on a free I2C bus at the same time. Each master generates its own clock on the SCL line. Therefore, there must be a method for deciding which master generates LOW and HIGH periods of the SCL line. Clock synchronization does it.

Once a DLN adapter or any other I2C master outputs LOW on its clock line, the SCL line goes LOW. When a master releases its clock line, the SCL line goes HIGH only if no other master has its clock line in LOW state. The master with the longest LOW period holds the SCL line in LOW state. Masters with shorter LOW periods stay in a HIGH wait-state during this time.

When all masters concerned have released their clock lines, the SCL line goes HIGH and all the masters start counting their HIGH periods. The first master that completes its HIGH period pulls the SCL line LOW again.

Therefore, the master with the longest clock LOW period determines a LOW period on the SCL line; the master with the shortest clock HIGH period determines a HIGH period on the SCL line.

The following figure shows clock synchronization for the DLN adapter and the Master2 device. The DLN adapter has a shorter HIGH period; it pulls the SCL line LOW. The Master2 device has a longer LOW period, only when it releases its clock line, both masters start counting HIGH period.

![A description...](data:None;base64,)

## Clock Generation

The SCL clock is always generated by the I2C master. The specification requires minimum periods for the low and high phases of the clock signal. Hence, the actual clock rate may be lower than the nominal clock rate e.g. in I2C buses with large rise times due to high capacitances.

## Clock Stretching

I2C devices can slow down communication by stretching SCL: During an SCL low phase, any I2C device on the bus may additionally hold down SCL to prevent it from rising again, enabling it to slow down the SCL clock rate or to stop I2C communication for a while. This is also referred to as clock synchronization.

Note: The I2C specification does not specify any timeout conditions for clock stretching, i.e. any device can hold down SCL as long as it likes.

## Arbitration

Several I2C multi-masters can be connected to the same I2C bus and operate lconcurrently. By constantly monitoring SDA and SCL for start and stop conditions, they can determine whether the bus is currently idle or not. If the bus is busy, masters delay pending I2C transfers until a stop condition indicates that the bus is free again.

However, it may happen that two masters start a transfer at the same time. During the transfer, the masters constantly monitor SDA and SCL. If one of them detects that SDA is low when it should actually be high, it assumes that another master is active and immediately stops its transfer. This process is called arbitration.