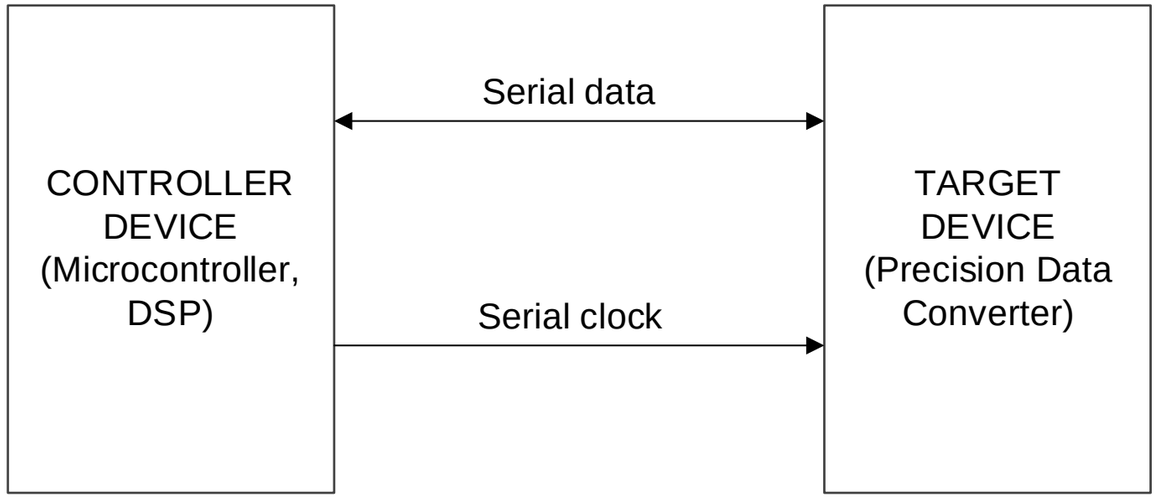
Basics of I2C: The I2C Protocol

TIPL 6101
TI Precision Labs – Digital Communications

Prepared by Joseph Wu

Presented by Alex Smith

I2C Introduction



I2C Introduction
I2C – Inter Integrated Circuit
Created by Philips Semiconductor in 1982
No license needed since 2006, many I2C compatible device manufacturers
Widely used protocol

I2C Communication Modes

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra-Fast Mode	5 Mbps

I2C Communication Modes

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra-Fast Mode	5 Mbps

Similar in implementation, with different timing requirements

I2C Communication Modes

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra-Fast Mode	5 Mbps

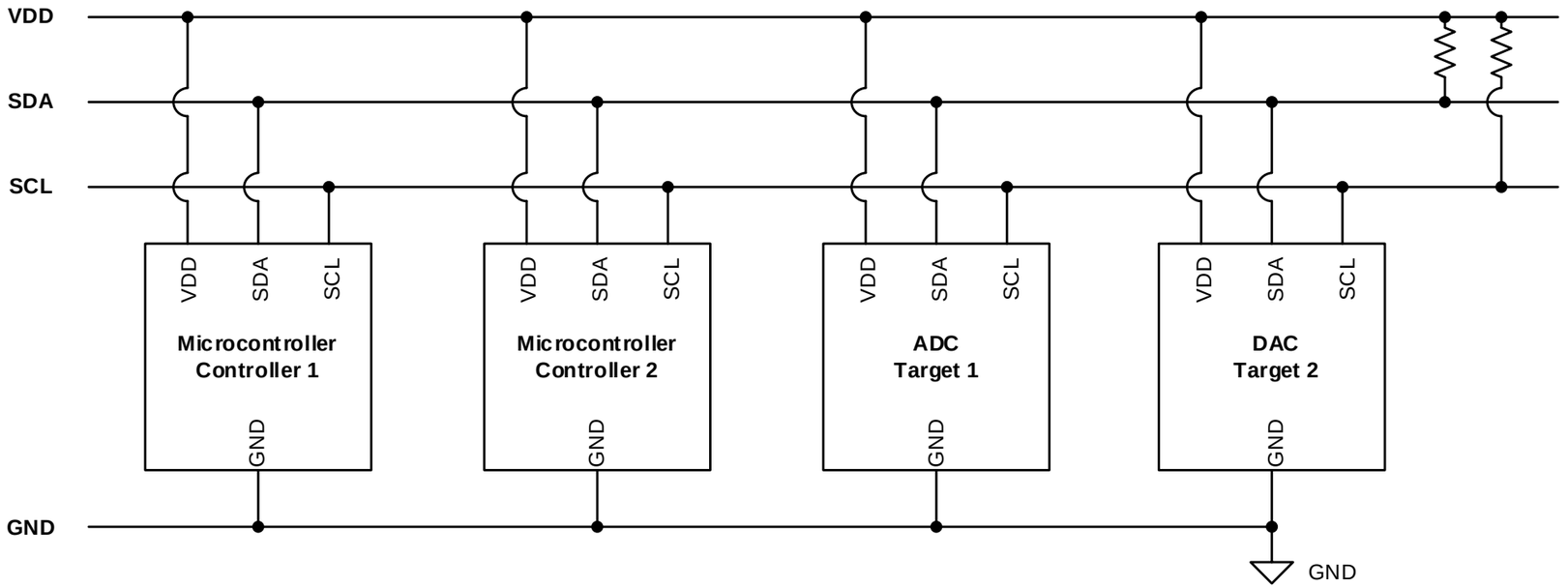
Requires controller code for high speed transfer

I2C Communication Modes

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra-Fast Mode	5 Mbps

Write-only, omits some standard I2C features

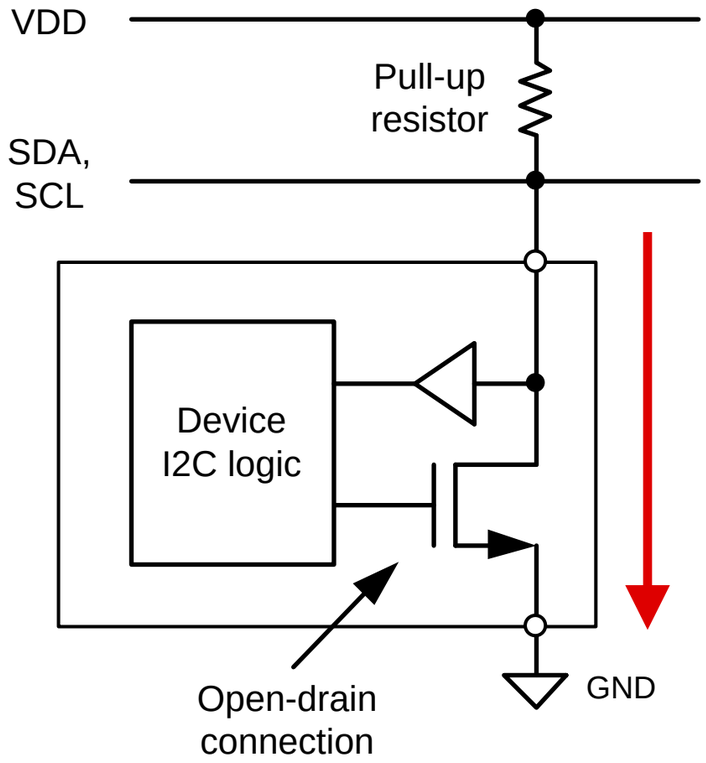
I2C Physical Layer



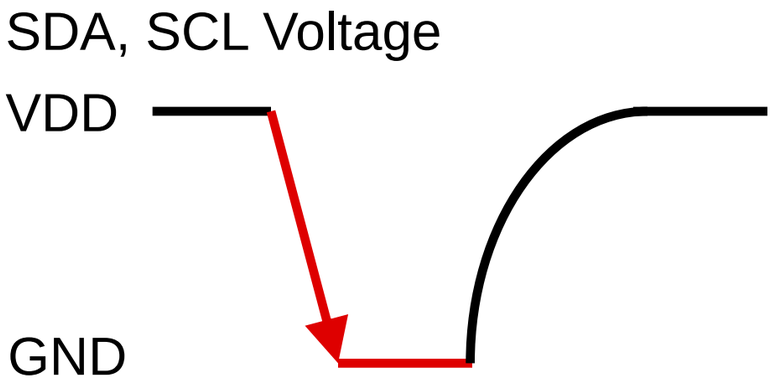
I2C System Features

- Only two communication lines for all devices on the bus (SDA, SCL)
- Bi-directional communication, half duplex
- Allows for multiple controllers and multiple targets
- Requires pull-up resistors on both SDA and SCL

I2C Physical Layer – Open-Drain Connection

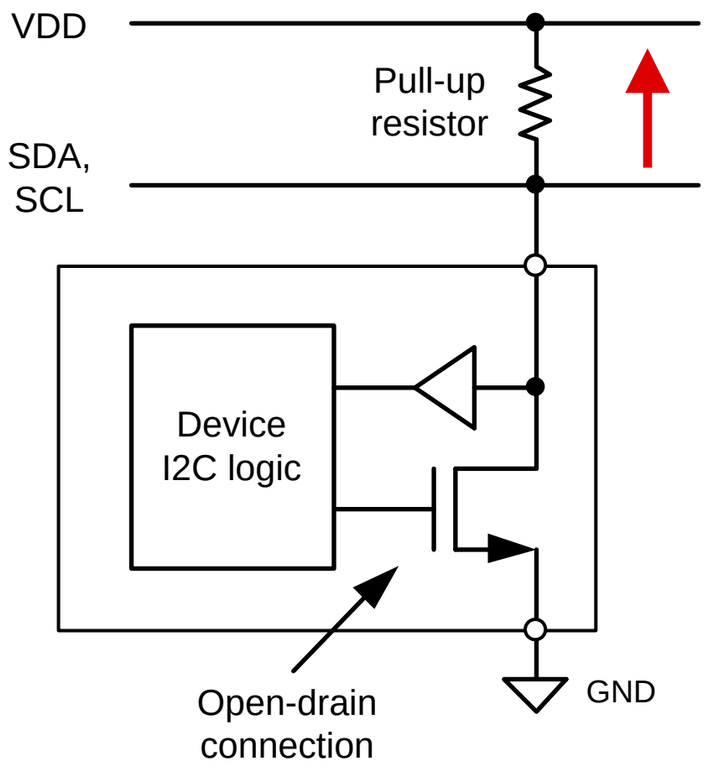


When NMOS turns ON,
SDA or SCL is pulled low

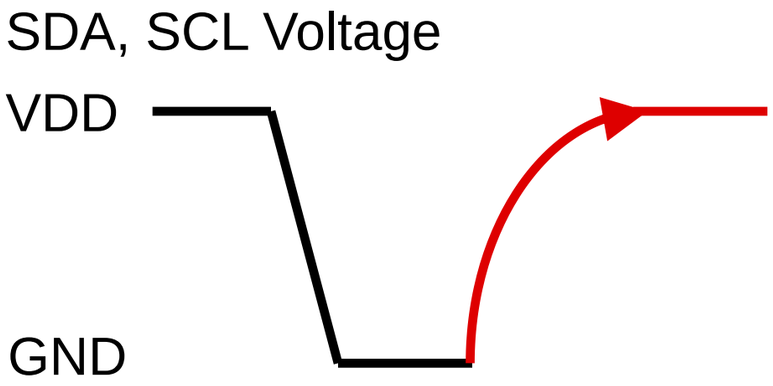


Quick transition from high to low as NMOS pulls charge from any bus capacitance from SDA, SCL

I2C Physical Layer – Open-Drain Connection



When NMOS turns OFF,
SDA or SCL is released and returns high from the pullup resistor

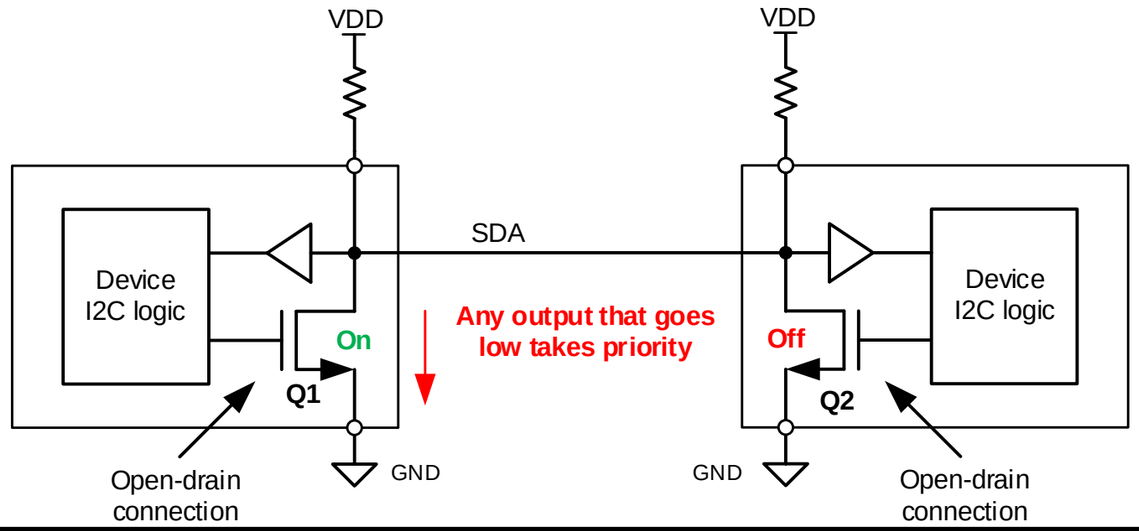


Exponential rise depends on capacitance on SDA or SCL and pullup resistor size

Low resistance: faster communication, more power
High resistance: slower communication, less power

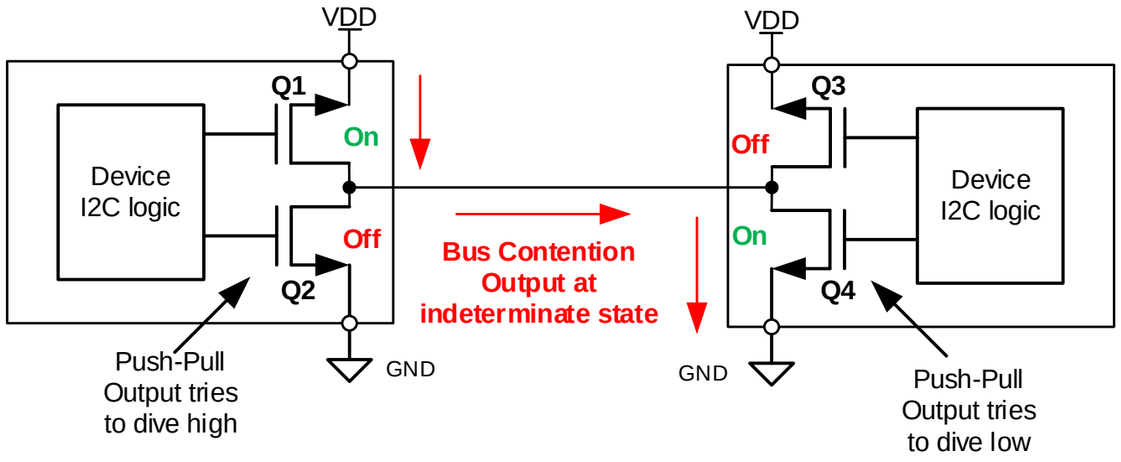
I2C Physical Layer – Open Collector vs Push-Pull

Open Drain



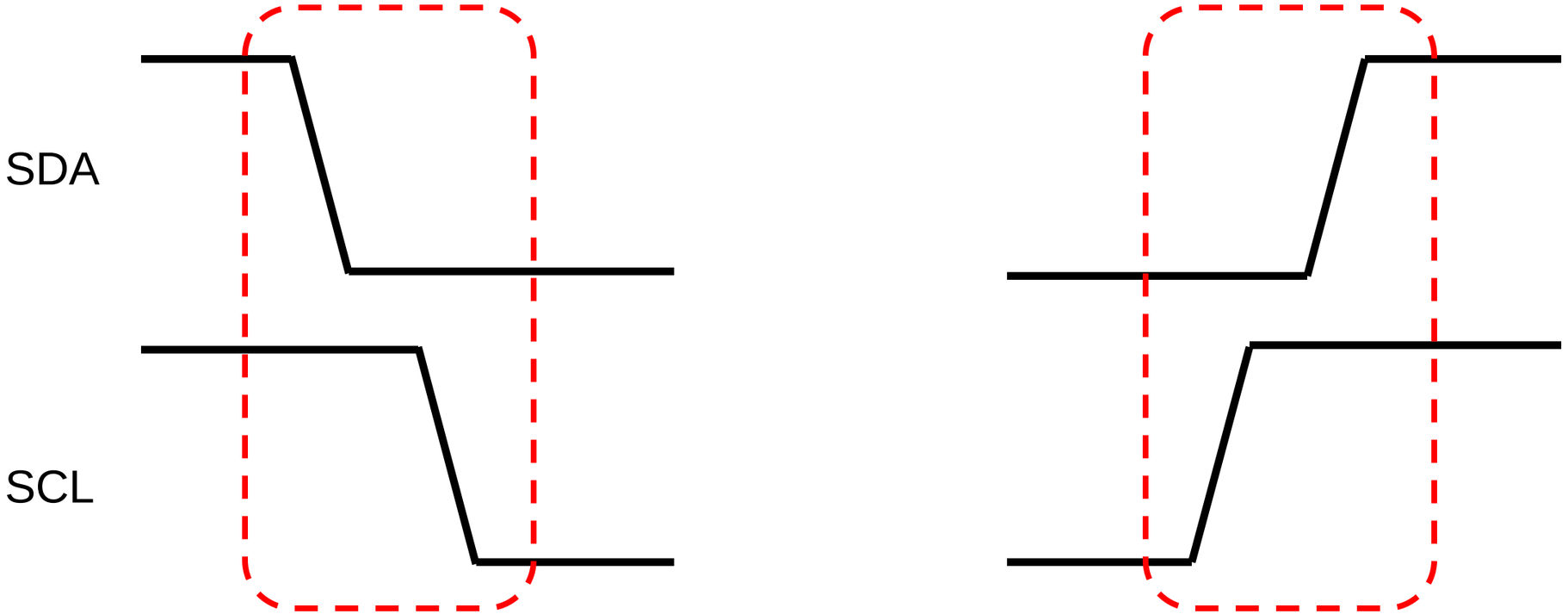
- Open drain output can connect together
- Any output that goes low will pull the bus low
- This type of connection is called a “wired-OR”

Push-Pull



- Open drain output cannot connect together
- Connecting outputs together can cause a bus contention where the output state is indeterminate

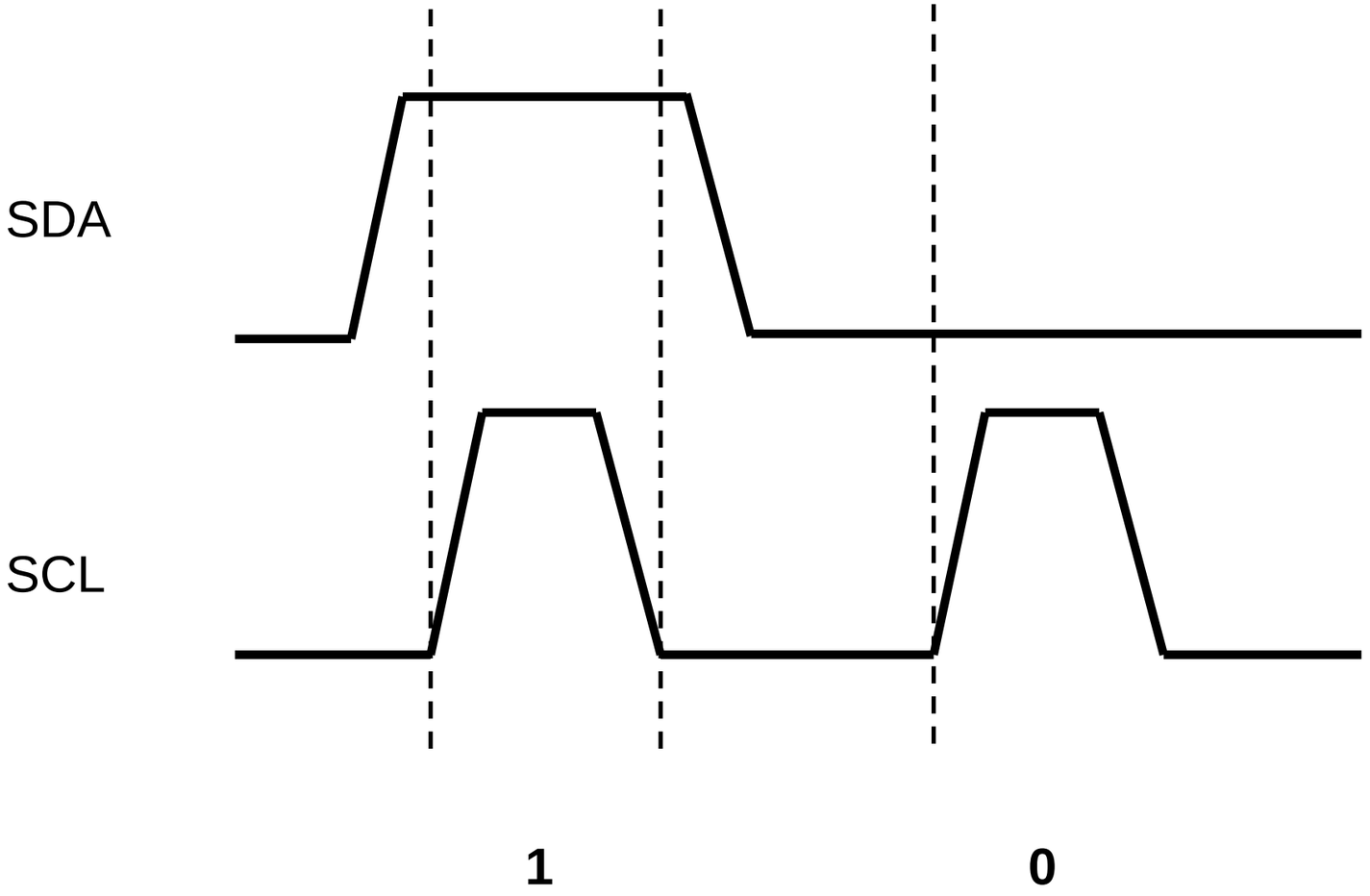
I2C Protocol – START and STOP



I2C START
A controller device claims the I2C bus for communication with a target device

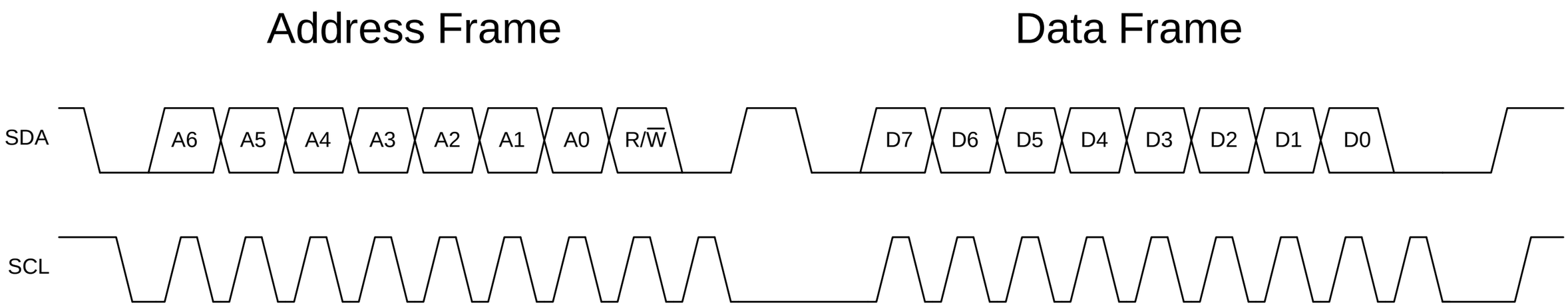
I2C STOP
A controller device completes communication with a target device and releases the I2C bus

I2C Protocol – Logical Ones and Zeros

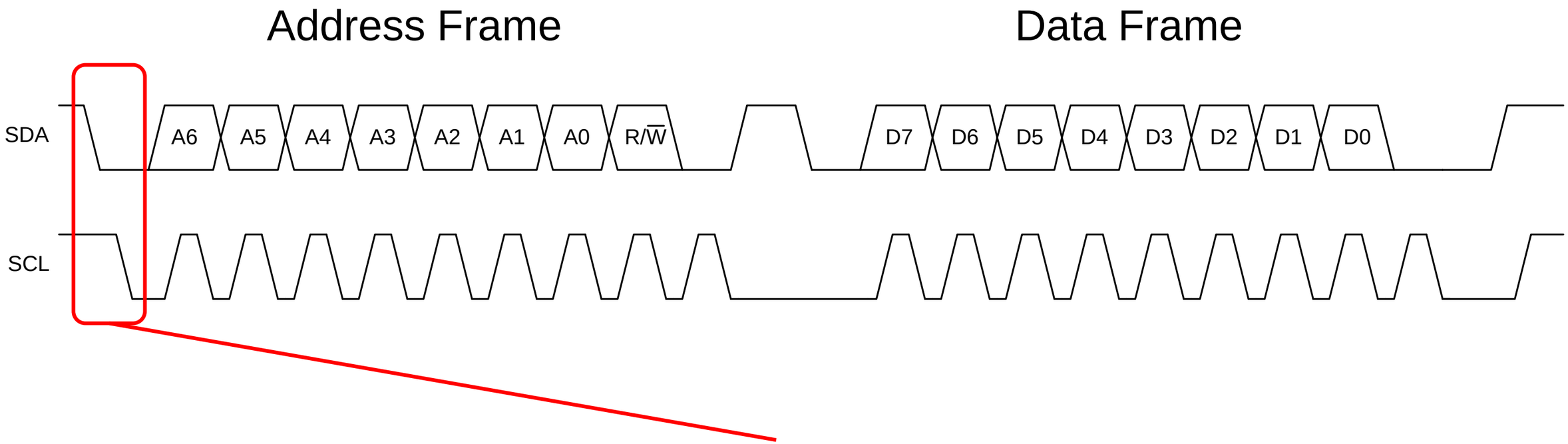


I2C Logical Bits
SDA is the data line, SCL is serial clock
SDA only transitions when SCL is low (except during START and STOP)
SDA is high when SCL pulses is a logical one
SDA is low when SCLK pulses is a logical zero

I2C Protocol – Timing Diagram

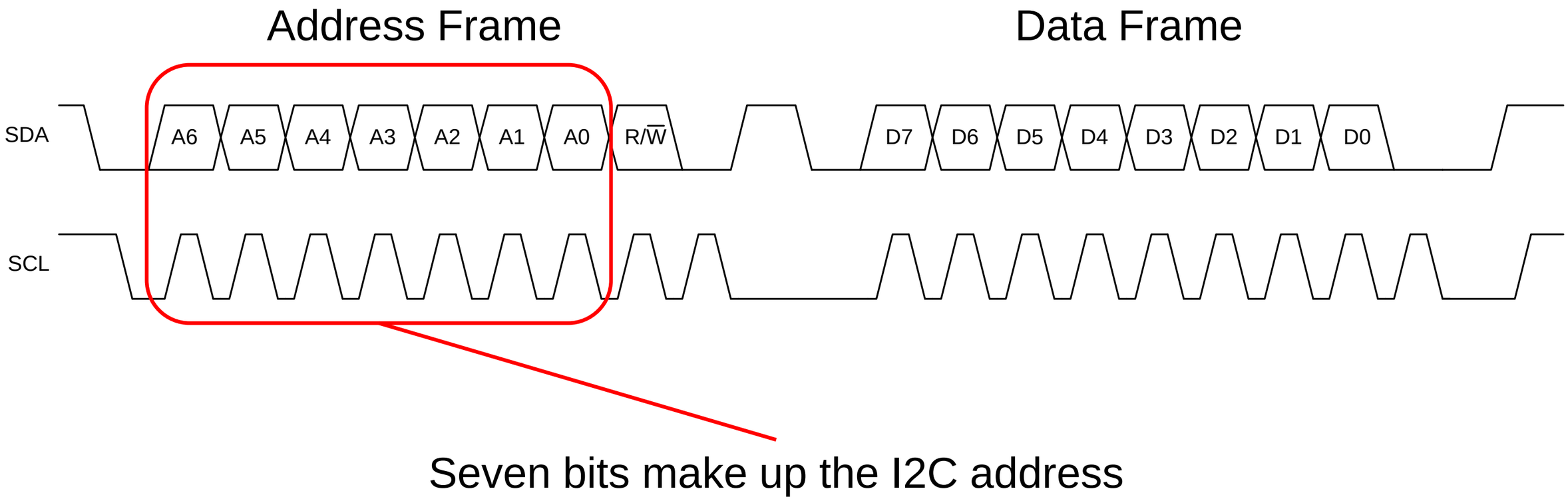


I2C Protocol – Timing Diagram

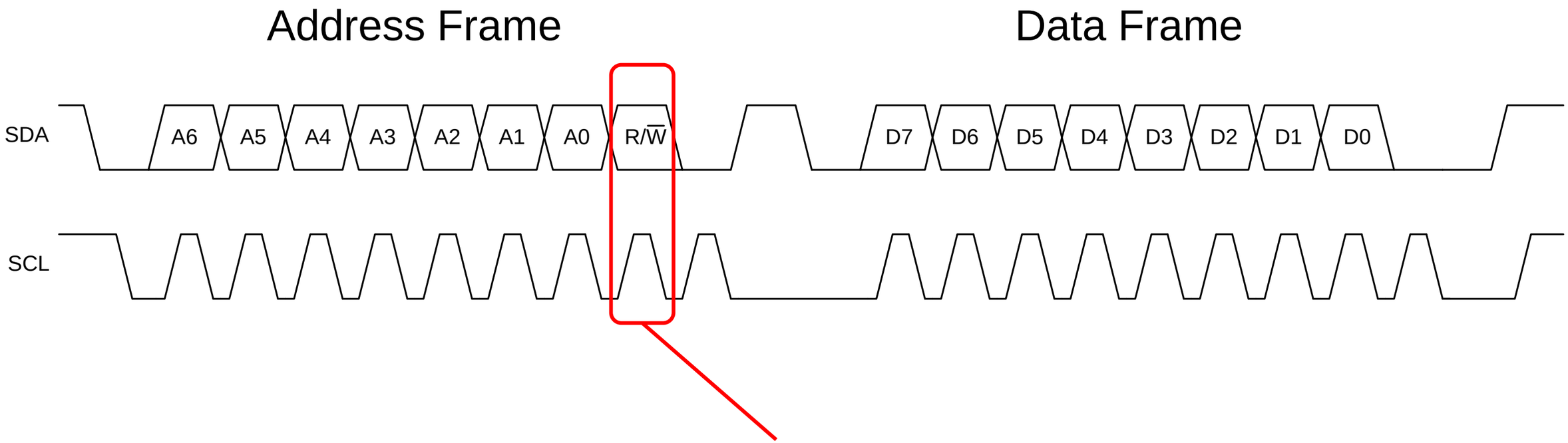


An I2C START condition comes from the controller and sends SDA low before SCL is sent low to claim the bus

I2C Protocol – Timing Diagram

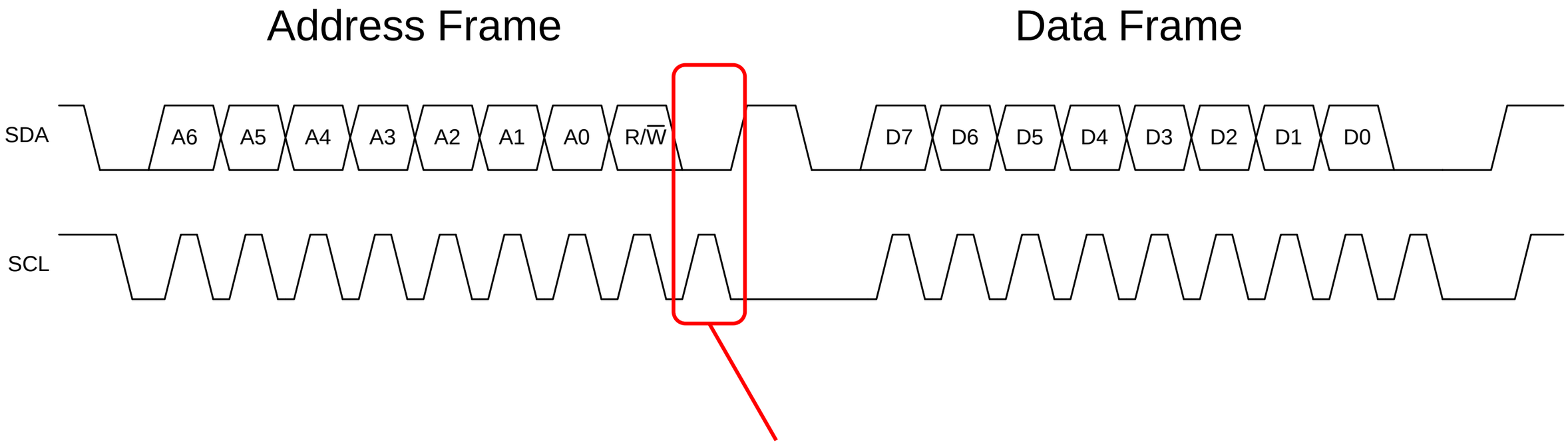


I2C Protocol – Timing Diagram



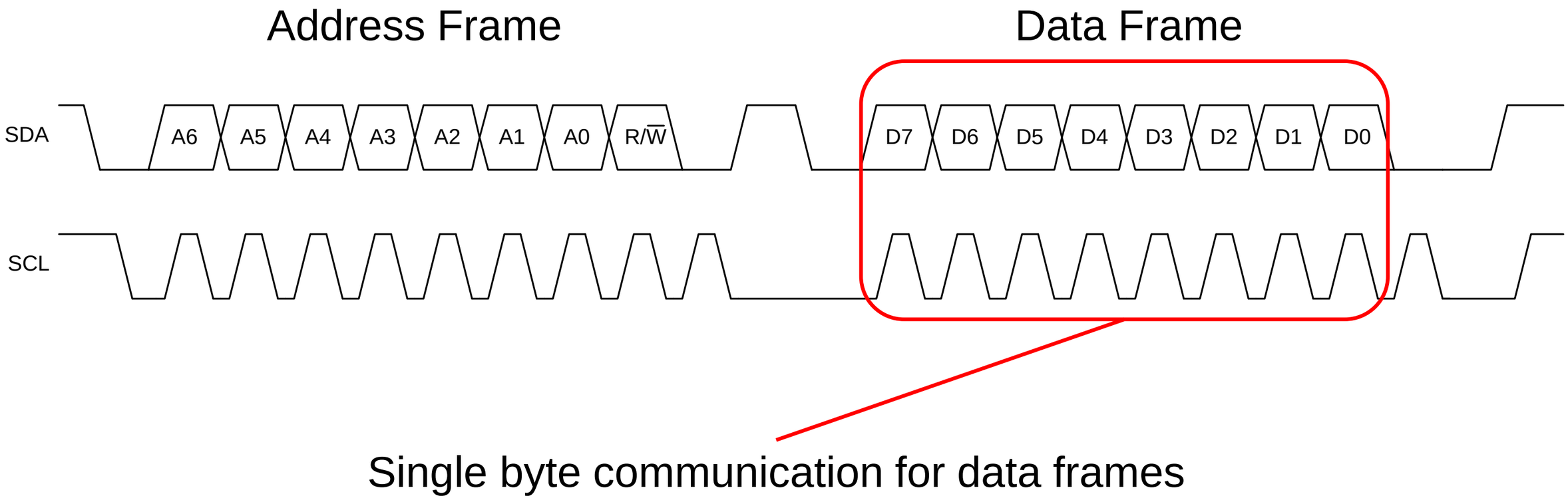
R/W bit indicates the direction of communication
1: Controller wants to read from the target device
0: Controller wants to write to the target device

I2C Protocol – Timing Diagram



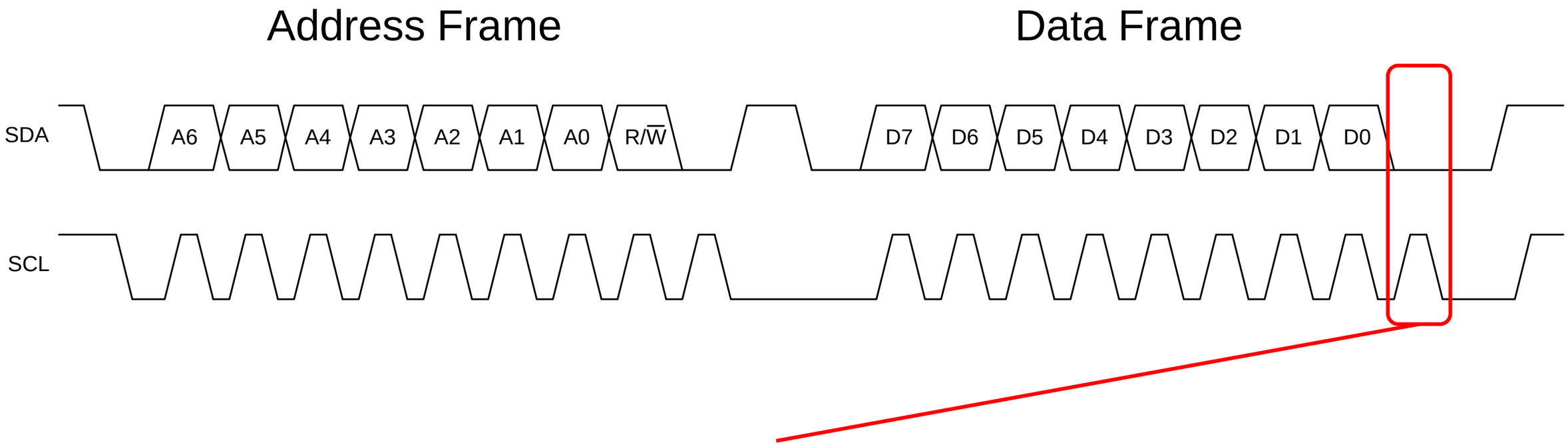
SDA is pulled down as an ACKT (acknowledge, target)
After the address byte, the target device ACKs the communication

I2C Protocol – Timing Diagram



Single byte communication for data frames

I2C Protocol – Timing Diagram

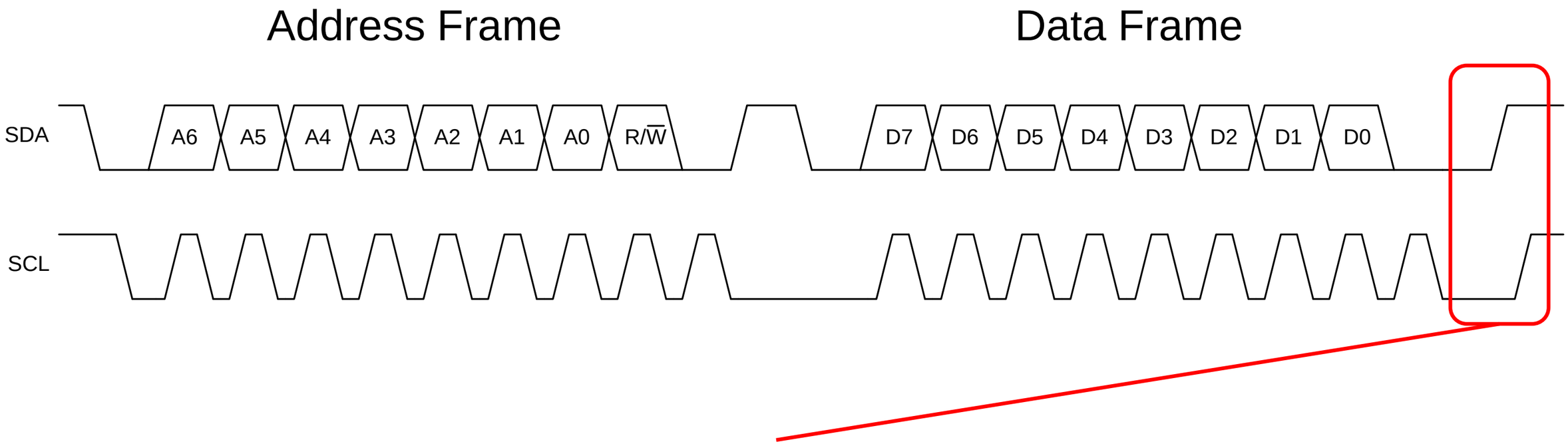


ACK follows each data frame

Write to the target – ACKT comes from the target device

Read from the target – ACKC comes from the controller device

I2C Protocol – Timing Diagram



I2C STOP condition comes from the controller and sends SDA high before SCL is sent high to release the bus

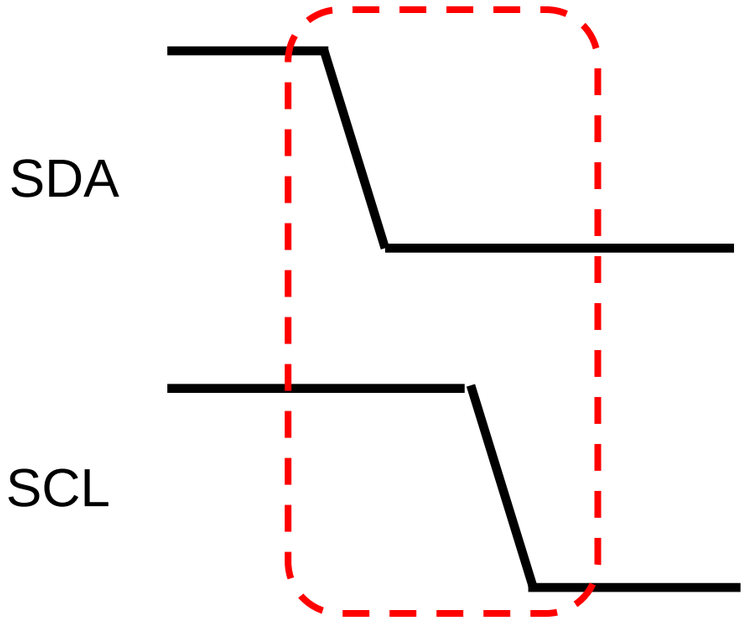
Thanks for your time!
Please try the quiz.

Quiz: Basics of I2C: The I2C Protocol

1. Before the address frame of I2C communication, what actions make up the START condition?
   a. The controller device sets the SDA low, and then sets the SCL low
   b. The controller device sets the SCL low, and then sets the SDA low
   c. The controller device sets the SCL low, and the target device pulls the SDA low as an ACK

Quiz: Basics of I2C: The I2C Protocol

1. Before the address frame of I2C communication, what actions make up the START condition?
   a. The controller device sets the SDA low, and then sets the SCL low
   b. The controller device sets the SCL low, and then sets the SDA low
   c. The controller device sets the SCL low, and the target device pulls the SDA low as an ACK

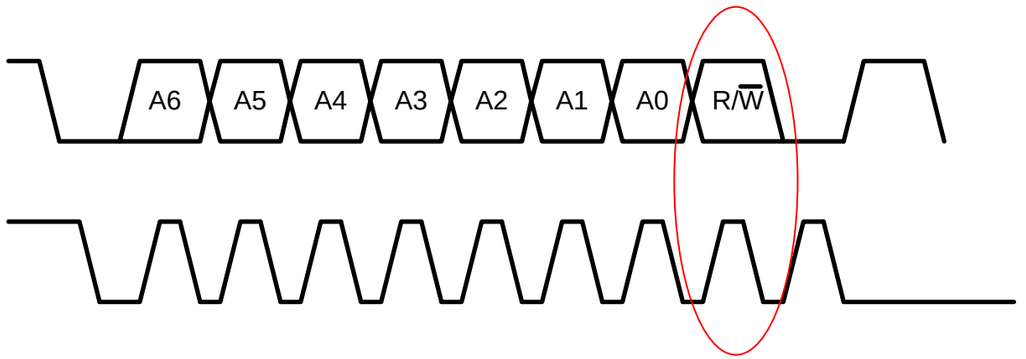


Quiz: Basics of I2C: The I2C Protocol

2. In the address frame, after the controller device sends the 7 bit address, what is the next part of the I2C protocol sent?
   a. The target device sends the ACK to acknowledge the communication coming from the controller device
   b. The controller device sends the R/W bit to indicate if it wants to read from or write to the target device
   c. The controller device send a STOP condition before sending the next data

Quiz: Basics of I2C: The I2C Protocol

2. In the address frame, after the controller device sends the 7 bit address, what is the next part of the I2C protocol sent?
   a. The target device sends the ACK to acknowledge the communication coming from the controller device
   b. The controller device sends the R/W bit to indicate if it wants to read from or write to the target device
   c. The controller device send a STOP condition before sending the next data

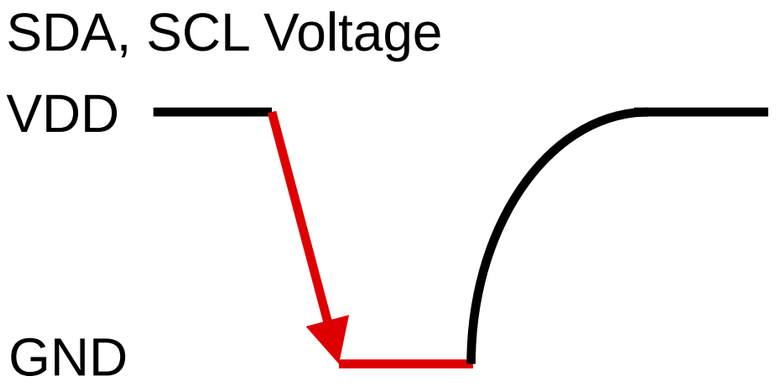


Quiz: Basics of I2C: The I2C Protocol

3. Because of the NMOS open-drain connection to SDA and SCL, which part of the communication waveform is faster?
   a. The rise time of SDA and SCL
   b. The fall time of SDA and SCL
   c. The rise time and fall time of SDA and SCL are the same

Quiz: Basics of I2C: The I2C Protocol

3. Because of the NMOS open-drain connection to SDA and SCL, which part of the communication waveform is faster?
   a. The rise time of SDA and SCL
   b. The fall time of SDA and SCL
   c. The rise time and fall time of SDA and SCL are the same



Open-drain connections are actively pulled down instead and are faster than a resistive pull up

Quiz: Basics of I2C: The I2C Protocol

4. What is the benefit of having an open-drain connection over push-pull outputs for I2C?
   a. High speed drive for the bus outputs
   b. Reduction of bus capacitance
   c. Prevents destructive current draw during bus contention when outputs are tied together

Quiz: Basics of I2C: The I2C Protocol

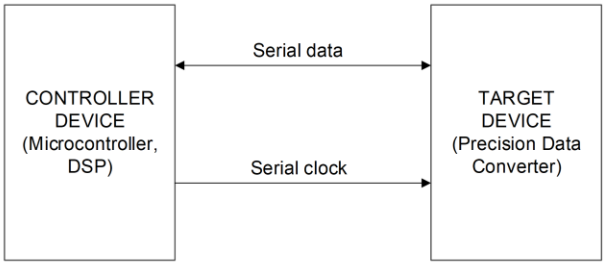
4. What is the benefit of having an open-drain connection over push-pull outputs for I2C?
   a. High speed drive for the bus outputs
   b. Reduction of bus capacitance
   c. Prevents destructive current draw during bus contention when outputs are tied together

Push-Pull outputs may pull a large current when the outputs are tied together and there is bus contention

Thanks for your time!

Hello, and welcome to our in-depth look at communications with precision data converters. In this video, we describe the basics of I2C communication. We'll discuss the digital lines and the structure of the I2C protocol. Finally, we'll give an example of how data is transmitted to and received from a precision data converter using I2C. By the end of the presentation, you should understand the basics of how I2C is implemented, the structure of the I2C protocol, and how to I2C is used to read from and write to different peripheral devices.

I2C Introduction



I2C Introduction
I2C – Inter Integrated Circuit
Created by Philips Semiconductor in 1982
No license needed since 2006, many I2C compatible device manufacturers
Widely used protocol

I2C, often called I two C, stands for the Inter-Integrated Circuit protocol. I2C was invented in 1982 by Philips Semiconductor (now NXP Semiconductor) as a low speed communication protocol for connecting microprocessor controller devices with lower-speed peripheral target devices. Since 2006, implementing the I2C protocol does not require a license, and many semiconductor device companies, including TI, have introduced I2C compatible devices.

I2C is a widely used protocol for many reasons. It requires only two lines for communications. Like other serial communication protocols, there is a serial data line and a serial clock line. I2C can connect to multiple devices on the bus with only the two lines. The controller device can communicate with any target device through a I2C address sent through the serial data line. I2C is simple and economical for device manufacturers to implement.

I2C Communication Modes

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra-Fast Mode	5 Mbps

I2C has several speed modes starting with the standard mode, which is a serial protocol that goes up to 100 kilobits per second. This is followed by the Fast mode which tops out at 400 kilobits per second. Both of these protocols are widely supported and may be used by the controller if the bus capacitance and drive capability allow for the faster speed.

The Fast mode plus allows for communication as high as 1 megabit per second. To achieve this speed, drivers may require extra strength to comply with faster rise and fall times.

I2C Communication Modes

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra-Fast Mode	5 Mbps

Similar in implementation, with different timing requirements

These three modes are relatively similar, using a communication structure that is the same. However, they all have different timing specifications for each of the modes and hardware implementation of the I2C in the devices are different to accommodate the different speeds.

I2C Communication Modes

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra-Fast Mode	5 Mbps

Requires controller code for high speed transfer

I2C also has two other modes for higher data rates.

High-Speed Mode has a data rate to 3.4 megabits per second. In this mode, the controller device must first use a controller code to allow for high-speed data transfer. This enables High-speed mode in the target device. This mode may also require an active pull-up to drive the communication lines at a higher data rate.

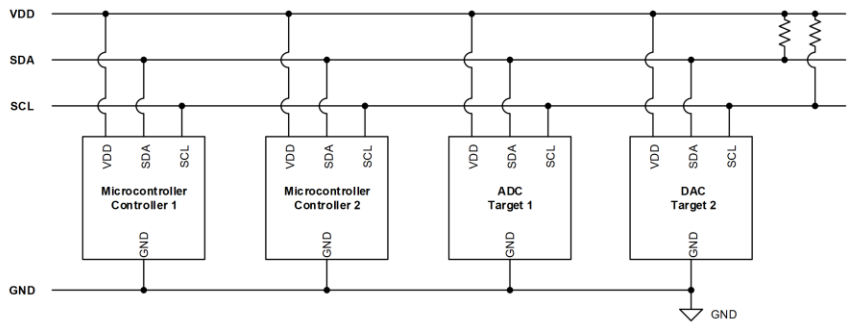
I2C Communication Modes

I2C Mode	Speed
Standard Mode	100 kbps
Fast Mode	400 kbps
Fast Mode Plus	1 Mbps
High Speed Mode	3.4 Mbps
Ultra-Fast Mode	5 Mbps

Write-only, omits some standard I2C features

Ultra-Fast is the fastest mode of operation and transfers data up to 5 megabits per second. This mode is write-only and omits some I2C features in the communication protocol.

I2C Physical Layer



I2C System Features
Only two communication lines for all devices on the bus (SDA, SCL)
Bi-directional communication, half duplex
Allows for multiple controllers and multiple targets
Requires pull-up resistors on both SDA and SCL

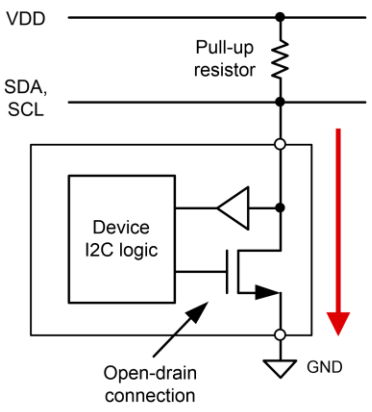
One of the reasons that I2C is a common protocol is because there are only two lines used for communications. The first line is SCL, which is a serial clock primarily controlled by the controller. SCL is used to synchronously clock data in and out of the target device. The second line is SDA, which is the serial data line. SDA is used to transmit data between the controller devices and target devices. In comparison, the serial peripheral interface or SPI protocol requires four lines for communication. In addition to the serial clock, the SPI chip select line selects the device for communication, and there are two data lines, used for input and output from the target device.

For I2C, the controller device controls the serial clock SCL, the SDA is used to send data in both directions. The SDA is bi-directional, which means that the controller devices and target devices can both send data on the line. For example, the controller device can send configuration data to the target device, and the target device can send conversion data back to the controller device. Communication is half duplex where only a controller or a target device is sending data on the bus at a time.

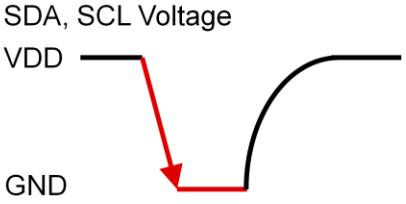
An I2C controller device starts and stops communication, which removes the potential problem of bus contention. Also, communication with a target device is sent through a unique address on the bus. This allows for both multiple controller and multiple target devices on the I2C bus.

The SDA and SCL lines have an open drain connection to all devices on the bus. This requires a pull-up resistor to a common voltage supply.

I2C Physical Layer – Open-Drain Connection



When NMOS turns ON,
SDA or SCL is pulled low

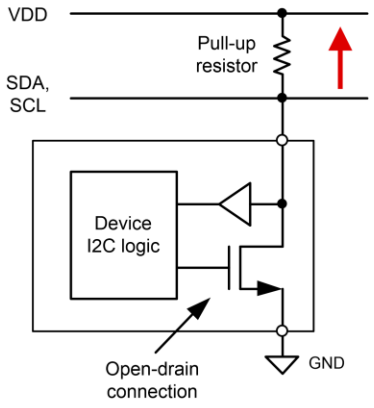


Quick transition from high to low as NMOS pulls charge from any bus capacitance from SDA, SCL

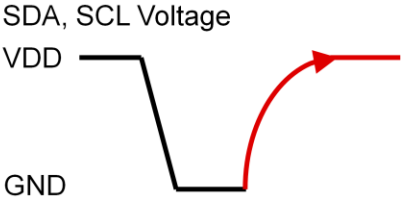
The open-drain connection are used on both SDA and SCL lines and connect to an NMOS transistor. This diagram shows an I2C device connected to an SDA or SCL line with a pull-up resistor to VDD. This open-drain connection controls the I2C communication line and pulls it low or releases it high.

To set the voltage level of the SDA or SCL line, the NMOS is set ON or OFF. When the NMOS is ON, the device pulls current through the resistor to ground. This pulls the line low. Typically the transition from high to low for I2C is fast as the NMOS pulls down on SDA and SCL. The speed of the transition is determined by the NMOS drive strength and the bus capacitance on SDA or SCL.

I2C Physical Layer – Open-Drain Connection



When NMOS turns OFF,
SDA or SCL is released and returns high from the pullup resistor



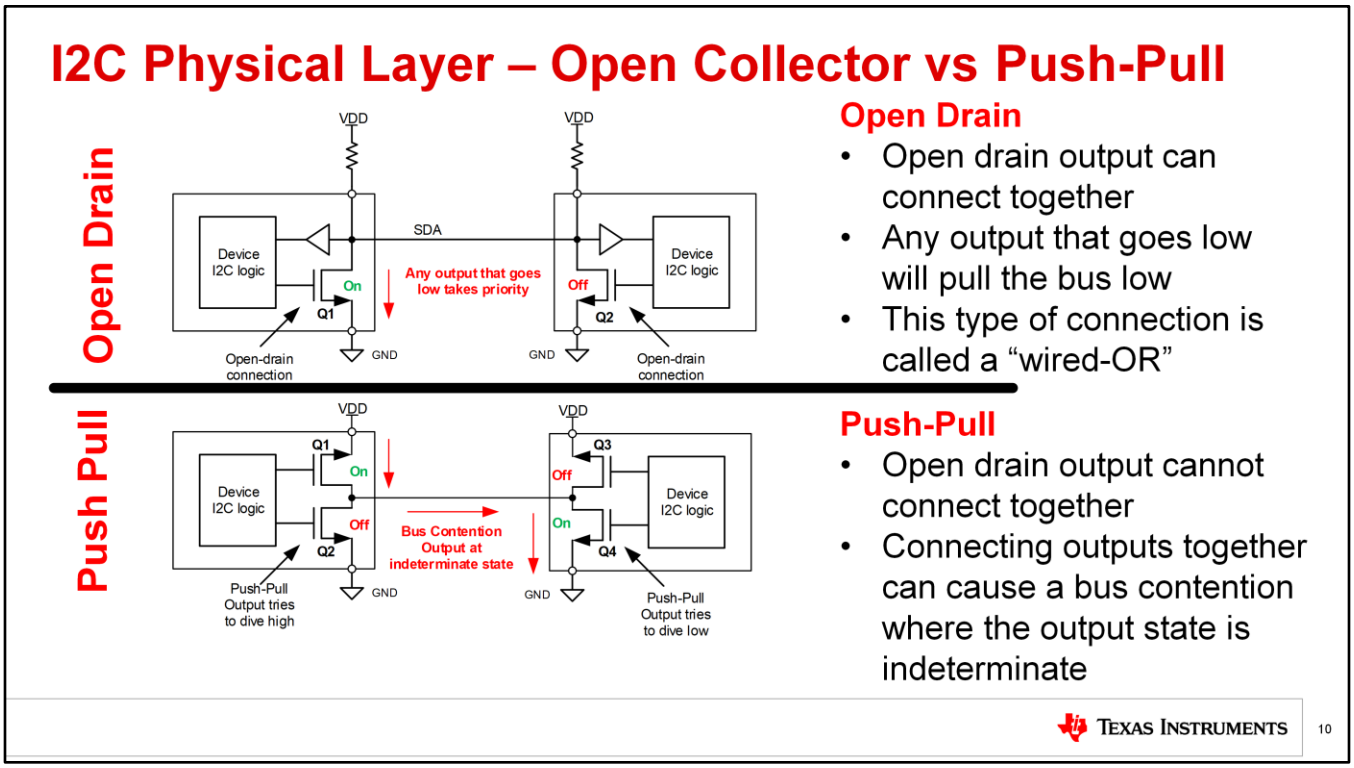
Exponential rise depends on capacitance on SDA or SCL and pullup resistor size

Low resistance: faster communication, more power
High resistance: slower communication, less power

When the NMOS is OFF, the device stops pulling current, and the pull-up resistor pulls the SDA or SCL line to VDD. This pulls the line high. Through control of this open-drain connection, both SDA and SCL can be set high and low, enabling the I2C communication. Because of capacitance on the I2C communication line, the SDA or SCL line discharge with an exponential RC time constant depending on the size of the pullup resistor and capacitance on the I2C bus.

Typically, pull-up resistors are set between 1 kiloOhm to 10 kiloOhms. The bus speed may help determine the size of the resistance. With higher resistive values, the I2C bus may pull up the line slower and limit the bus speed. Capacitance on the bus lines also has an impact on communication. Higher capacitance limits the speed of I2C communication, the number of devices, and the physical distance between devices on the bus.

A smaller pullup resistor has a faster rise time, but require more power for communication. A larger pullup resistor has a slower rise time leading to slower communication, but requires less power.

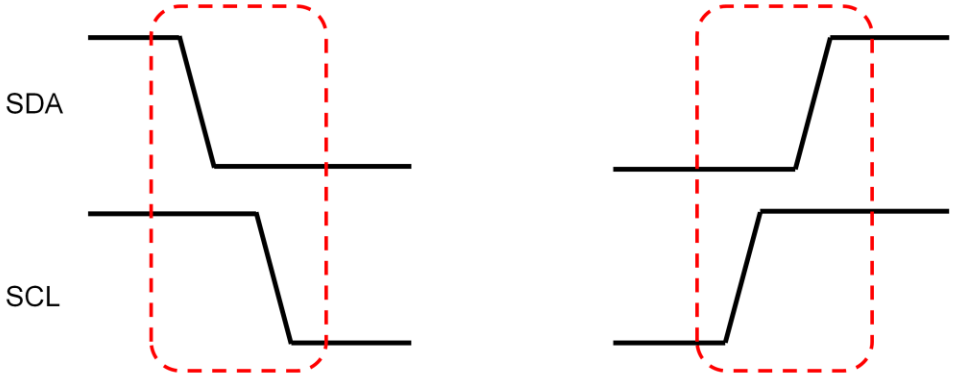


One of the benefits of I2C using an open collector is that bus contention will not put the bus into a destructive state. With an open drain output many devices can be connected together. For any output on that connection, if either output pulls the line low, the line will be low. This kind of connection is called a "wired-OR". The output is the logical OR of the all the outputs when tied together.

If the outputs were a push-pull type, they could not be tied together without the possibility of a destructive state. A push-pull output has complementary NMOS and PMOS transistors that drive the output high or low. Tied together, if one output is high and another output is low, this bus contention would have an undetermined state,

possibly settling at the mid supply point. Additionally, one device has NMOS conducting current and another device has a PMOS conducting current. This would source current from VDD to GND through a very low impedance path, conducting as much current as the transistors would allow. This could be a significant amount of current, potentially damaging the devices.

I2C Protocol – START and STOP



I2C START
A controller device claims the I2C bus for communication with a target device

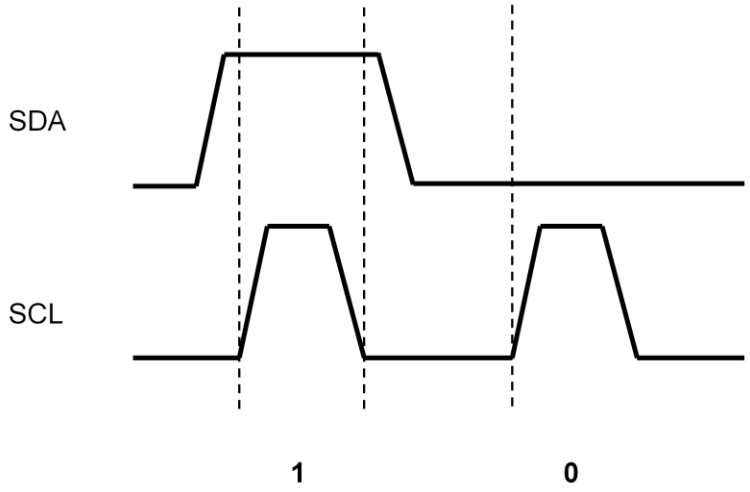
I2C STOP
A controller device completes communication with a target device and releases the I2C bus

Texas Instruments 11

I2C communication is initiated from the controller device with an I2C start condition. If the bus is open, an I2C controller may claim the bus for communication by sending an I2C START condition. To do this, the controller device first pulls the SDA low and then pulls the SCL low. This sequence indicates that the controller device is claiming the I2C bus for communication, forcing other controller devices on the bus to hold their communication.

When the controller device has completed it's communication, it releases the SCL high and then releases the SDA high. This indicates an I2C STOP condition. This releases the bus to allow other masters to communicate or to allow for the same controller to communicate with another device.

I2C Protocol – Logical Ones and Zeros



I2C Logical Bits
SDA is the data line, SCL is serial clock
SDA only transitions when SCL is low (except during START and STOP)
SDA is high when SCL pulses is a logical one
SDA is low when SCLK pulses is a logical zero

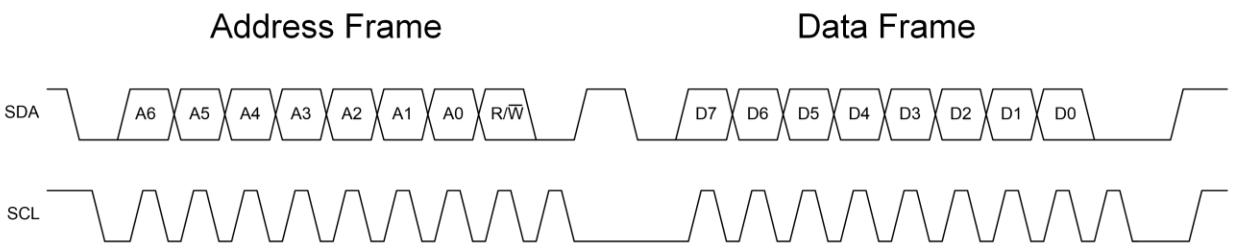
I2C uses a sequence of ones and zeros for its serial communication. SDA is used for the data bits while SCL is the serial clock that times the bit sequence.

A logical one is sent when the SDA releases the line, allowing the pull-up resistor to pull the line to a high level.

A logical zero is sent when SDA pulls down on the line, setting a low level near ground.

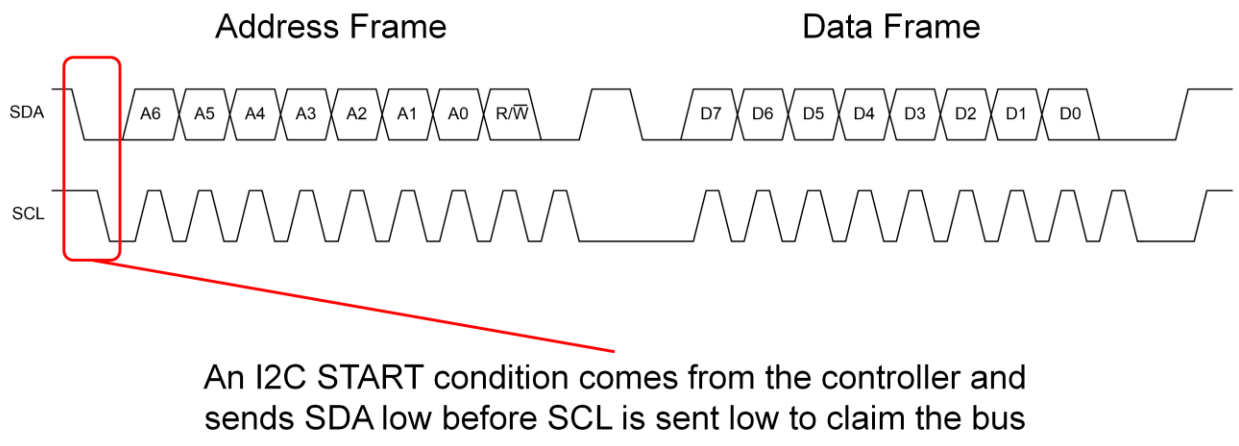
The ones and zeros are received when SCL is pulsed. For a valid bit, SDA does not change between a rising edge and the falling edge of SCK for that bit. If SDA changes between the rising and falling edges of the SCL, this may be interpreted as a STOP or START condition on the I2C bus.

I2C Protocol – Timing Diagram



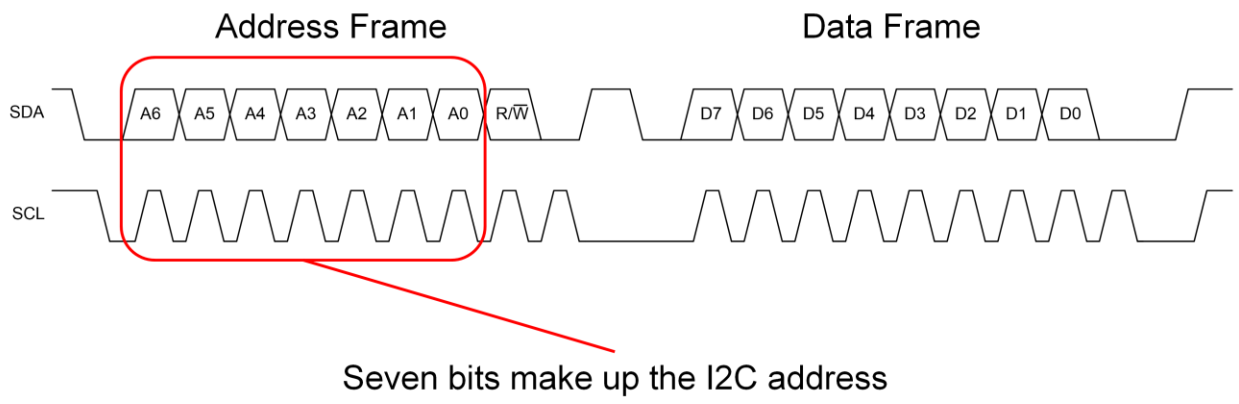
The I2C protocol is broken up into frames. Communication starts from the controller device with an address frame. The address frame is followed by one or more data frames consisting of one byte. Each frame also has an acknowledge bit to ensure that the target device or the controller device has received communication.

I2C Protocol – Timing Diagram



At the beginning of the address frame, the controller device initiates a START condition. First, the controller device pulls SDA low, and then it pulls SCL low for the START. This allows the controller device to claim the bus without contention from other controller devices on the bus.

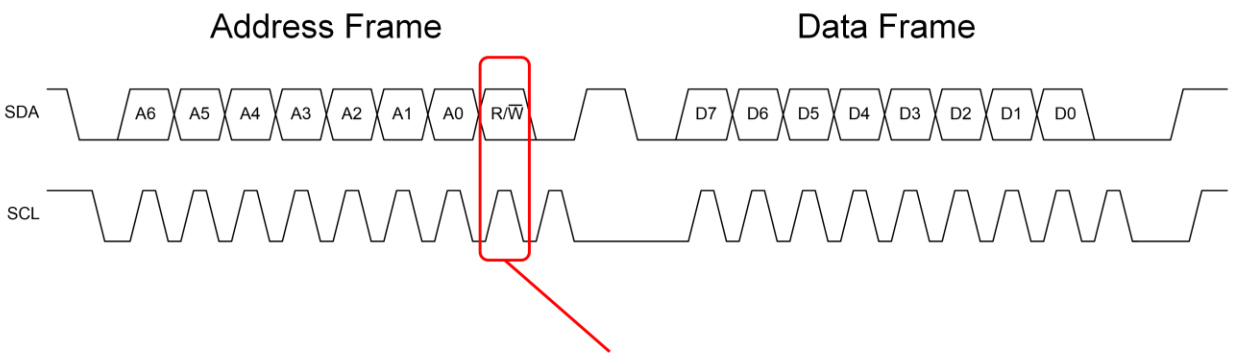
I2C Protocol – Timing Diagram



Each I2C target device has an associated I2C address. When the controller device wants to communicate with a particular device it uses its device address to send or receive data in the following I2C frames. The I2C address consists of 7 bits and devices on the I2C should have a unique address on the bus.

A 7 bit address would normally imply 2^7 (or 128) unique addresses. However, there are several reserved I2C addresses which limits the number of possible devices. The address is sent with the SDA as the data and SCL as the serial clock. With this information, you should be able to read through the I2C communication of a device and understand what is being sent back and forth from the controller device and the target device.

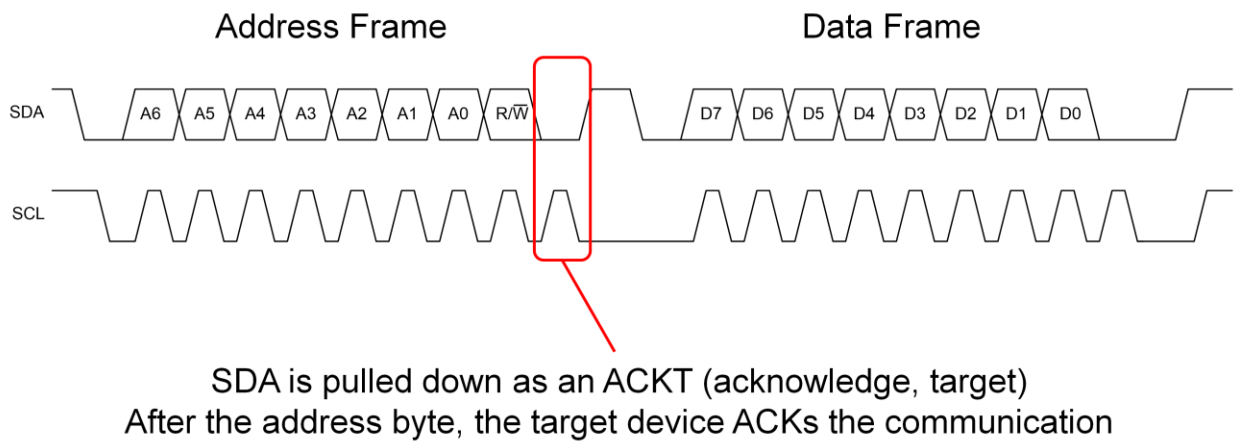
I2C Protocol – Timing Diagram



R/W bit indicates the direction of communication
1: Controller wants to read from the target device
0: Controller wants to write to the target device

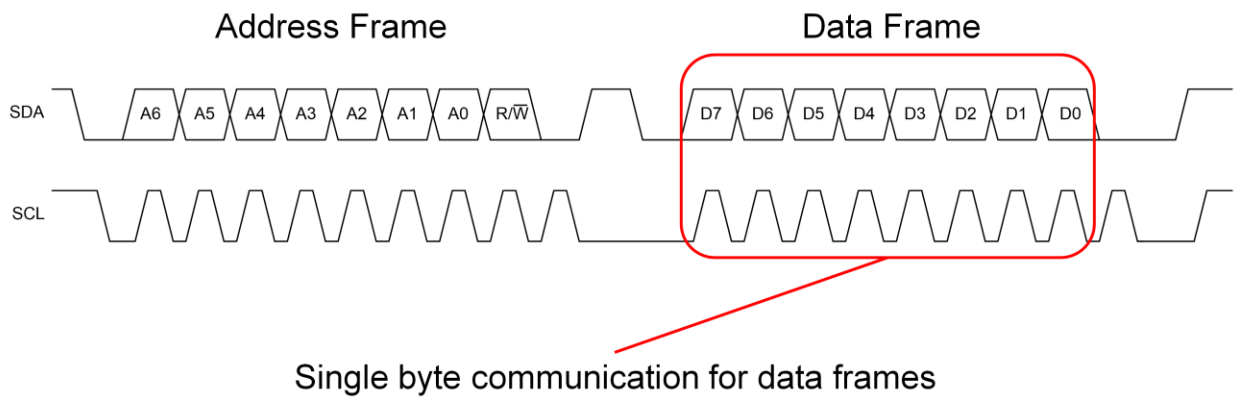
Following the address is the read – write bit. If this bit is 1, then the controller is asking the target device to read data from it. If this bit is 0, then the controller is asking to write data to the target device.

I2C Protocol – Timing Diagram



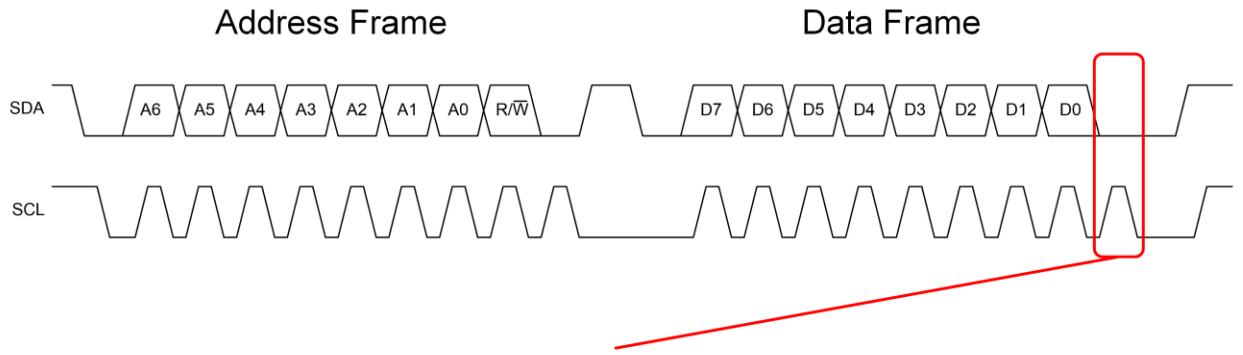
After any byte of communication between the controller device and the target device, one more bit is used to verify the communication was successful. At the end of the address byte communication, the target device pulls down the SDA during the SCL pulse to indicate that it understood that it was being contacted by the controller. This is known as an ACKT or acknowledge bit for the target. If this bit is high, then no target device understood that it was being contacted and the communication was unsuccessful. If the bit is high, this is known as a NACK and there was no acknowledge bit.

I2C Protocol – Timing Diagram



The address frame is followed by one or more data frames. These frames are sent one byte at a time.

I2C Protocol – Timing Diagram

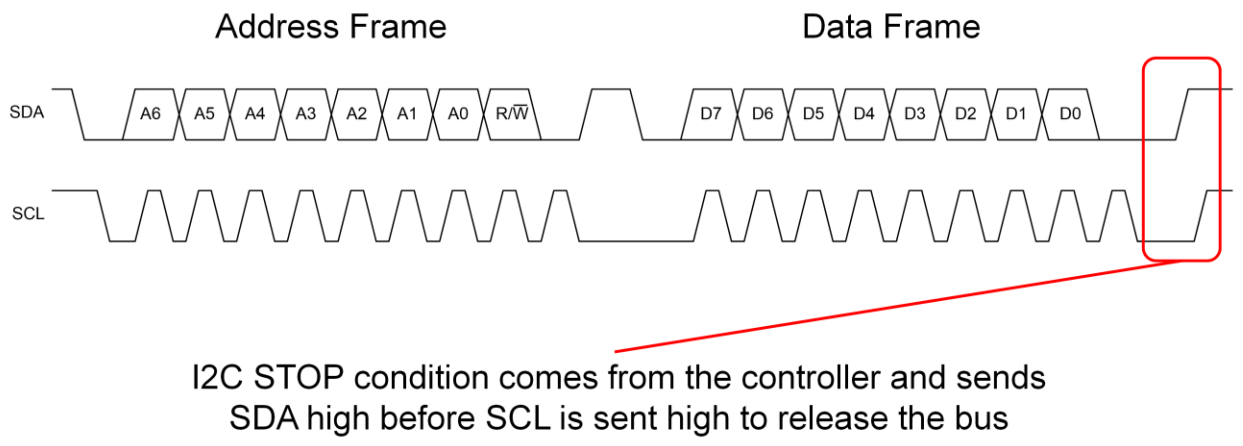


ACK follows each data frame
Write to the target – ACKT comes from the target device
Read from the target – ACKC comes from the controller device

After the data byte is transferred, there is another ACKT, or acknowledge from the target. If the data byte is a write to the device, then the target device pulls the SDA low to ACKnowledge the transfer.

If the data byte is a read from the device, the controller pulls the SDA low for an ACKC, or acknowledge from the controller, to acknowledge the receipt of the data.

I2C Protocol – Timing Diagram



After the communication is completed, the controller issues an I2C STOP condition. SCL is first released and then SDA is release. This is the controller indicating that the communication is completed and the I2C bus is released.

This is the basic setup for any I2C communication between the controller device and the target device. Communication may be comprised of more than on byte of data, and it may take a write and a read from the device to read any give device register.

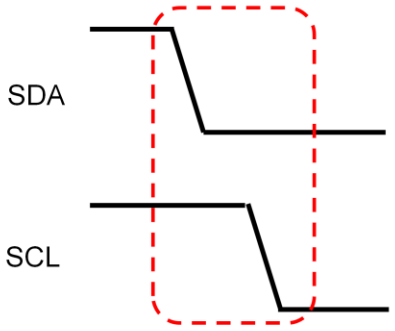
That concludes this video – thank you for watching! Please try the quiz to check your understanding of this video's content.

Quiz: Basics of I2C: The I2C Protocol

1. Before the address frame of I2C communication, what actions make up the START condition?
   a. The controller device sets the SDA low, and then sets the SCL low
   b. The controller device sets the SCL low, and then sets the SDA low
   c. The controller device sets the SCL low, and the target device pulls the SDA low as an ACK

Quiz: Basics of I2C: The I2C Protocol

1. Before the address frame of I2C communication, what actions make up the START condition?
   a. The controller device sets the SDA low, and then sets the SCL low
   b. The controller device sets the SCL low, and then sets the SDA low
   c. The controller device sets the SCL low, and the target device pulls the SDA low as an ACK

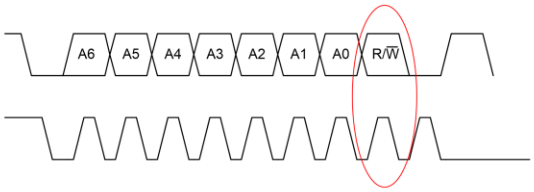


Quiz: Basics of I2C: The I2C Protocol

2. In the address frame, after the controller device sends the 7 bit address, what is the next part of the I2C protocol sent?
   a. The target device sends the ACK to acknowledge the communication coming from the controller device
   b. The controller device sends the R/W bit to indicate if it wants to read from or write to the target device
   c. The controller device send a STOP condition before sending the next data

Quiz: Basics of I2C: The I2C Protocol

2. In the address frame, after the controller device sends the 7 bit address, what is the next part of the I2C protocol sent?
   a. The target device sends the ACK to acknowledge the communication coming from the controller device
   b. The controller device sends the R/W bit to indicate if it wants to read from or write to the target device
   c. The controller device send a STOP condition before sending the next data

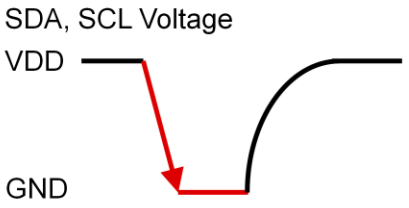


Quiz: Basics of I2C: The I2C Protocol

3. Because of the NMOS open-drain connection to SDA and SCL, which part of the communication waveform is faster?
   a. The rise time of SDA and SCL
   b. The fall time of SDA and SCL
   c. The rise time and fall time of SDA and SCL are the same

Quiz: Basics of I2C: The I2C Protocol

3. Because of the NMOS open-drain connection to SDA and SCL, which part of the communication waveform is faster?
   a. The rise time of SDA and SCL
   b. The fall time of SDA and SCL
   c. The rise time and fall time of SDA and SCL are the same



Open-drain connections are actively pulled down instead and are faster than a resistive pull up

Quiz: Basics of I2C: The I2C Protocol

4. What is the benefit of having an open-drain connection over push-pull outputs for I2C?
   a. High speed drive for the bus outputs
   b. Reduction of bus capacitance
   c. Prevents destructive current draw during bus contention when outputs are tied together

Quiz: Basics of I2C: The I2C Protocol

4. What is the benefit of having an open-drain connection over push-pull outputs for I2C?
   a. High speed drive for the bus outputs
   b. Reduction of bus capacitance
   c. Prevents destructive current draw during bus contention when outputs are tied together

Push-Pull outputs may pull a large current when the outputs are tied together and there is bus contention