

Introduction to I2C & SPI

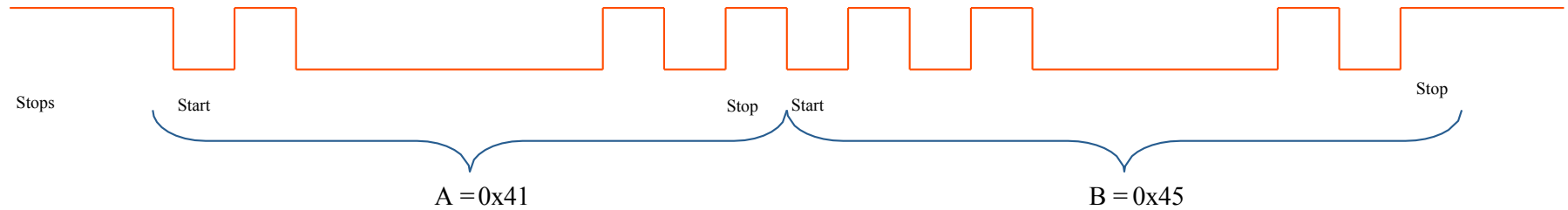
Issues with Asynch. Communication Protocols

Asynchronous Communications

- Devices must agree ahead of time on a data rate
- The two devices must also have clocks that are close to the same rate
- Excessive differences between clock rates on either end will cause garbled data
- Asynchronous serial ports require hardware overhead
- The UART at either end is relatively complex and difficult to accurately implement in software if necessary
- Most UART devices only support a certain set of fixed baud rates, and the highest of these is usually around 230400 bits per second

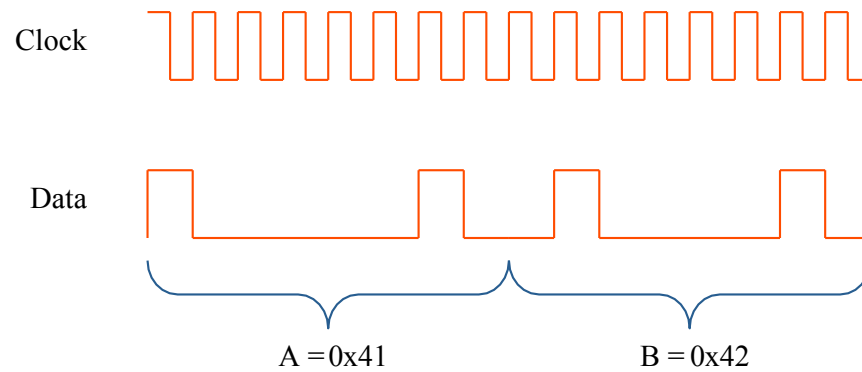
Asynchronous Transmission

Example of an "A" followed by a "B"



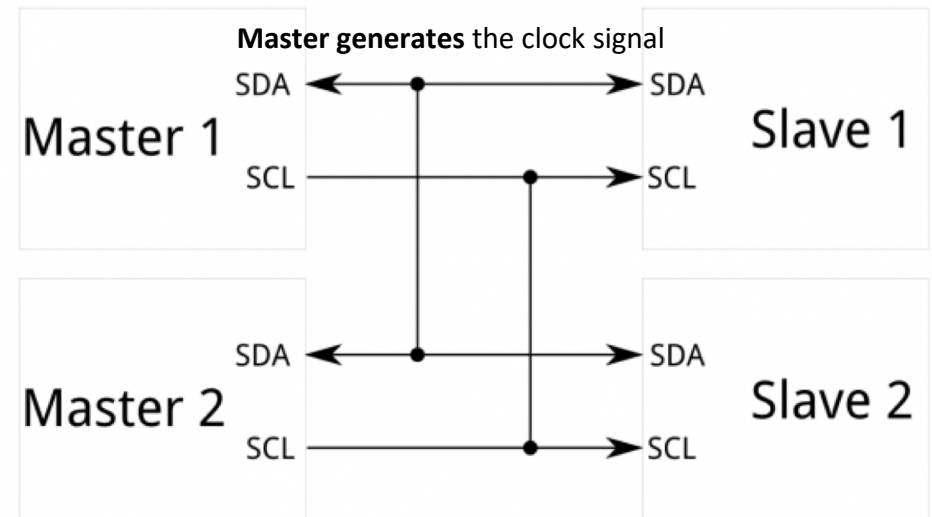
Synchronous Transmission

Example of an "A" followed by a "B"



The Inter-integrated Circuit (I²C)

- The Inter-integrated Circuit (I²C) Protocol is a protocol intended to allow multiple “slave” (or **secondary**) digital integrated circuits (“chips”) to communicate with one or more “master” chips.
- Multi-master system, allowing more than one master ” (or **primary**) to communicate with all devices on the bus
- When multiple primary devices are used, the master devices can’t talk to each other over the bus and must take turns using the bus lines.
- In I²C there are **three additional** modes specified: fast-mode plus, at 1MHz; high-speed mode, at 3.4MHz; and ultra-fast mode, at 5MHz.



Characteristics

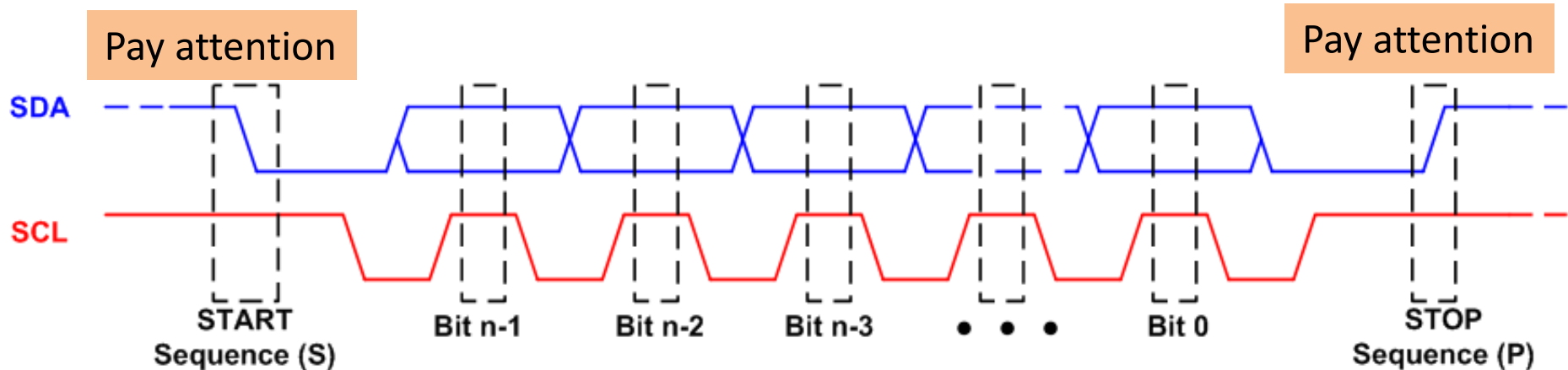
- Serial, byte-oriented
- Multi-master, multi-slave
- Two bidirectional open-drain lines, plus ground

Serial Data Line (SDA)

Serial Clock Line (SCL)

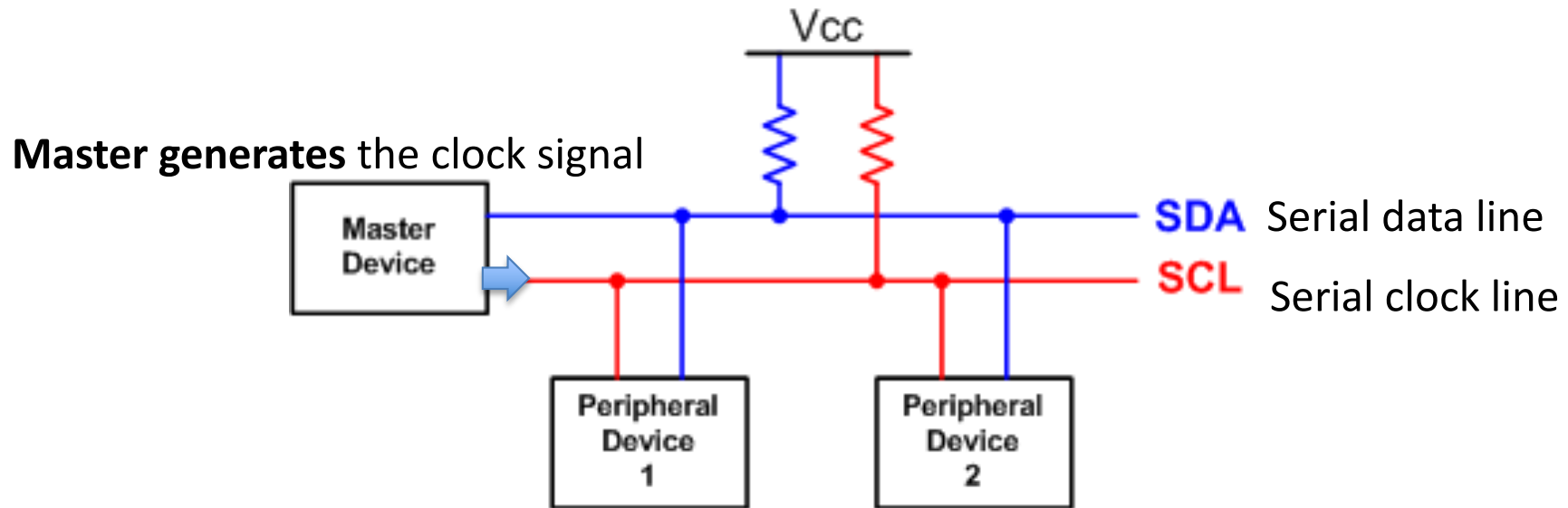
SDA and SCL need to pull up with resistors

Timing Diagram



- A **START** condition is a high-to-low transition on SDA when SCL is high.
- A **STOP** condition is a low to high transition on SDA when SCL is high.
- The address and the data bytes are sent most significant bit first.
- **Master generates** the clock signal and sends it to the slave during data transfer

Inter-Integrated Circuit (I2C)

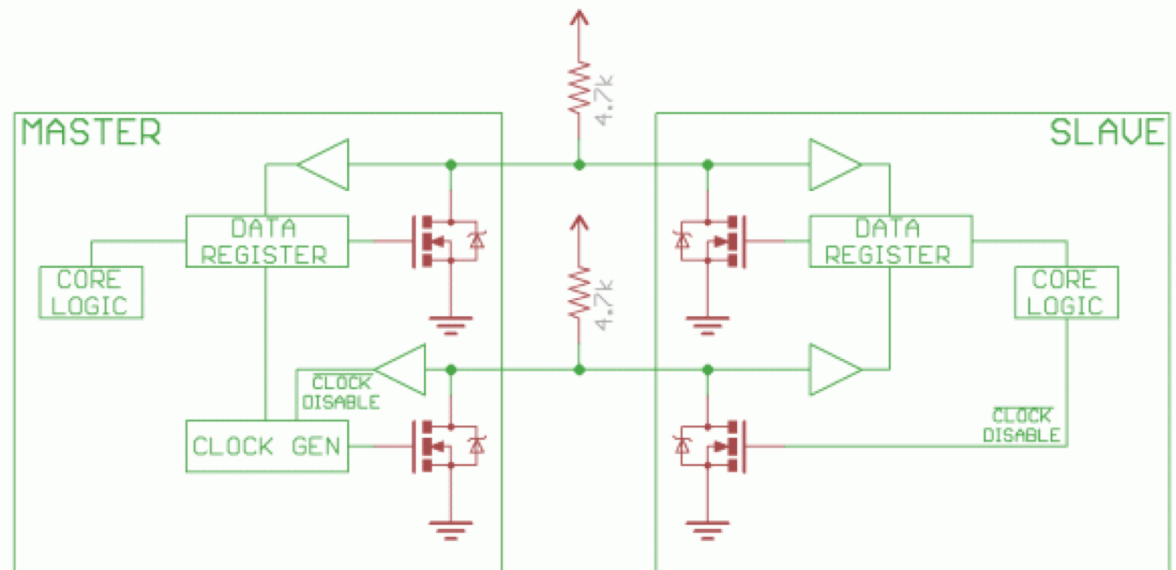


- SDA and SCL have to be **open-drain**
 - Connected to positive if the output is 1
 - In high impedance state if the output is 0
- Each Device has an unique address (7, 10 or 16 bits). Address 0 used for broadcast
- STM32 internal pull-up is too weak (internal 100K Ω)
- External pull-up (4.7 k Ω for low speed, 3 k Ω for standard mode, and 1 k Ω for fast mode).
 - Fast mode refers to fast rise time!

Inter-Integrated Circuit (I2C)

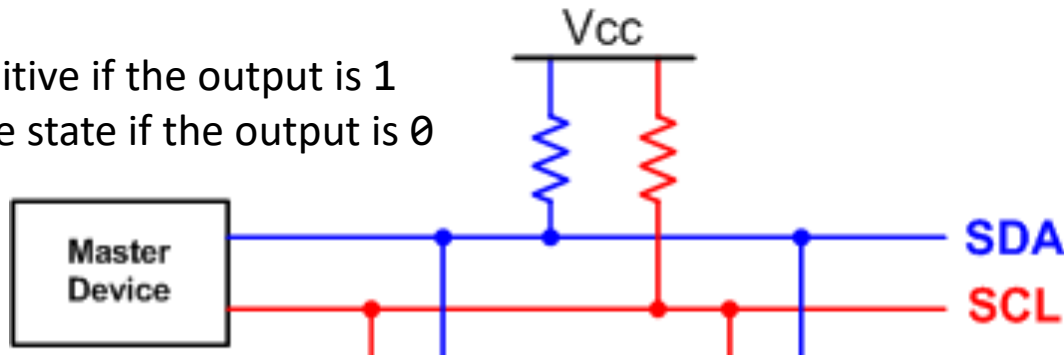
- The I²C bus drivers are OPEN DRAIN meaning that they can pull the corresponding signal line **low, but cannot drive it high**
- There can be no bus contention where one device is trying to drive the line high while another tries to pull it low, eliminating the potential for damage to the drivers or excessive power dissipation in the system
- STM32 internal pull-up is too weak (internal 100K Ω)
- External pull-up (4.7 k Ω for low speed, 3 k Ω for standard mode, and 1 k Ω for fast mode – fast rise time!)

“Wired-AND” bus: A sender can pull the lines to low, even if other senders are trying to drive the lines to high



Multiple Masters

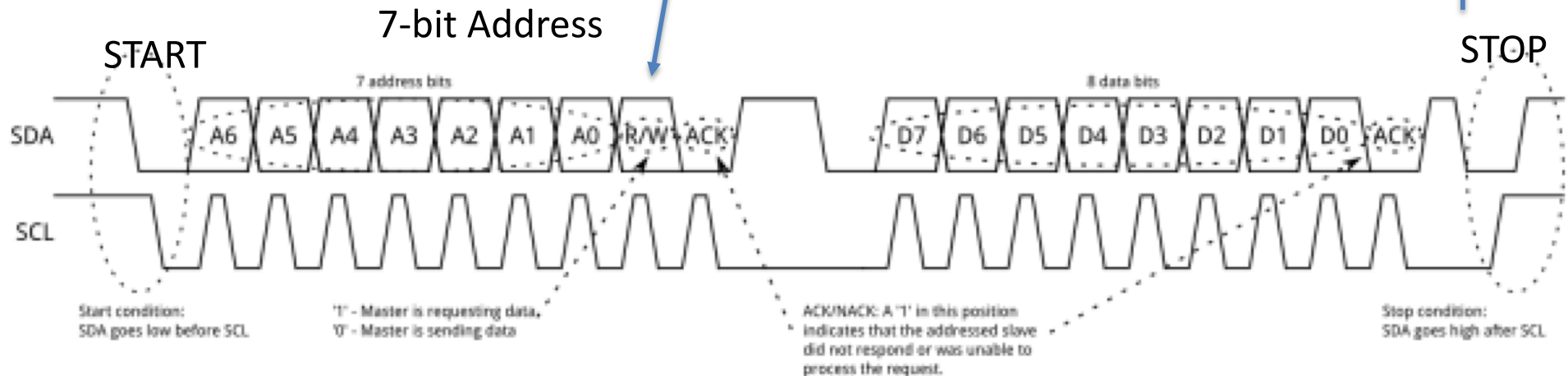
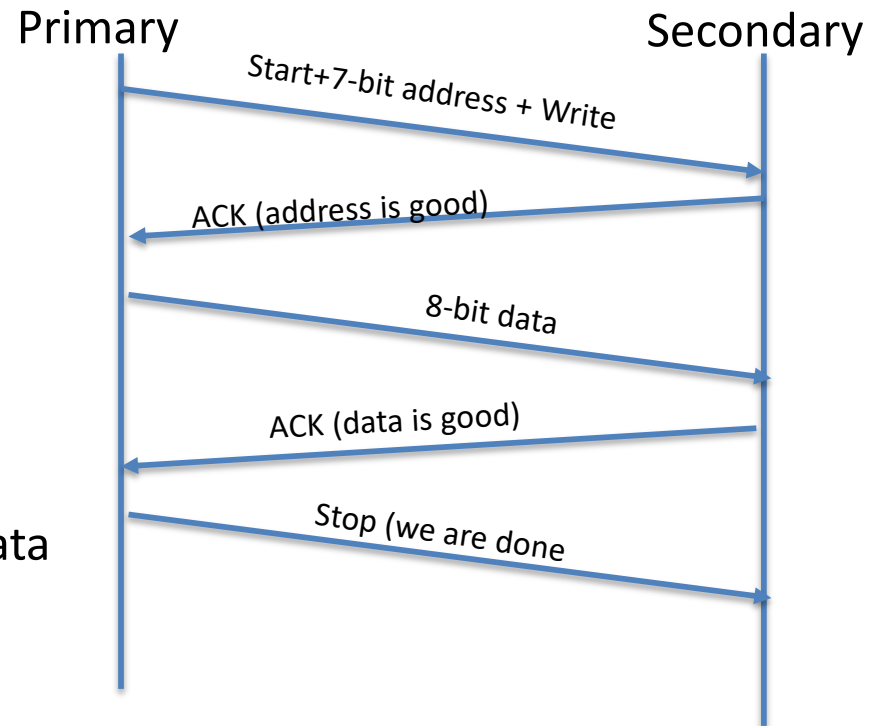
Connected to positive if the output is 1
In high impedance state if the output is 0



- In single master systems, arbitration is not needed.
- **Arbitration** for multiple masters:
 - During data transfer, the master constantly checks whether the SDA voltage level matches what it has sent.
 - When two masters generate a START setting **concurrently**, the first master which detects SDA low while it has actually intended to set SDA high will **lose the arbitration** and let the other master complete the data transfer.

Basic Protocol (7-bit Addressing)

- Setting the address
- Write the data value
- R/W- : 0 --> Primary sends data



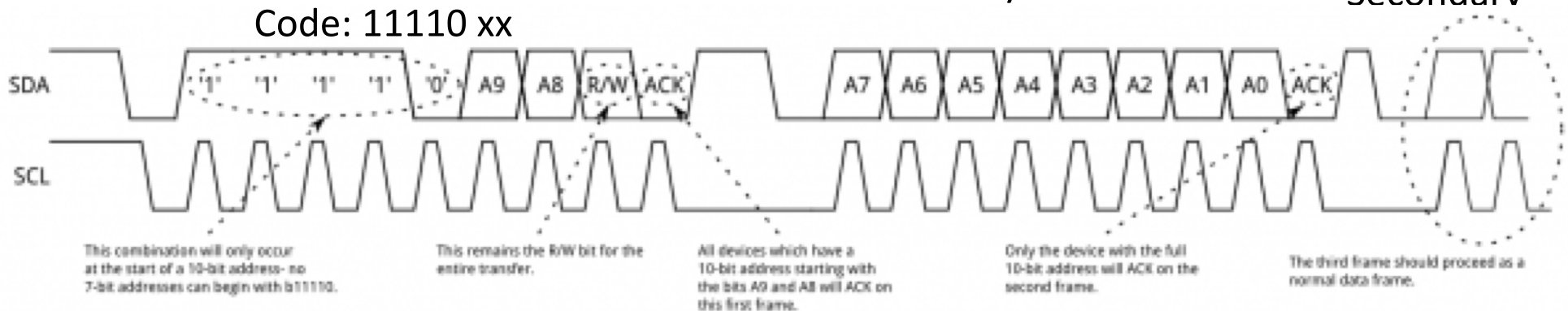
Basic Protocol (10-bit Addressing)

- Setting the address
- Write the data value
- R/W- : 0 --> Primary sends data



Primary

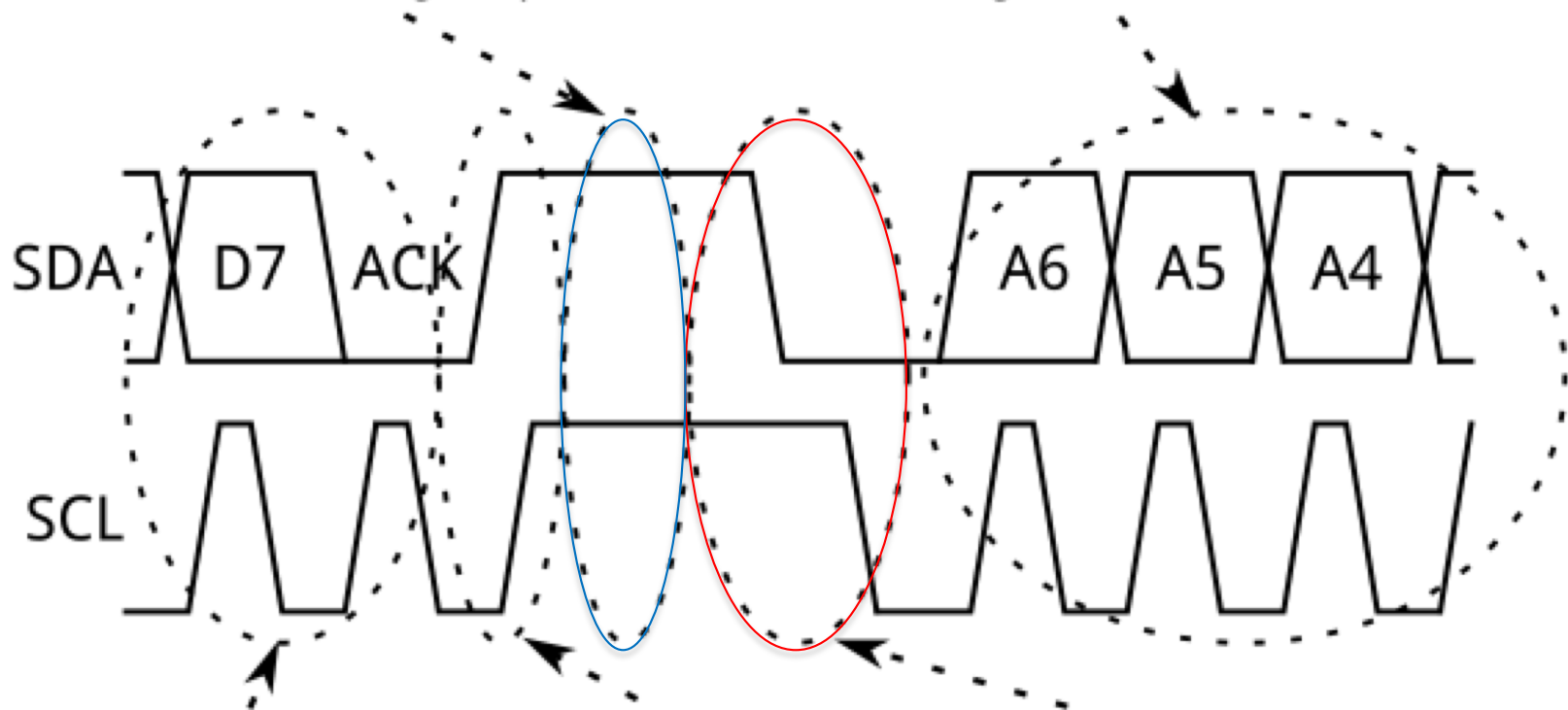
Secondary



Repeated Starts

Despite the idle state of the bus, no other master may assert control of the bus during this period.

After the repeated start, a new transfer, complete with address frame(s), must begin.

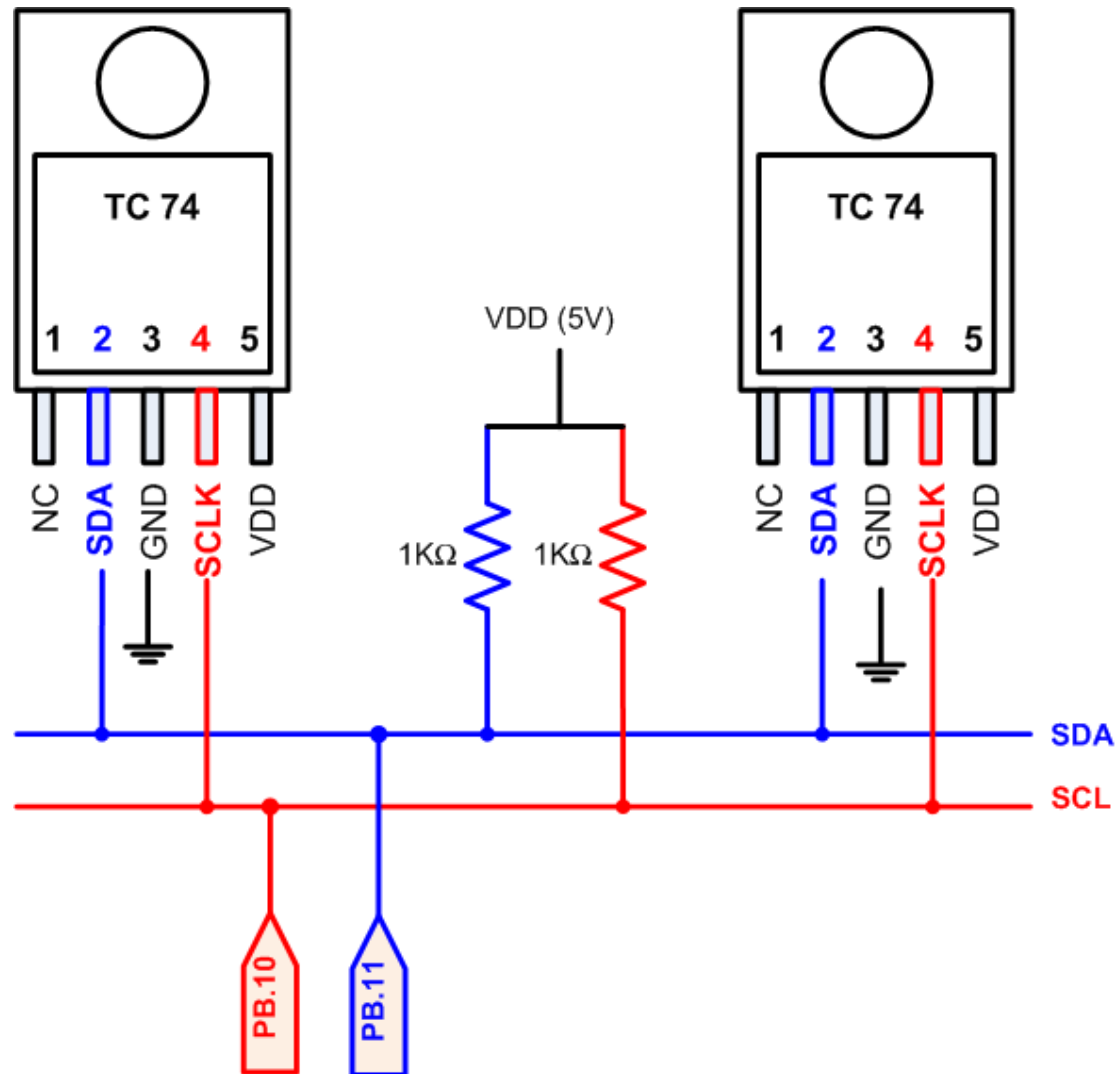


Last data frame of prior transfer

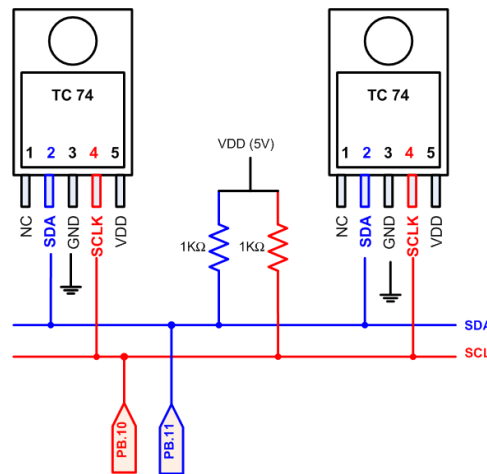
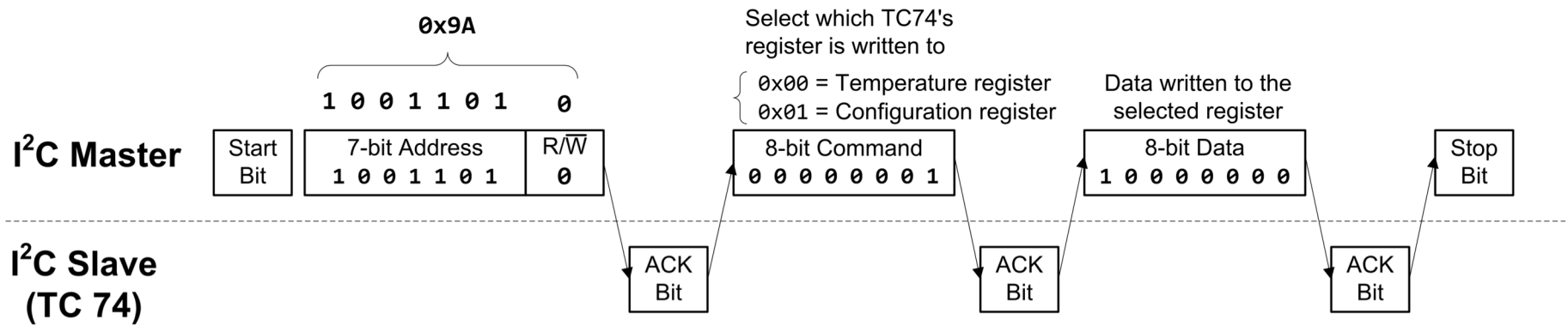
No stop condition is present

Another start occurs- this is the "repeated start"

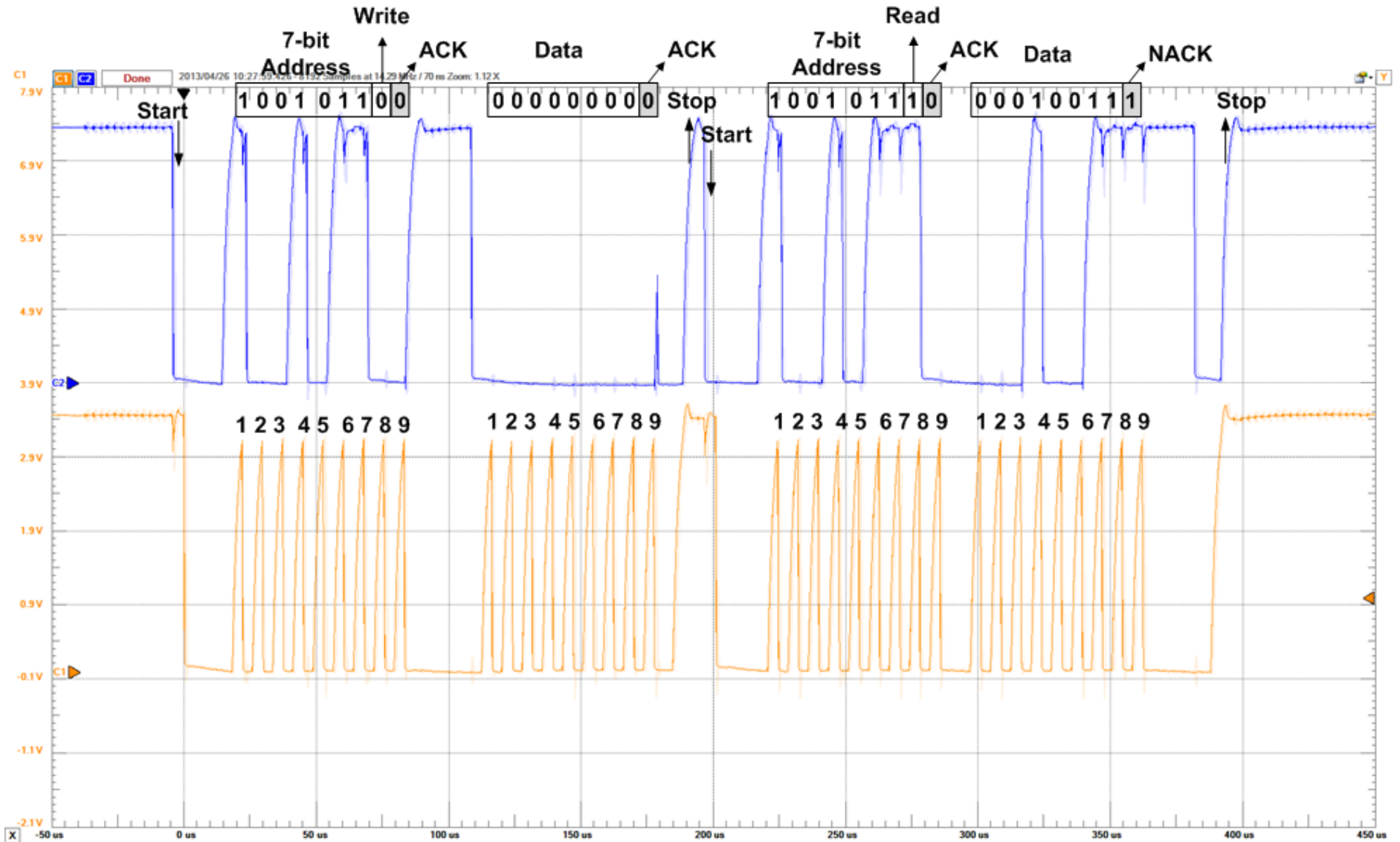
Interfacing Serial Digital Thermal Sensor



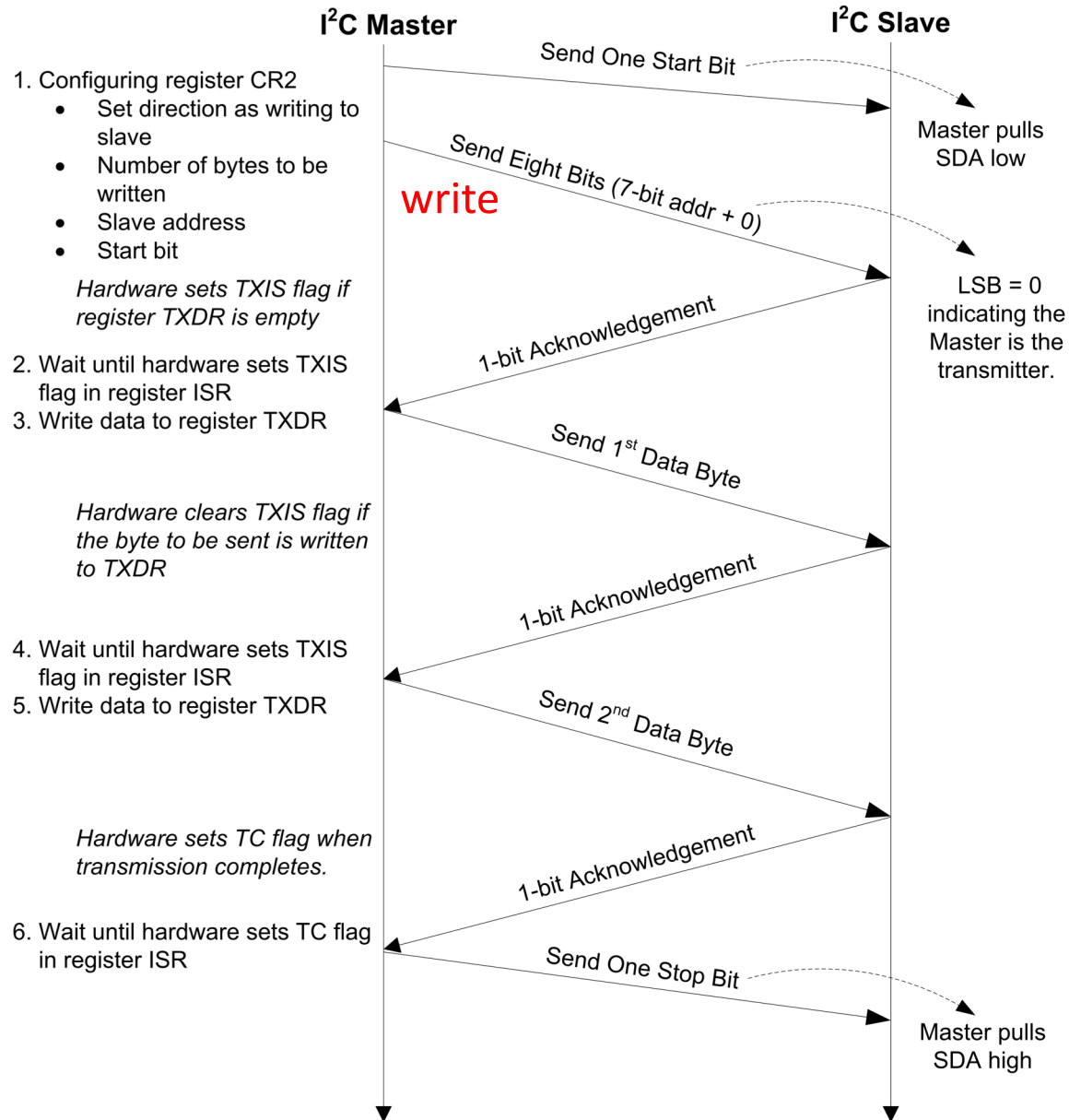
Communicating with TC74 with an address of 0x4D



I2C Data



Sending Data to I2C Secondary Via Polling



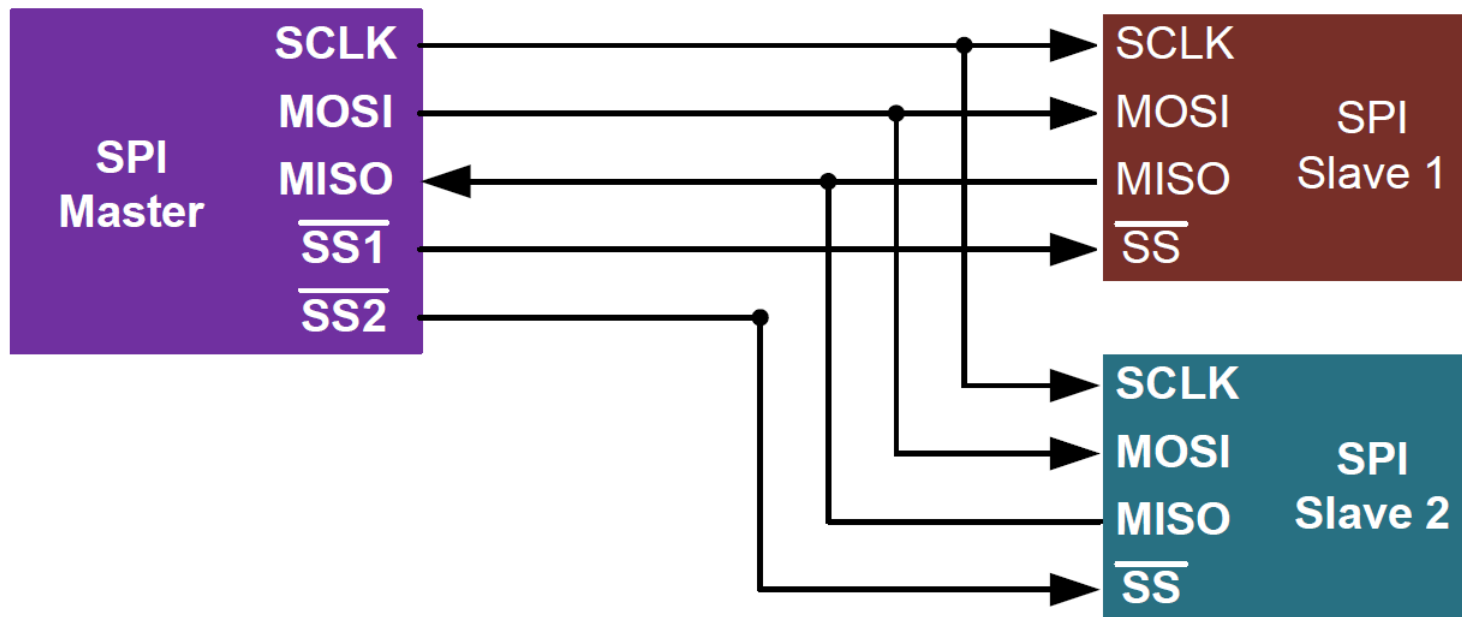
A Brief Introduction to SPI

SPI: Serial Peripheral Interface



Serial Peripheral Interface (SPI)

- Synchronous full-duplex communication
- Can have multiple slave (secondary) devices
- No flow control or acknowledgment
- Slave (secondary) cannot communicate with slave directly.



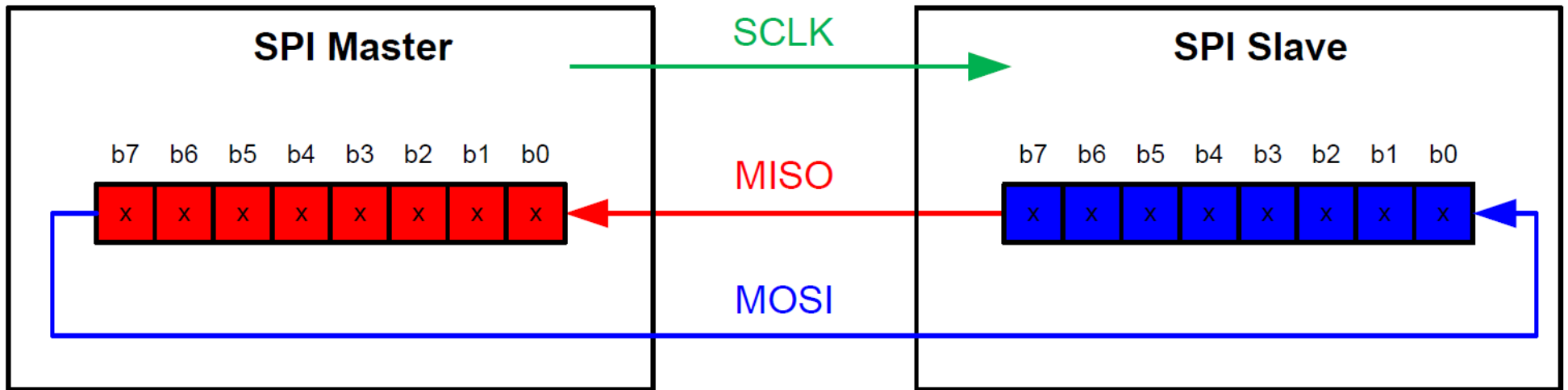
SCLK: serial clock

MOSI: master out slave in

SS: slave select (active low)

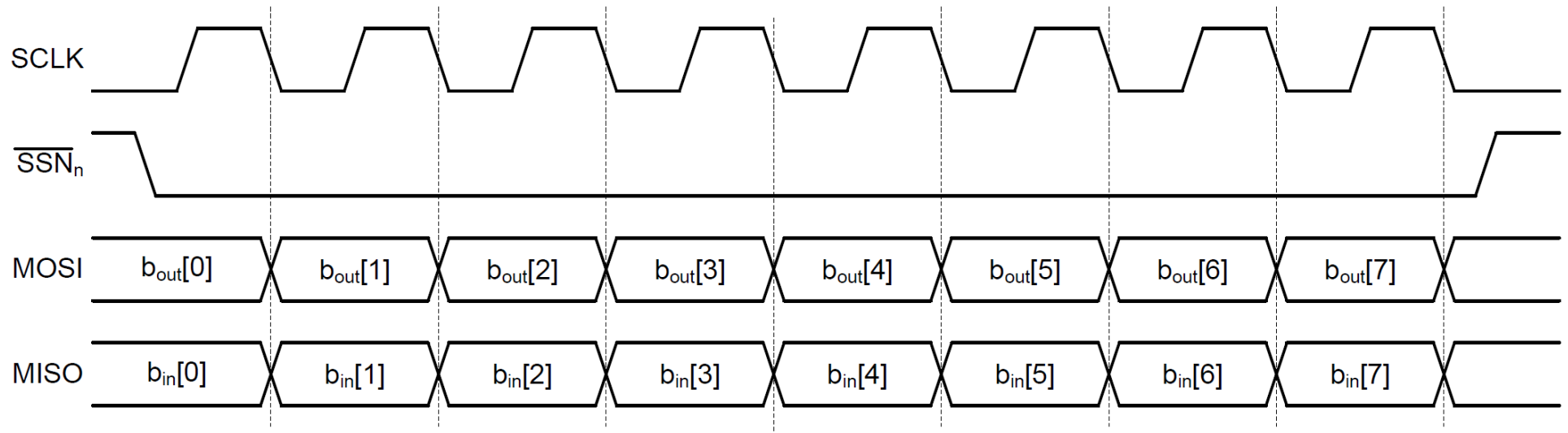
MISO: master in slave out

Data Exchange

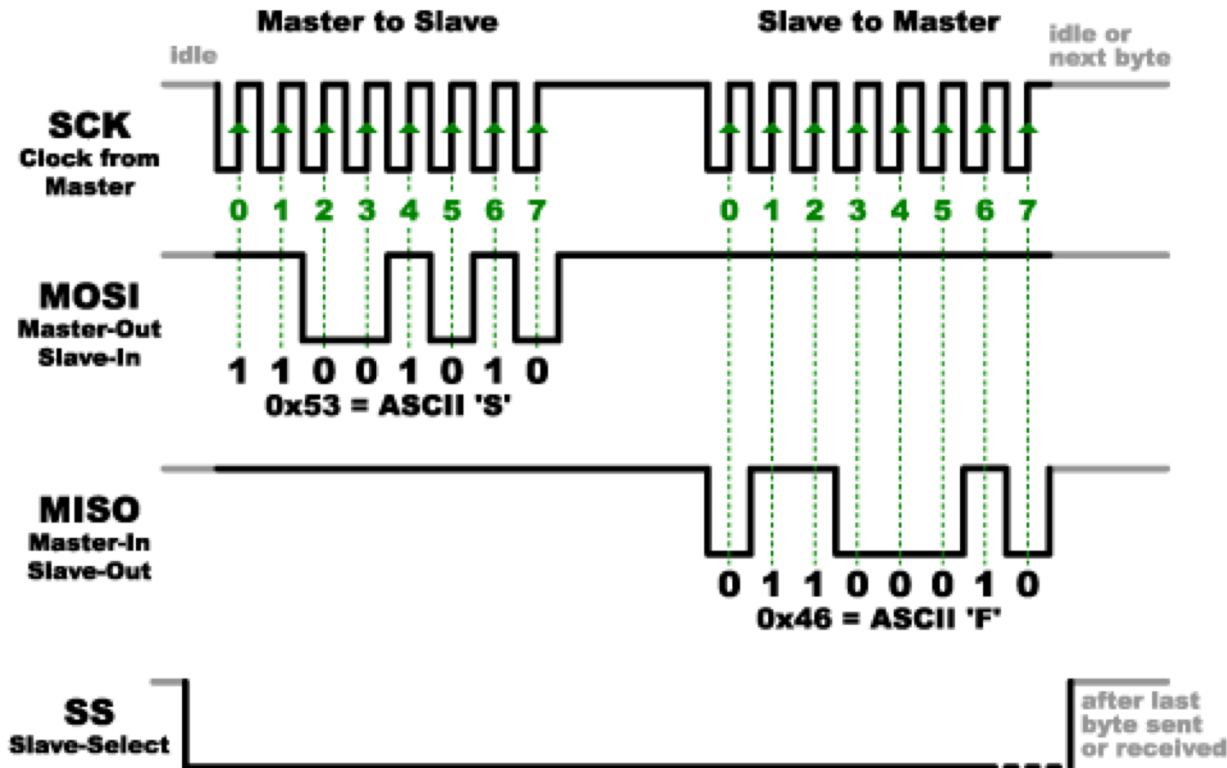
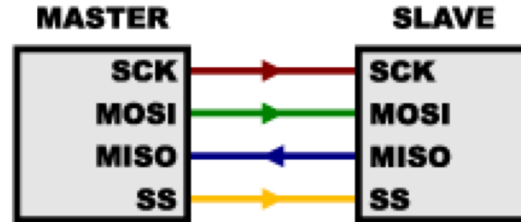


- Master has to provide clock to slave
- **Synchronous exchange**: for each clock pulse, a bit is shifted out and another bit is shifted in at the same time. This process stops when all bits are swapped.
- Only **master can start the data transfer**

Clock



Read and Write



Multiple Chip Selects

