Application Report

SCPA054–July 2018

Why, When, and How to use I^{2}C Buffers

Francis Houde

ABSTRACT

Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction to I²C

There is an enormous range of devices that use the I²C communications interface and system designers are only limited by their own creativity. Here are just a few types of devices that can be used: Input/Output (I/O) expanders, temperature sensors, light sensors, memory, key pad scanners, pressure sensors, humidity sensors, ADCs, DACs, and a variety of other devices. Having a bus composed of parallel connections is advantageous because adding devices is as simple as connecting to the bus at any location. Furthermore, the additions of devices can be done at any time.

The I²C bus is a bidirectional interface that uses a controller, known as the master, to communicate with slave devices. A slave may not transmit data unless it has been addressed by the master. Each device on the I²C bus has a specific device address to differentiate between other devices that are on the same I²C bus. The physical I2C interface consists of two wires, which are the serial clock (SCL) and serial data (SDA) lines. Both SDA and SCL lines have an open drain or collector drive with an input buffer that supports bidirectional communications or data transfer and must be connected to V_{CC} through a pullup resistor. The size of the pullup resistor is determined by the amount of capacitance on the I²C bus. Either the master can generate the low signal on the bus or the slave can generate the low signal. The high is completely dependent on the pullup resistor.

2 Why, When, and How to Use I²C Buffer or Repeaters

The I²C interface has been standardized and therefore all devices should be designed such that they meet the standard to ensure that communication works reliably. The I²C interface is split up into modes of operation, which specify the maximum clock frequency range of operation, maximum bus capacitance, maximum rise time, and many other variables. The three most often used modes of operation are Standard Mode, Fast Mode, and Fast Mode Plus. The most important parameters for selecting and using an I²C buffer, which is sometimes referred to as repeater, are listed in Table 1:

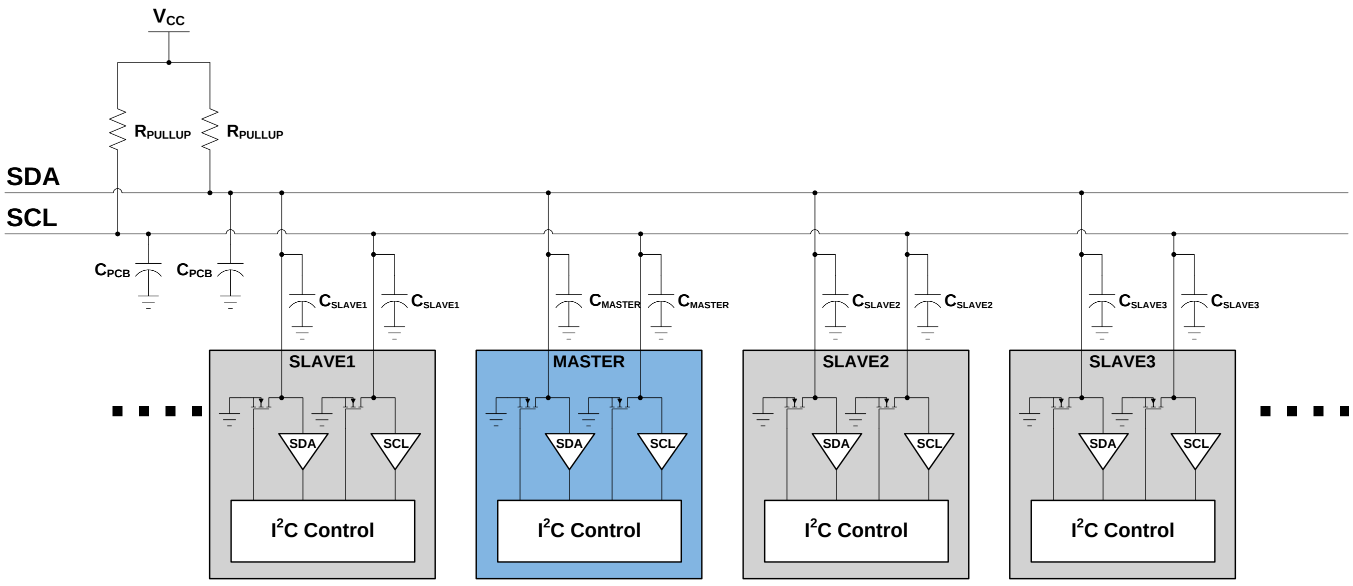
Table 1. Summary of Key Parameters for Most Common Modes of Operation used in I²C.

	Standard Mode	Fast Mode	Fast Mode Plus
$f_{CLOCK\ MAX}$	100 kHz	400 kHz	1,000 kHz
$C_{BUS\ MAX}$	400 pF	400 pF	500 pF
$t_{RISE\ MAX}$	1,000 ns	300 ns	120 ns

2.1 Why do I Need a Buffer or Repeater

2.1.1 Capacitances on the Bus

The I²C bus is a bus in which slave devices are connected in parallel, where multiple devices can be added on the bus which adds capacitance on the bus along with the PCB traces on the bus, see Figure 1. Both SDA and SCL lines of the I²C bus must comply with the standard. This application note discusses a single bus, but it will pertain to both SCL and SDA lines. The single bus is meant to represent either SDA or SCL.



$$C_{BUS} = C_{PCB} + C_{MASTER} + C_{SLAVE1} + C_{SLAVE2} + C_{SLAVE3} + \cdot\ \cdot\ \cdot\ \cdot$$

Figure 1. Cumulative Bus Capacitance due to PCB, Master, and Slaves on the Bus

2.1.2 Logic Control of Open Drain Interface

The I^{2}C interface uses an open drain driver with an input buffer to determine logic signals on the bus, which is very different compared to other interfaces that use push-pull drivers. The I^{2}C interface idles in the high state, meaning the signals on the bus are high unless there is communications. The pullup resistor generates the high for both directions of data transfer, see Figure 2.

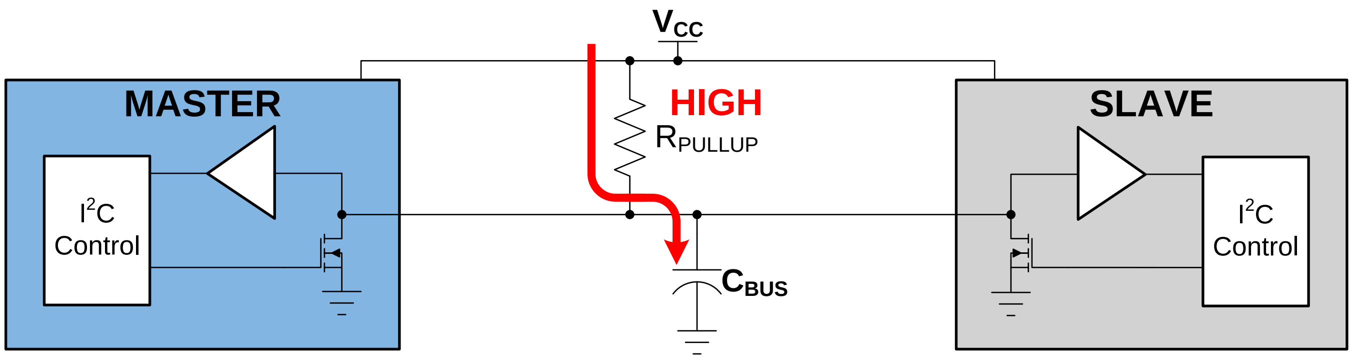


Figure 2. Diagram of How Pullup Resistor Charges Bus to Establish the High Logic Level on the Bus

The low on the bus is generated when the internal FET is turned on by either the master or the slave. The direction of the communication is controlled by which device is controlling the lows, see Figure 3 and Figure 4.

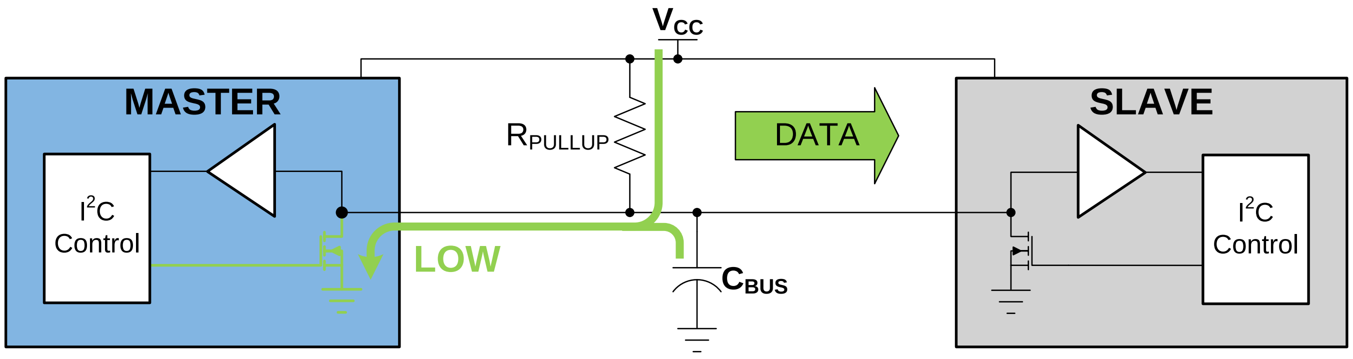


Figure 3. Diagram of How Master Discharges Bus and Pulls the Bus to Ground to Generate Logic Level Low on the Bus

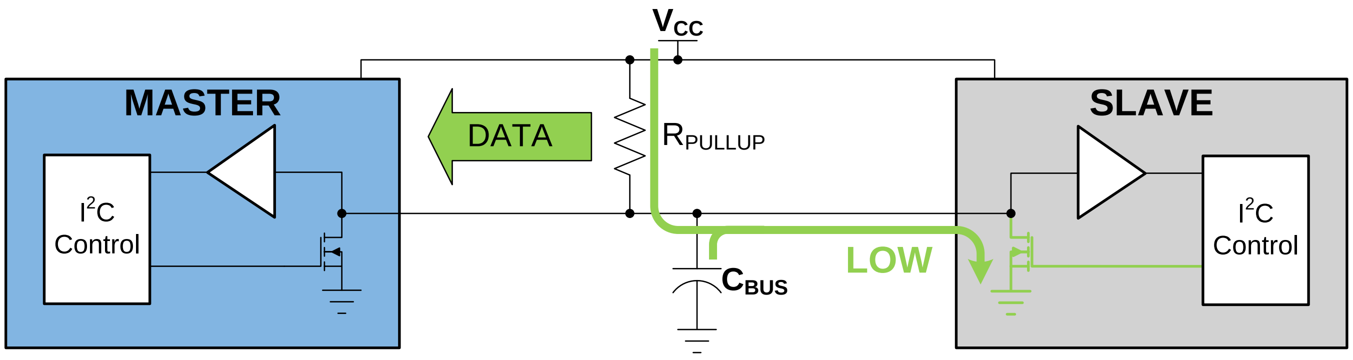


Figure 4. Diagram of How Slave Discharges Bus and Pulls Bus to Ground to Generate Logic Level Low on the Bus

2.1.3 Rise and Fall Time on Open Drain Interface

It is important to note that the effective resistance on the bus to ground when the FET on is much lower (on the order of 5 to 133 ohms) than the pullup resistance to V_{CC}, which can range from 1k to 10k ohms. Given that there is a finite amount of capacitance on the bus, this means that the fall time will be much faster than the rise time. The rise time and fall time is defined by the time constant tau (t), where t = R´C. C is the bus capacitance and R is either pullup resistor or resistance to ground due to FET being turned on. The most critical timing parameter is the rise time, which is determined by the pullup resistor and the capacitance on the bus. The upper limit of the pullup resistance must be selected such that it does not exceed the max rise time for that mode of operation. The max rise and fall times is defined from 30% to 70% of V_{CC} for the I²C interface, see Figure 5.

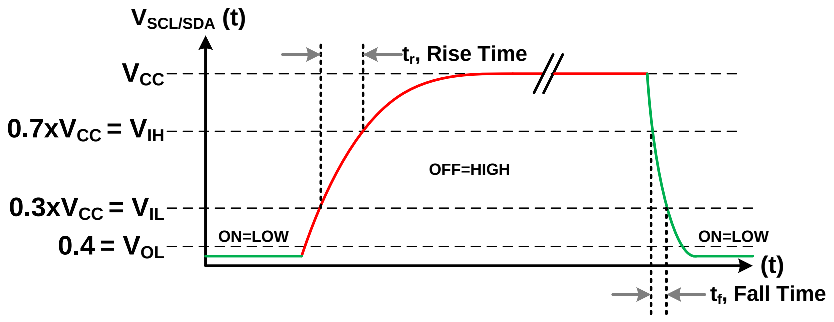


Figure 5. I²C Waveform with Logic Level, Rise Time, and Fall Time Information for SCL and SDA

2.1.4 R_{PULLUP} and How it Affects V_{OL}

There is a lower limit to the pullup resistor because if the resistance is too low (meaning pullup current is to strong/high) then the V_{OL} could potentially be higher than the V_{IL} of the input buffer, thus it could not detect logic low properly. The lower limit of the pullup resistor is a function of the drain to source resistance of the FET while on (R_{DS_ON}) and the pullup resistor (R_{PULLUP}). This is a simple resistor divider between R_{DS_ON} and R_{PULLUP}, see Figure 6. The pullup resistor affects the V_{OL}, where R_{PULLUP} is inversely proportional to V_{OL}. As R_{PULLUP} decreases in value the V_{OL} will increase in value and V_{OL} must be kept below the V_{IL} of any of the devices on the bus. It is also worth noting the lower the pullup resistor is the greater the power dissipation is when communicating.

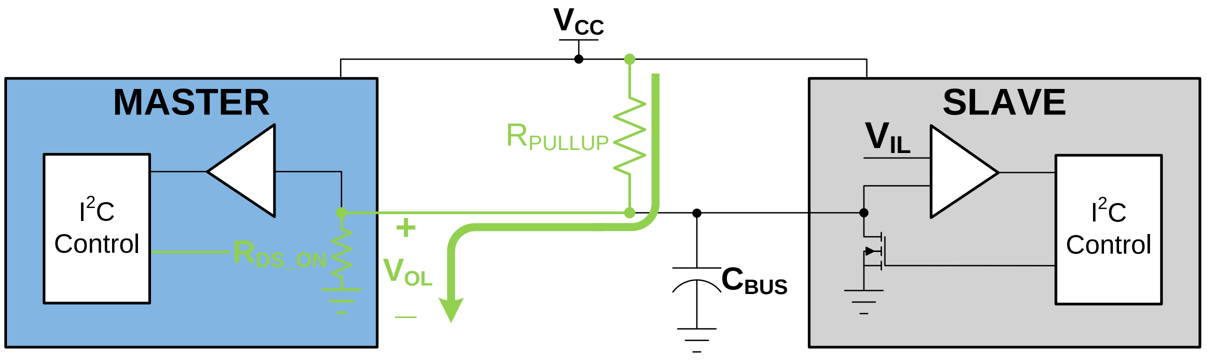


Figure 6. Resistor Divider Created when Pull Down FET is on With the Pullup Resistor and How V_{OL} is Determined

2.1.5 Subdividing Bus Capacitance with a Buffer

Once the resistance range is fixed, there is only one other variable that can be changed in order to control the rise time and that is the bus capacitance. Buffers subdivide a single bus capacitance into two separate bus capacitances. This allows rise times to be changed based on that subdivision of initial bus capacitance and the pullup resistance used on each of the two separated buses. See the subdivided bus capacitance and pullup resistance (C_{BUS1} $R_{PULLUP1}$, C_{BUS2}, and $R_{PULLUP2}$) in Figure 7. For each bus capacitance, there will be a min and max resistance value based V_{OL} and rise time calculations. The ability to reduce the max rise time by subdividing a single bus capacitance into two separate bus capacitances is the reason why buffers are used.

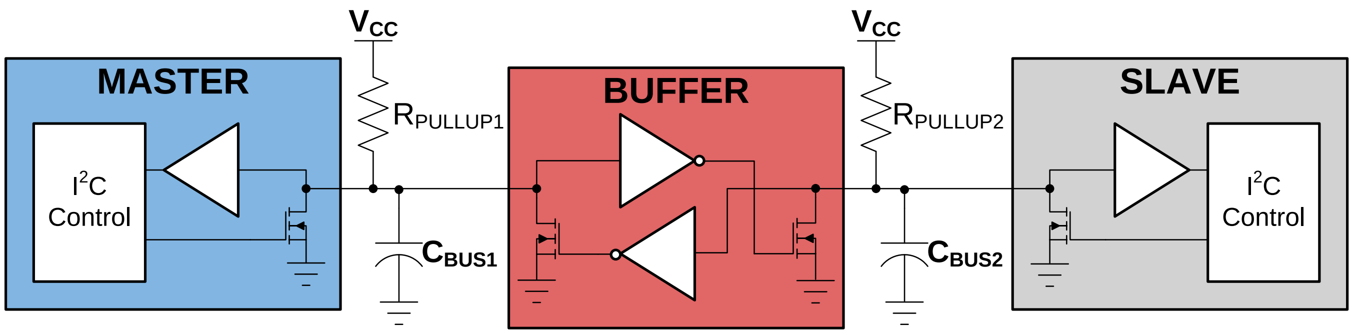


Figure 7. How C_{BUS} and R_{PULLUP} gets Subdivided into C_{BUS1}, $R_{PULLUP1}$, C_{BUS2}, and $R_{PULLUP2}$ Using a Buffer

2.2 *When do I Need a Buffer (Sometime Referred to as a Repeater)*

The I^2C standard specifies a max bus capacitance ($C_{BUS\ MAX}$) of 400 pF for both Standard Mode and Fast Mode and specifies 550 pF for Fast Mode Plus. This is not a lot of capacitance when considering the cumulative capacitance from PCB traces and the number of devices that can be supported by the I^2C interface, which has the ability to use multiple hundreds of device addresses.

Suppose an I^{2}C system using Fast Mode has 40 slave devices, which adds 10 pF per slave, a master device that adds 10 pF, and a PCB trace that adds 200 pF of capacitance, see Figure 8. The total capacitance on that bus is equal to 610 pF, which exceeds the max bus capacitance ($C_{BUS\ MAX}$) specification that is outlined in the I^{2}C standard for Fast Mode operation.

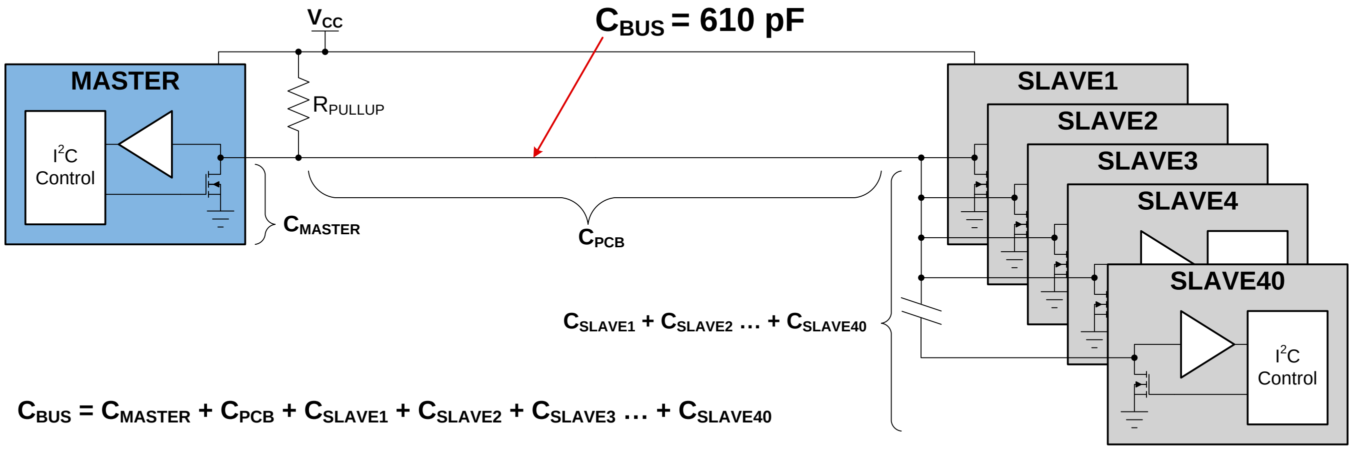

V_{CC}
C_{BUS} = 610 pF
MASTER
I^{2}C Control
R_{PULLUP}
C_{MASTER}
C_{PCB}
SLAVE1
SLAVE2
SLAVE3
SLAVE4
SLAVE40
I^{2}C Control
C_{SLAVE1} + C_{SLAVE2} … + $C_{SLAVE40}$
C_{BUS} = C_{MASTER} + C_{PCB} + C_{SLAVE1} + C_{SLAVE2} + C_{SLAVE3} … + $C_{SLAVE40}$

Figure 8. Example I^{2}C System that Includes Capacitance on the Bus due to Master, PCB, and all the Slaves on the Bus

If the system requires having a larger bus capacitance, then we must use a buffer to subdivide the bus into two buses. This redistributes the total capacitive load amongst BUS1 and BUS2 so that it meets the I^{2}C standard, see Figure 9. In this example the bus capacitance due to the PCB was divided in half between BUS1 and BUS2. Half of the slaves (odd numbered slaves) plus the master were put onto BUS1 and the other half the slaves (even numbered slaves) were put on BUS2. The buffer adds 10 pF of capacitance (CBUS1 and CBUS2) to each side of the buffer. BUS1 has a total of 320 pF on the bus, which is less than the maximum of 400 pF per the I^{2}C standard, and BUS2 has a total of 310 pF. Each bus can have a total of 400 pF per the I^{2}C standard; therefore the total system bus capacitances could total 800 pF if evenly distributed and still meet the I^{2}C standards.

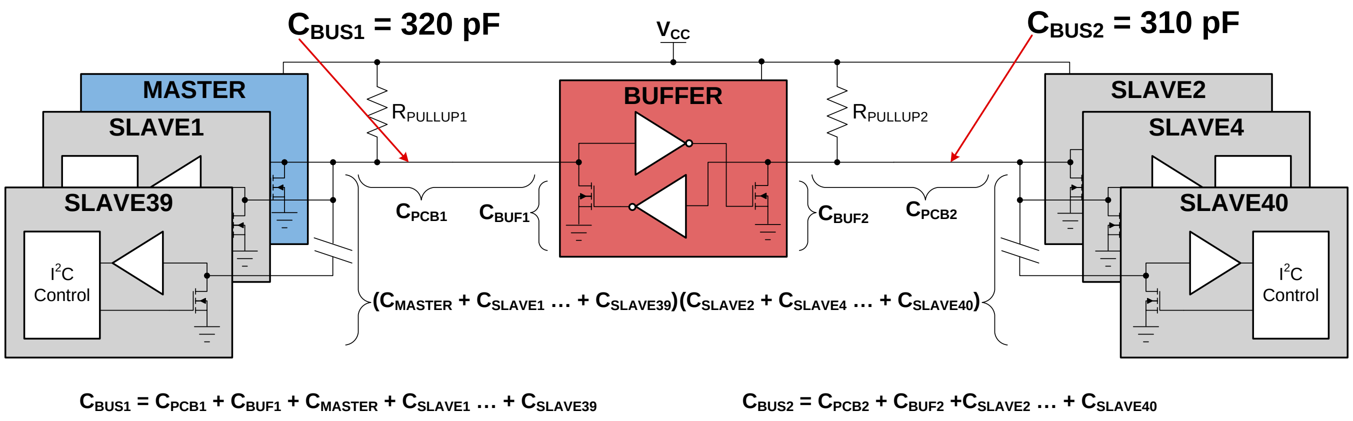

C_{BUS1} = 320 pF
V_{CC}
C_{BUS2} = 310 pF
MASTER
SLAVE1
SLAVE39
I^{2}C Control
$R_{PULLUP1}$
BUFFER
$R_{PULLUP2}$
SLAVE2
SLAVE4
SLAVE40
I^{2}C Control
C_{PCB1}
C_{BUF1}
C_{BUF2}
C_{PCB2}
(C_{MASTER} + C_{SLAVE1} … + $C_{SLAVE39}$)(C_{SLAVE2} + C_{SLAVE4} … + $C_{SLAVE40}$)
C_{BUS1} = C_{PCB1} + C_{BUF1} + C_{MASTER} + C_{SLAVE1} … + $C_{SLAVE39}$
C_{BUS2} = C_{PCB2} + C_{BUF2} +C_{SLAVE2} … + $C_{SLAVE40}$

Figure 9. Example of a Buffer Subdividing the Total bus Capacitance into two Buses that are Evenly Distributed

Adding the buffer indicates that there are two pullup resistors, one for each bus, to set the max rise time and set V_{OL} on the bus. The I^{2}C standard specifies the max bus capacitance, the max rise time, and the V_{OL}, therefore the standard dictates when you will need to add a buffer.

2.3 How an I²C Buffer Works

2.3.1 What Happens When a Buffer is not Designed Properly

The fact that the I²C interface is bidirectional adds another level of complexity. Figure 10 illustrates what can happen if a buffer is not designed to support bidirectional data transmission. The circuit is initially in an idle state where both sides of the buffer are high.

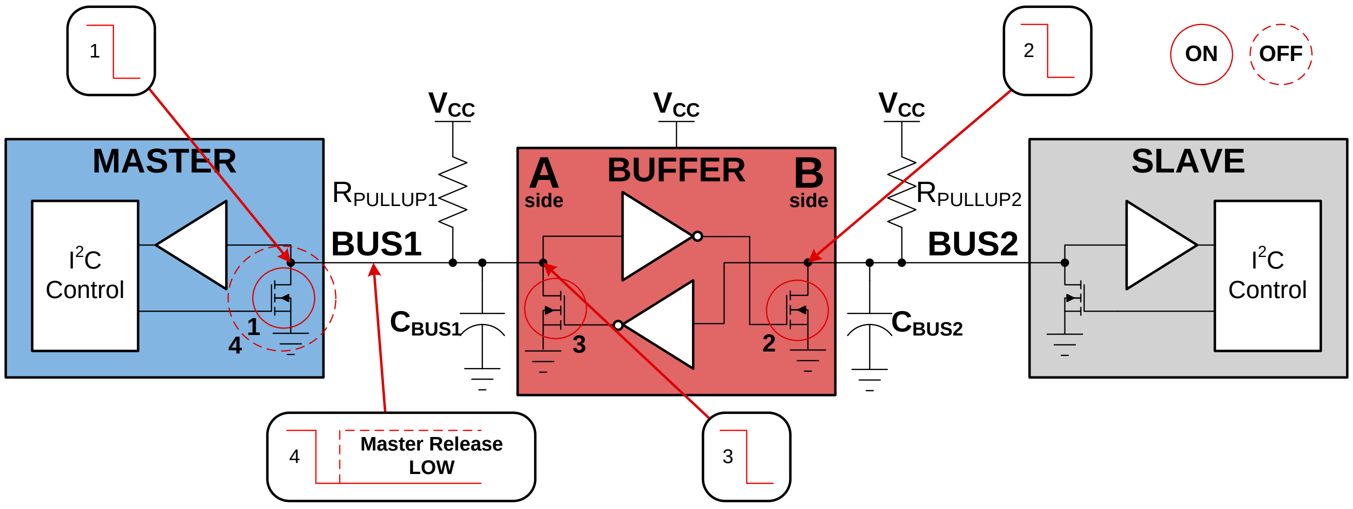


Figure 10. Timing Sequence of two Opposing Buffers to Support Bidirectional Signal and How it Gets Locked Up

2.3.2 How to Determine Which Side of the Buffer is Pulled Low

A method is needed to differentiate which side of the device is pulled low, the master or the slave? There are many ways that can be achieved, but this application note will concentrate on the method that uses the static voltage offset (SVO). The static voltage offset is a method to have single side have two states that meet the standard definition of a low in I²C, which is $V_{IL} \leq 0.3 \times V_{CC}$. There are two low states that determine which side is pulling low. State 1 is from V_{IL} to SVO and State 2 is from SVO to 0V, see Figure 11.

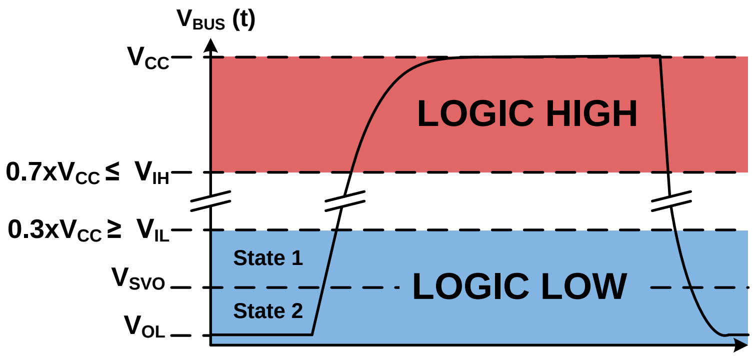


Figure 11. Diagram of Valid Logic High and Logic Low Levels for I²C and the Two States Created with a Static Voltage Offset.

2.3.3 How the Static Voltage Offset Works

Figure 12 shows a buffer that has a static voltage offset towards the slave side of the bus. The static voltage offset of this particular buffer is 0.5 V and is represented as a low voltage Zener diode above the pull down FET. Zener diodes are reverse biased diodes that are designed to conduct current when the reverse voltage on the diode exceeds its Zener voltage, which is the voltage at which the Zener is designed to regulate. A Zener is a voltage controlled current sink that requires a minimum current to regulate properly.

Each side of the buffer is labeled as side A and side B to help with describing how the low is propagated from one side to the other. I^{2}C is a bidirectional bus so both A and B sides can have either a master or slave. In this case, the master is connected to the A side of the buffer and the slave is connected to the B side of the buffer.

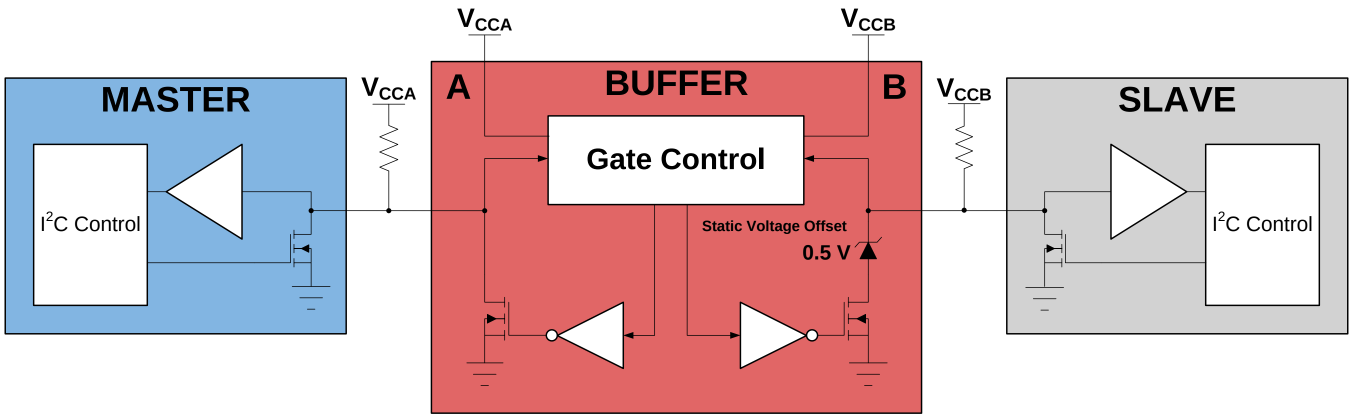


Figure 12. Example of a Buffer That has a Static Voltage Offset That is on the "B" Side of the Buffer

2.3.4 Low is Generated from A Side and Propagated to B Side

Consider when the master pulls low, see Figure 13. The master's pulldown FET is turned on (1) and is pulled low on the A side of the device. The buffer then proceeds to turn on the B side FET (2), which places the static voltage offset of 0.5 V onto the B side bus. The SVO is lower than the V_{IL} of the slave thus it is interpreted as a low.

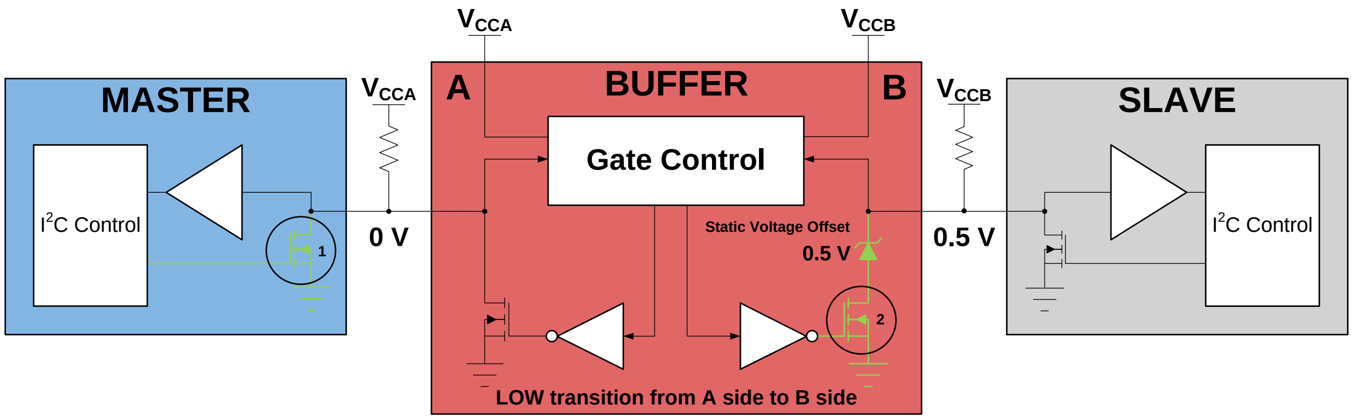


Figure 13. An Example of a Logic Level Low Being Propagated from the A Side to the B Side of the Buffer

2.3.5 Low is Generated from B Side and Propagated to the A Side

Now consider when the slave pulls low, see Figure 14 . The slave pulldown FET in turned on (1) and generates a low for the B side. That gets relayed across buffer to the A side's pull down FET which gets turn on (2) generating a low on the A side.

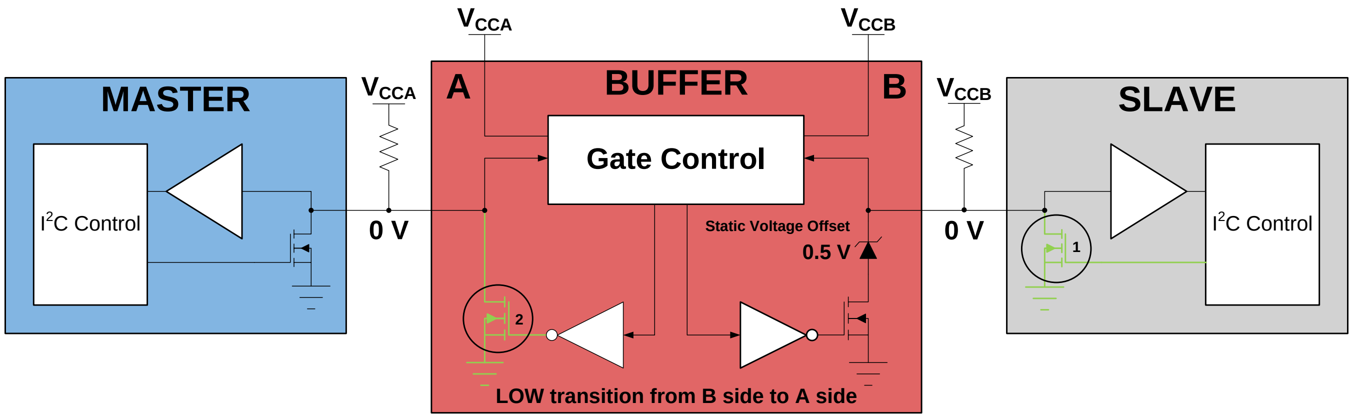


Figure 14. An example of a logic level low being propagated from the B side to the A side of the buffer.

It is clear that the "Gate Control" must monitor the B side voltage and if it sees a voltage of 0.5 V then it knows it came from the A side and not from something on the B side of the buffer (remember the buffer detects when it turns on either of its internal FETs). If the buffer sees a voltage lower than 0.5 V, it knows that a device on the B side other than itself has turned on their internal FET(1). The buffer propagates low to the A side and turns on its own FET(2). This method determines what side has generated the low and the bus no longer gets locked up.

2.4 *How to use I²C Buffers*

When using an I²C buffer with a static voltage offset, there are some concerns in selecting the device and how to implement buffers in the system. In the previous example, there was a buffer with a static voltage offset of 0.5 V. The actual static voltage offset voltage level can vary depending on how the buffer was designed. Texas Instruments has buffers with static voltage offsets that range from ~0.1 V to ~0.6 V. Furthermore, the static voltage offset is not always on the B side of a device. Some buffers have them on the B side, some are on the A side, some buffers have a static voltage offset that changes side (more on that later), and some devices use a static current offset to determine which side of the buffer is generating the low. For now, let's assume the buffer has the static voltage offset on the B side of the device.

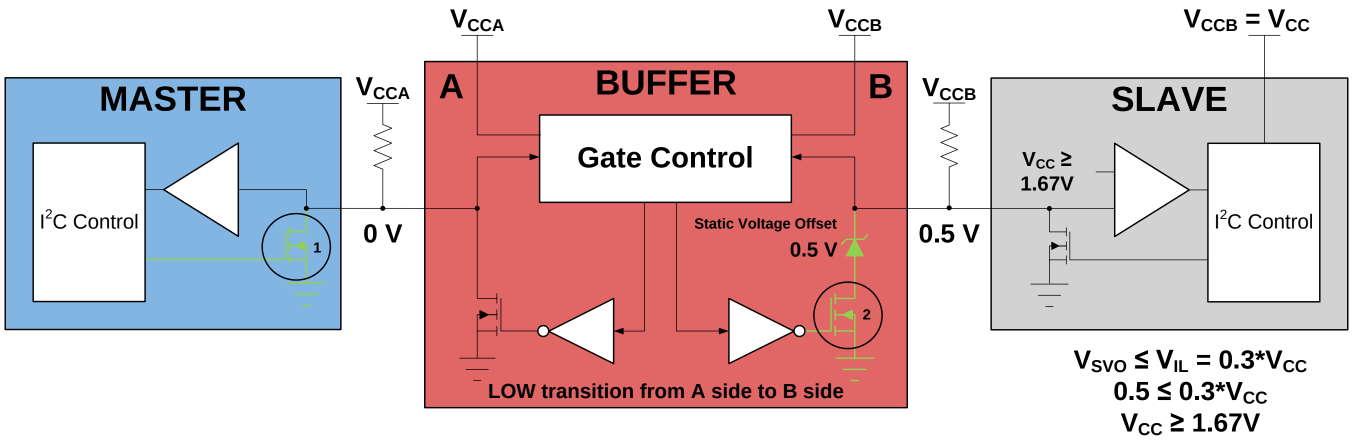


Figure 15. Diagram of how to Check of the SVO is not Violating the V_{IL} of the Slave Device

2.4.1 Are There Other Buffers on the Bus and do They Use a Static Voltage Offset?

There are buffers that have different SVO voltage levels and the SVO can be on either A or B sides of the device. It is important to always make sure that if buffers are connected in either series or in parallel they never connect two buffers SVOs together.

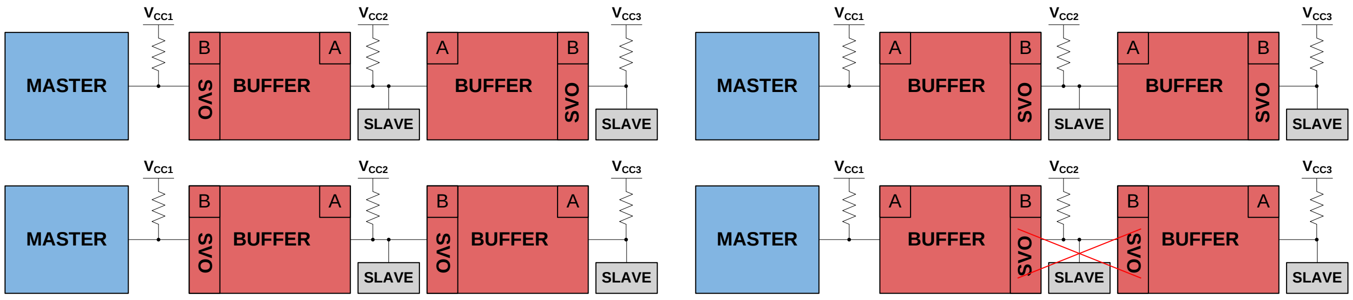


Figure 16. All Possible Series Buffer Combinations that are Allowed and not Allowed (red cross)

Two SVOs can never be connected in parallel. Figure 17 has examples of parallel connections and if they are acceptable or not acceptable (red cross).

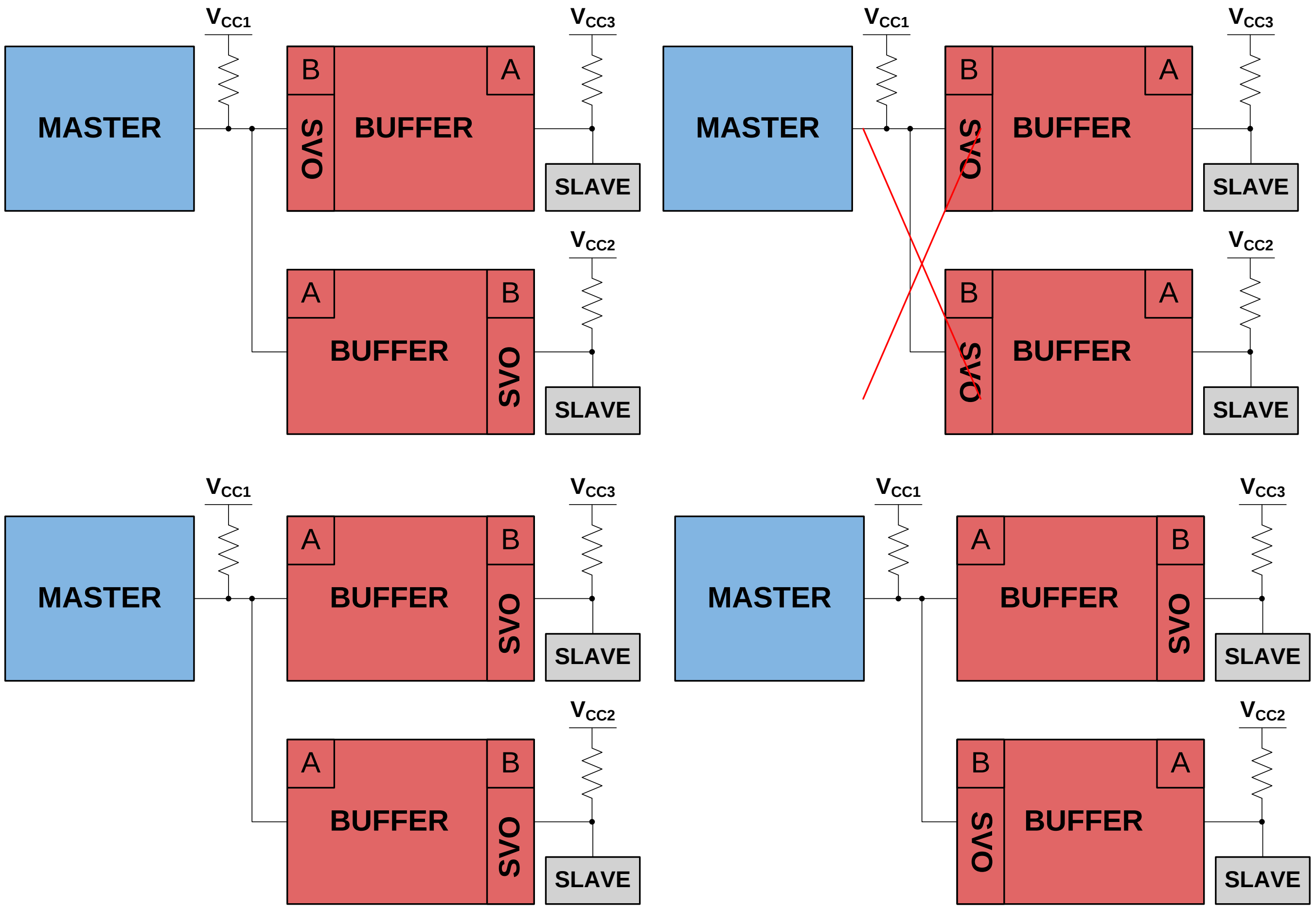


Figure 17. All Possible Buffer Combinations that are Allowed and Not Allowed (red cross).

The reason that two SVOs cannot be connected together is simple. There are errors in relaying a low across the buffer depending on which side is pulled low. In the example below, there is one buffer that has an SVO = 0.5 V and the other buffer that has an SVO = 0.4 V. The low only propagates when generated from the master, see Figure 18, and not when propagated from the slave, see Figure 19. The low needs to be propagated from both directions seeing that this is a bidirectional interface. Thus, the rules for never connecting two static voltage offsets together whether in series or parallel must be followed.

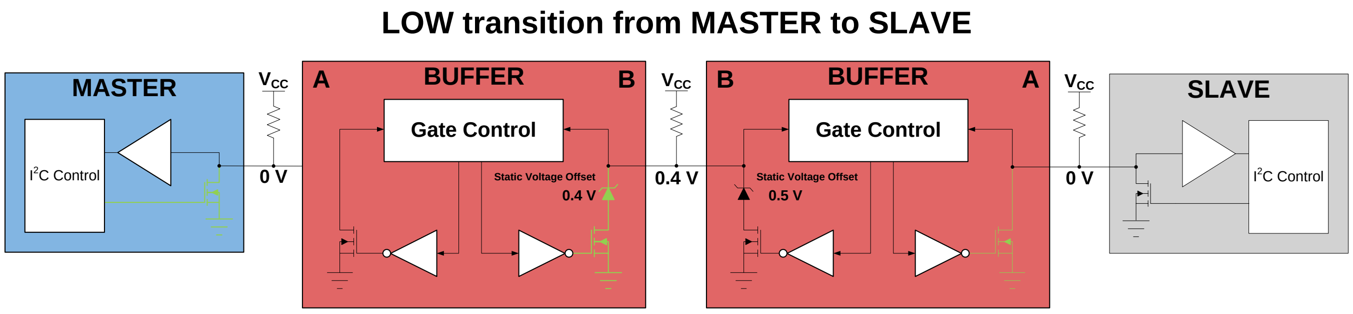


Figure 18. Diagram for Propagating a Low from the Master to the Slave.

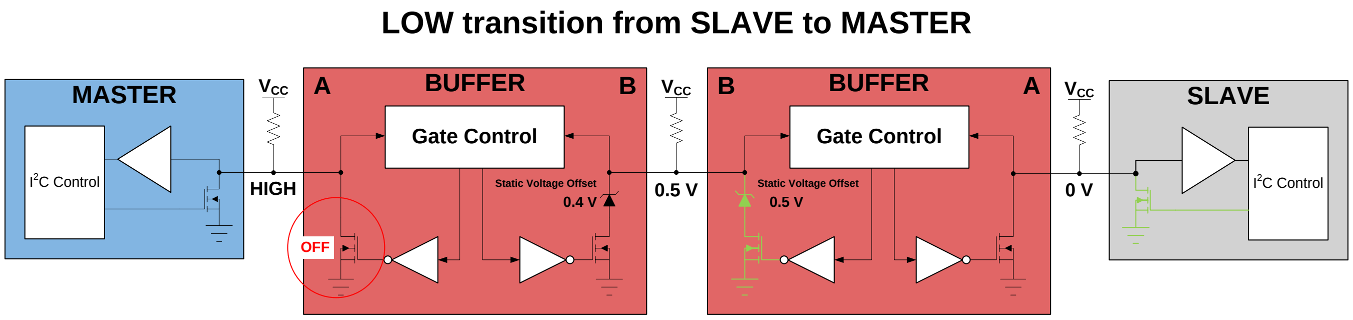


Figure 19. Diagram for Propagating a Low from the Slave to the Master Which Shows the Master Failing to see the Low.

2.4.2 Are There any Supply Requirements that Affect how the Device is Placed with Respect to A Side and B Side?

When putting buffers in series or in parallel in an I^2C system, always check to make sure there are no V_{CC} rule violations outlined for the supplies of the buffer. Each side of the buffer (A side or B side) has an operating voltage range and many times has specific rules about how each side of the buffer must relate to each other. Here are some examples of the rules that must be considered:

- $V_{CCA} \leq V_{CCB}$
- $V_{CCA} \leq (V_{CCB}$ -1 V)
- V_{CC} Single Supply (no translation and only buffering)
- No Rules (Meaning V_{CCA} can be either ≥ or ≤ V_{CCB}, as long as you stay within each sides operating voltage range).

These rules need to be considered in conjunction with SVO and operating mode rules when designing your I^2C system. Here is an example where the system designer tries to use two TCA9617B buffer, which has a $V_{CCA} \leq V_{CCB}$ rule, to separate bus capacitances and perform voltage translation using Fast Mode, see Figure 20. Figure 20 shows that the voltage range for V_{CC1}, V_{CC2}, and V_{CC3} are within each buffer's respective V_{CCA} and V_{CCB} ranges, but the second buffer is violating the V_{CC} rule of $V_{CCA} \leq V_{CCB}$. Figure 21 shows the how the designer flips the A and B side to fix the V_{CC} rule, but inadvertently creates a SVO rule violation. One solution is to mix two different buffers (TCA9617B and TCA9517) that do not violate V_{CC} rules, see Figure 22. In fact, there are a number of ways this could have been accomplished, but each configuration must always have the V_{CC} rules checked to ensure proper operation.

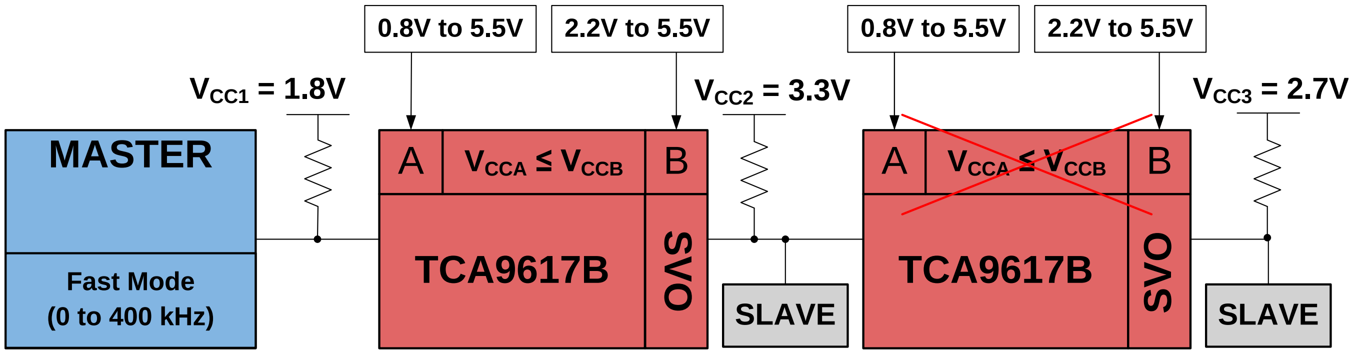


Figure 20. Trying to use Two TCA9617B to Separate Bus Capacitance and Perform Translation for Three Different Buses

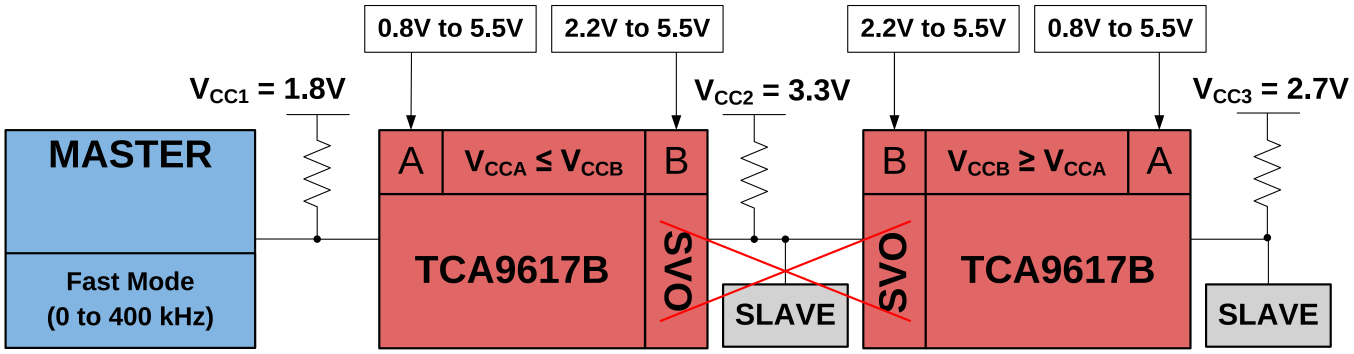


Figure 21. Trying to Switch A and B Sides to not Violate V_{CC} Rules, SVO Rule this is Violated

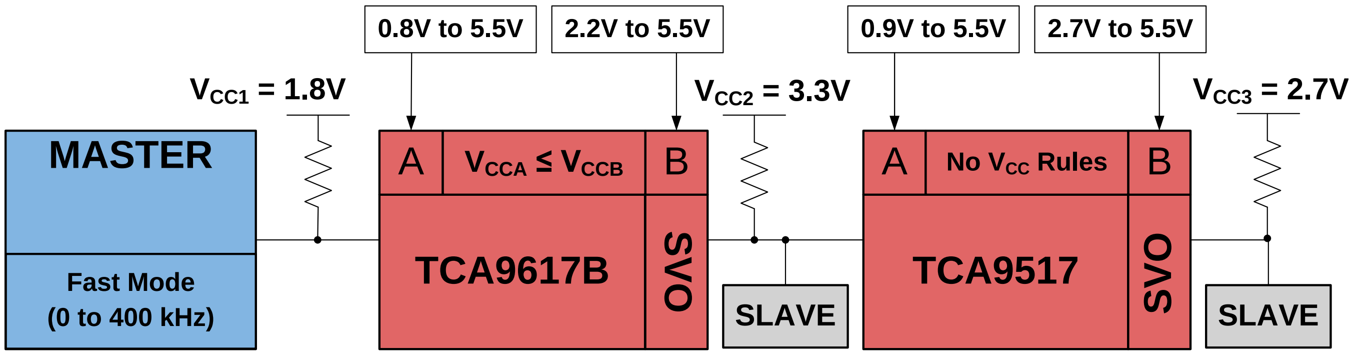


Figure 22. Example of One Solution that Will Work for Separating Bus Capacitance and Translations for Three Buses

2.4.3 Is There More Than One Mode of Operation (Clock Frequency) on the Bus?

If a master needs to switch between more than one mode of operation then the lowest frequency mode needs to be isolated when the faster mode is being used. Isolating the lower speed bus from the higher speed bus removes the possibility of having a slave designed for the lower speed to be exposed to logic signals outside its specified and verified data frequency range. The previous example, Figure 23, is not allowed if the decision is to operate at Fast Mode Plus (1 MHz) and want to communicate to the slave on the third bus. This is because the TCA9517 does not support Fast Mode Plus at 1 MHz, see Figure 24.

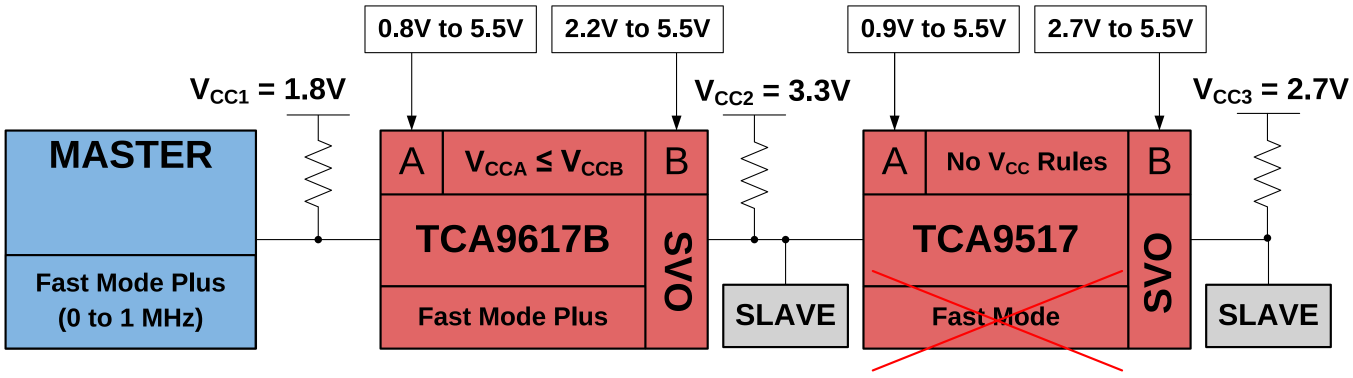


Figure 23. TCA9517 Does not Support Fast Mode Plus

This means the design needs to be changed such that all buffers are designed for Fast Mode Plus and follow all other rules too. Figure 24 shows an example of a solution that would support the slave's mode of operation, the V_{CC} rules, and the SVO rules.

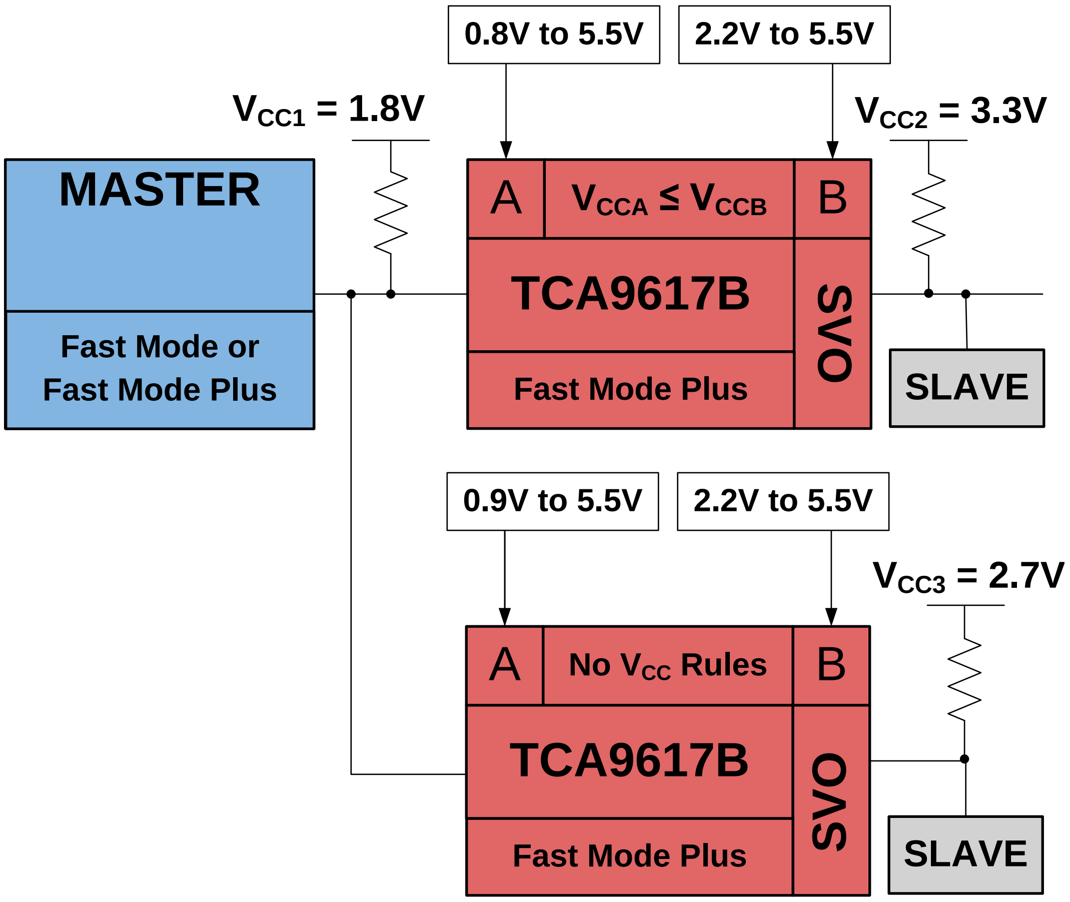


Figure 24. Example of Buffer Solution that Supports two Slaves that can Operate at Fast Mode Plus

If the master toggles between modes of operation then it has to disable the lower frequency bus off while communicating in the faster mode of operation. That requires disabling the buffer for the slower mode of operation prior to communicating using the faster mode of operation. This is usually done with the master disabling the lower speed device with the buffer's enable pin while using communicating to the faster mode device, see Figure 25.

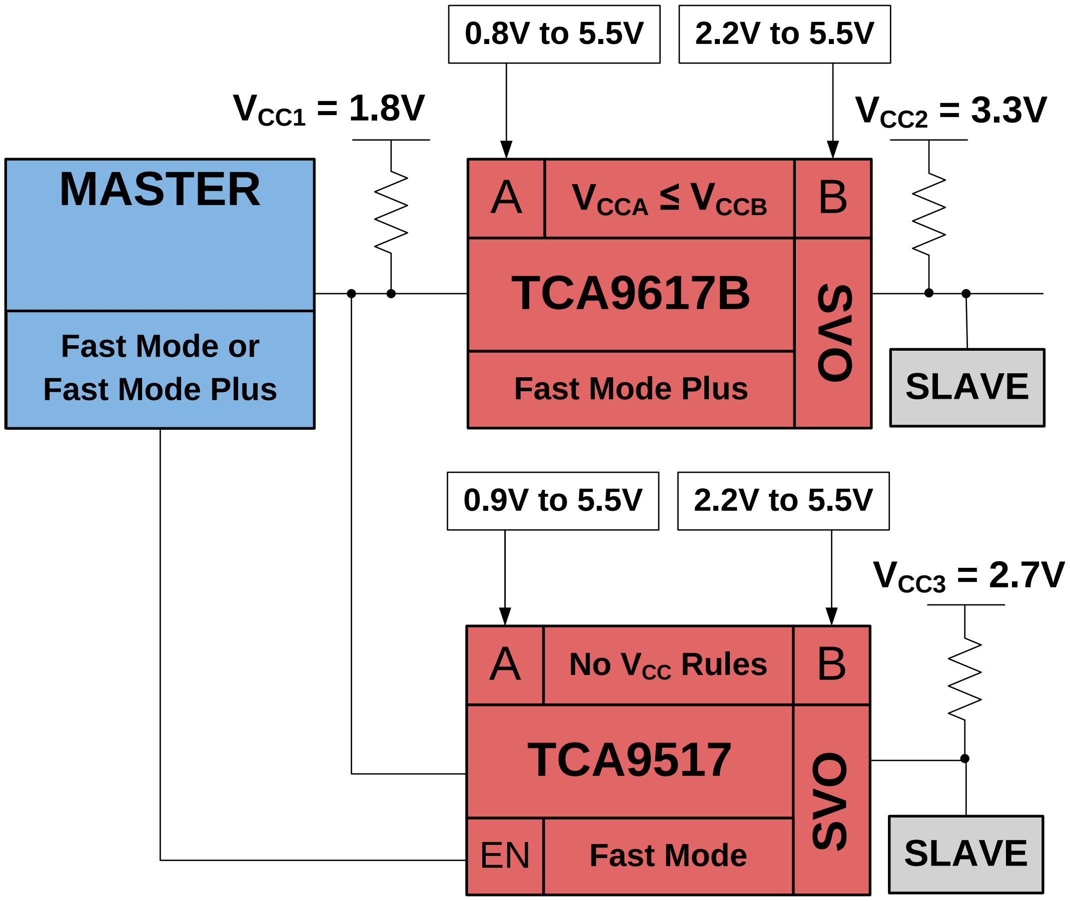


Figure 25. Example of an I^2C System with Two Slaves that Operate with Two Modes of Operation

It is assumed that if a slave is only designed to operate in one mode then its internal state machine has only been tested with signals going up to the maximum frequency of that mode of operation. There is a possibility that the state machine might actually respond or lock up at higher frequency clock and data lines; therefore, the lower speed mode should be disabled when communicating to slave using the faster mode.

Designing an I^2C system has many challenges and using buffers can solve many of the problems common to creating such a system. A good understanding of why, when, and how I^2C buffers are used is essential in creating an I^2C system that works reliably and meets the I^2C standards. There are a variety of design challenges when using buffers and great care needs to be taken when implementing them into I^2C system designs. Related I^2C documentation, training, and support (E2E) can be found at www.ti.com.

3 Support

I^{2}C E2E Community

4 Documentation

Choosing the Correct I2C Device for New Designs

Advantages and Design Considerations of TCA980x Family

Understanding the I2C bus

I2C Bus Pull-Up Resistor Calculations

Maximum Clock Frequency of I2C Bus Using Repeaters

Troubleshooting I2C Bus Protocol

SMBus Compatibility with an I2C Device