AN90044

A study of I2C with examples

Rev. 2 — 30 June 2023 Application note

Document information

Information	Content
Keywords	I2C, open-drain topology, timing specifications, capacitance, example register syntax
Abstract	This application note provides an overview of I2C technology and includes explaining the concepts.

nexperia

1. A brief overview of I²C

It has been over forty years since the inception of one of the most widely used communication protocols, and the name of that protocol is *Inter-Integrated Circuit*, more commonly known as I²C. It was originally developed in 1982 by Philips semiconductors and was created to provide an easy way to connect multiple target devices (peripherals) to a CPU – it accomplishes this through the open-drain topology of its two communication lines, SCL and SDA. These devices were originally designed and integrated into many of their consumer products, such as TV-sets and stereo systems.

I²C is a two-wire synchronous serial bus, with a standard operating frequency of 100kHz. Faster speeds, such as 400kHz, 1MHz, and 3.4MHz have also been introduced since its creation. The protocol uses 7-bit addressing that allows 128 unique addresses (16 of which reserved), which means a maximum of 112 controller or target devices can connect to the bus and communicate with one another. 10-bit extensions are also available; however, it is not common.

I²C specifics such as these faster speeds, open-drain topology, and example register syntax will be provided in this document. In addition to this, pull-up resistor and bus capacitance and its effects on the protocol will also be discussed.

1.1. Example system

A simplified representation of an I²C system is shown in Fig. 1. In I²C there are two different kinds of devices – controller and target devices. The controller is responsible for both communicating and receiving data from target devices on the same I²C bus. In the embedded system illustrated, the controller is found on the left as the microcontroller, with target devices on the right. All devices connect to two communication nets, and these nets are labeled as SCL and SDA. As the topology is open-drain, all nets connect to the system supply through pull-up resistors.

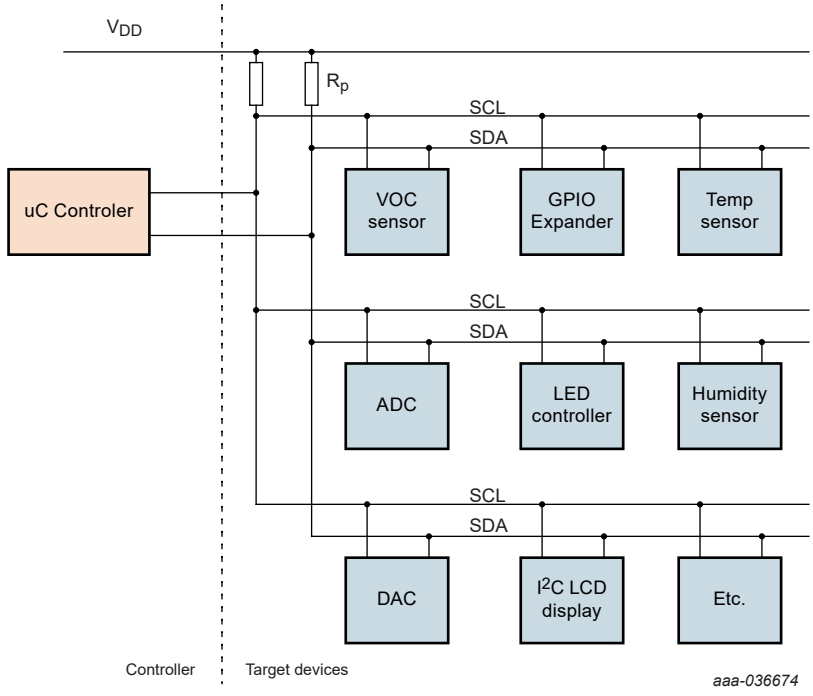


Fig. 1. Example I²C system

2. I²C modes

Table 1 displays the different modes for I²C, along with their respective Maximum Bus Capacitance specification and speed.

Table 1. I²C modes

I²C devices that are designed for a specific mode are also typically compatible with slower speeds.

Mode	Maximum capacitance	Maximum speed
Standard mode	400 pF	100 kHz
Fast mode	400 pF	400 kHz
Fast mode plus (Fm+)	550 pF	1 MHz
High-speed mode (Hs)	400 pF	3.4 MHz

3. Open-drain (open collector) topology of I²C

The inner architecture of all I²C compatible devices feature an open-drain topology, which includes a MOSFET that can connect the signal lines to ground or release the net allowing the node to rise to the supply voltage through the attached pull-up resistors (Rp). Unlike push-pull topologies, the open-drain architecture ensures that no shorting can occur if one device were to transmit a logic high while others transmit a logic low.

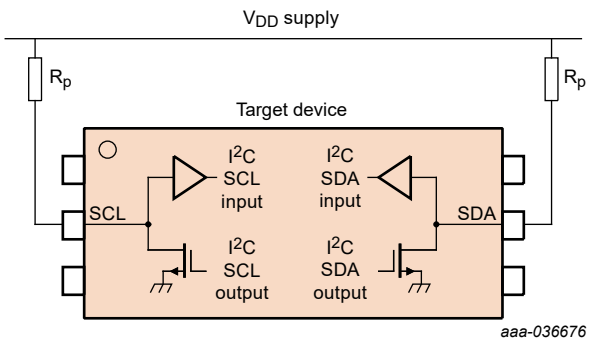


Fig. 2. Open-drain architecture

When the internal NMOS MOSFET is activated, it drives the signal net to ground, this is shown in Fig. 3. The drive strength of the FET is limited, and usually specified in the device datasheet. If there is capacitance on the net, the discharge will typically be linear, and can be represented by the equation I=C*(dv/dt).

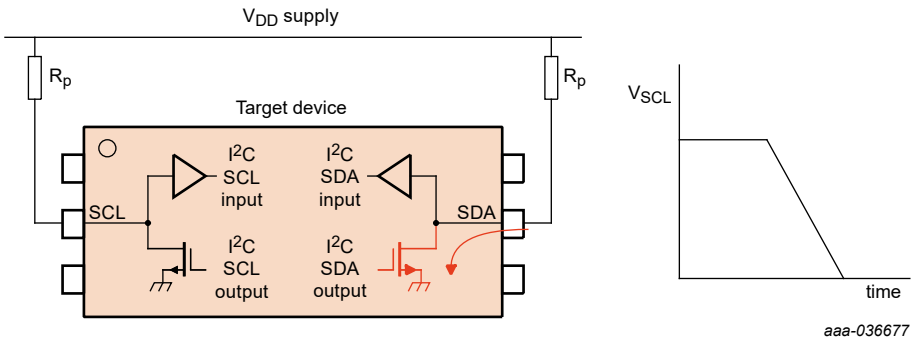


Fig. 3. SDA MOSFET connected to GND

When the node is released, and MOSFET deactivated, the SDA node will then be pulled up to V_{DD} through the external resistor. This resistance, along with any parasitic capacitance will affect the rise time and settling time of this node. This is illustrated in Fig. 4.

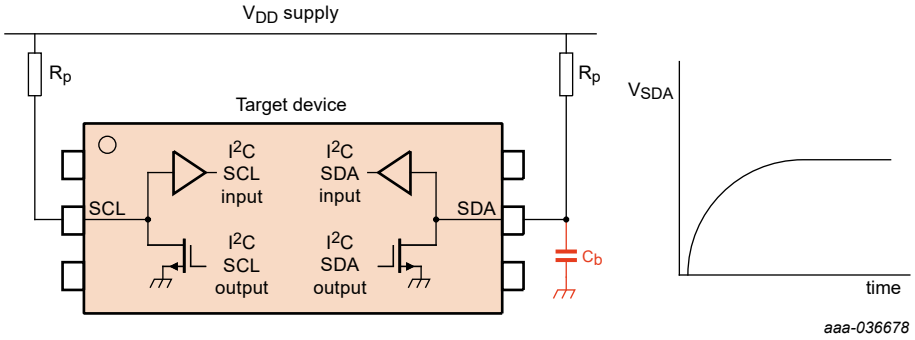


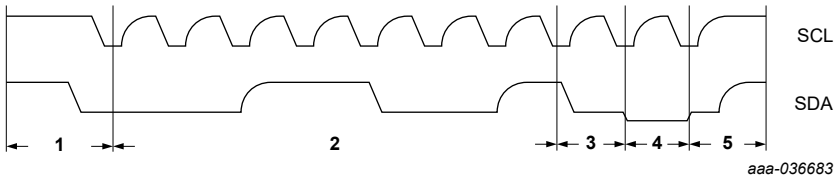
Fig. 4. SDA net released high

4. 8-bit I²C write example

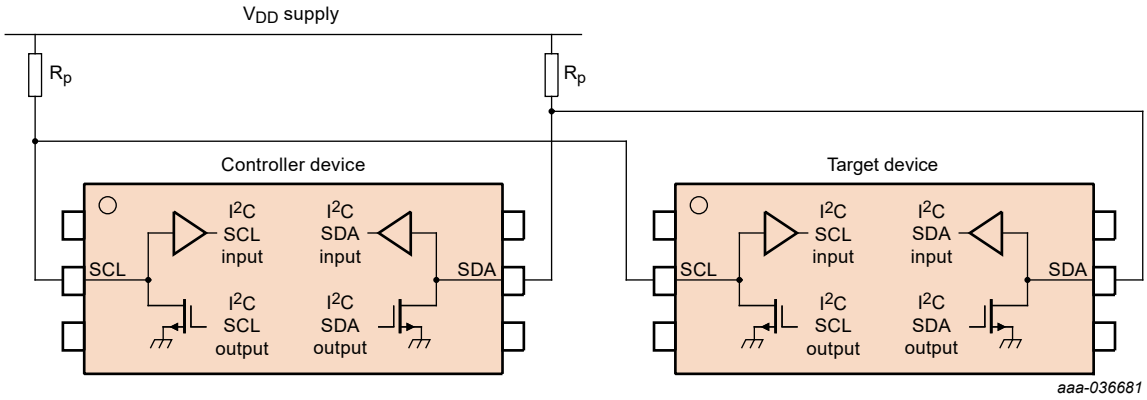
Fig. 5 shows an example controller and target device, with the controller transmitting an 8-bit write. The controller device is providing the system clock by pulsing SCL through its internal MOSFET. The controller is also communicating data to the target device by pulling its respective SDA MOSFET low when SCL is low and is also listening back for an acknowledge after the byte has been transmitted to the target device.

The target device listens to the controller through an internal buffer and acknowledges the controllers request to write to the 7-bit target address that was communicated.

The components of an I²C transaction are further explained in Fig. 5.



1. **Start condition**: The controller pulls the SDA line low while SCL is high, signaling the start of communication.
2. **Address frame:** The 7-bit (or possibly 10-bit) sequence that is unique to each target device.
3. **Read/write bit:** The 8th bit follows the address frame to signal to the target device whether the controller is requesting or writing data.
4. **ACK/NACK bit:** Every 8-bit frame in a message is followed by an ACK/NACK. The target device pulls down SDA signal to inform the controller that the data frame was successfully received.
5. **Stop condition:** The controller releases the SDA line from low to high, while SCL is high.



Controller		Target
SCL	**SDA**	**SCL**
MOSFET: Pulsing SCL	MOSFET: Communicating Address frame Buffer: Listening for "target" ACK	Buffer: Receiving data MOSFET: Sending ACK

Fig. 5. I²C components of 8-bit write

Several important conditions for valid I²C data transfer include the following:

- SDA transitions must only occur when SCL is low
- As SDA is sampled while SCL is high (target device), the SDA state must be stable throughout 'high' period of SCL
- Any changes to SDA while SCL is high can be interpreted as other conditions, such as Start and Stop, therefore can create some erroneous behavior.

5. I^2C timing specifications

In Table 2, some key parameters and specifications associated with the I^2C communication are shown. More are also included in the NXP I^2C user manual, where it lists the different specifications and required timing metrics for all different I^2C modes. This manual enables designers to comply with industry standards and build I^2C compatible devices.

Some example timing definitions are explained below.

Symbol	Parameter	Typical units
f_{SCL}	SCL clock frequency	kHz
t_{LOW}	Low period of SCL signal	µs
t_{HIGH}	Higher period of SCL signal	µs
t_r	Rise time (SDA)	µs
t_f	Fall time (SDA)	µs
$t_{SU(STA)}$	Set-up time for repeated start	µs
$t_{HD(STA)}$	Hold time for start/repeated start	µs
$t_{SU(DAT)}$	Data set-up time	µs
$t_{HD(DAT)}$	Data hold time	µs
$t_{SU(STOP)}$	Set-up time for stop condition	µs

I^2C timing specifications for the different modes of I^2C can be found at: here

5.1. Re-start/start hold time: $t_{HD(STA)}$

The hold time for a start or restart condition is the minimum amount of time data should go low before SCL goes low. The first point is measured at 30% amplitude of SDA when it is transitioning from high-to-low or falling. The second point is measured at 70% amplitude of SCL when it is transitioning from high-to-low or falling. The delta between the second and first point is calculated as the Hold time.

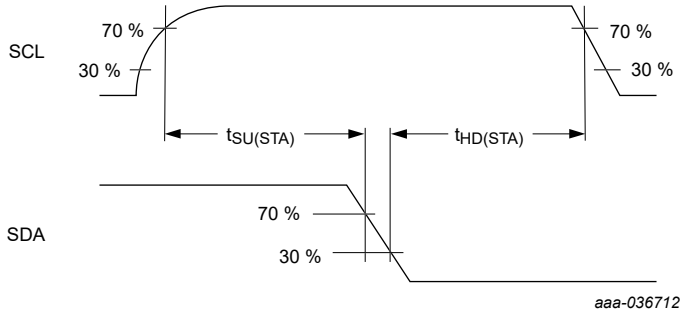


Fig. 6. Hold time: time from SDA (falling) 30% amplitude to SCL (falling) 70% amplitude

5.2. Re-start setup time: $t_{SU(STA)}$

Setup time is applicable for restart conditions. It is the time SDA must remain high before initiating a restart. The first point is measure at 70% amplitude of SCL when it is transitioning from low-to-high. The second point is measured at 70% amplitude of SDA when it is transitioning from high-to-low. The delta between these two points is labeled as the Setup time. See Fig. 6

5.3. Rise and fall time: t_r, t_f

Rise times are calculated as the time it takes for waveforms to reach from 30-to-70% amplitude levels, while fall times are calculated from 70-to-30% amplitude levels.

Rise time: Time from 30-to-70% amplitude

Fall time: Time from 70-to-30% amplitude

Fig. 7. Rise and fall times

5.4. Data setup time: $t_{SU(DAT)}$

Data setup time can be expressed as the time it takes SDA to reach final state before the rising edge of SCL. If SDA has transitioned from Low-to-High, the points measured are 70% SDA amplitude to 30% SCL amplitude. If SDA has transitioned from High-to-Low, the points measure are 30% SDA amplitude to 30% SCL amplitude.

Data hold is the minimum amount of time required for SDA to hold after SCL falls. It can be calculated as the time from 30% SCL to 70% SDA.

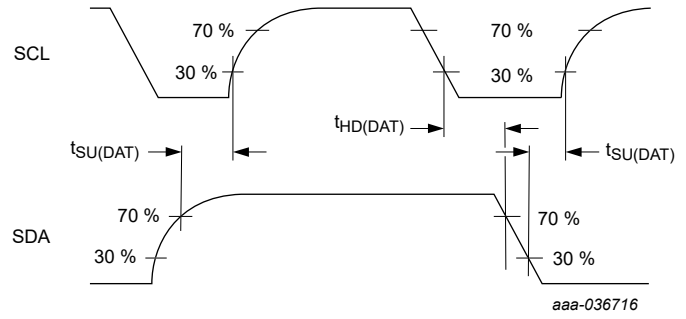


Fig. 8. Data setup time and data hold time

5.5. Stop setup time: $t_{SU(STOP)}$

Stop Setup time specification is the required minimum amount of time from when SCL is brought high to when SDA is brought high. And this is calculated as 70% amplitude of SCL during rising transition to 30% SDA on rising transition.

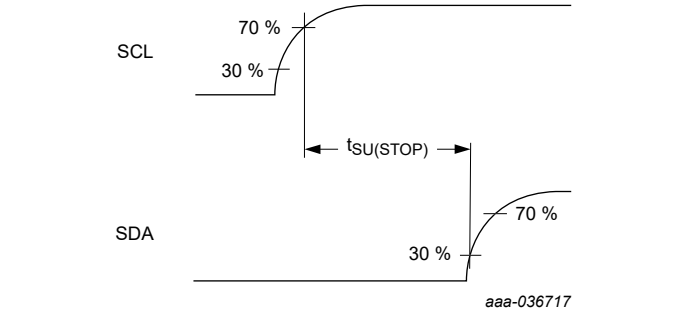


Fig. 9. Stop setup time

6. I²C transmission examples

In the examples below, I²C write and read transmissions are illustrated for target devices with either single or multiple internal registers.

6.1. 8-bit reigster write - single internal register

An example of an 8-bit register write is provided in Fig. 10. After the controller initiates a start, a 7-bit target (device) address is communicated to all targets that are connected to the BUS, this is followed by a write or read bit. In this example the controller is writing a byte a data to the specified target. After the byte is communicated, the target with matching address acknowledges the controller by pulling SDA low. If there are other target devices on the same bus they will send NACKs, which does not affect the SDA signal. The controller then clocks out 8-bit data, in this example 0 × 67 hexadecimal, after of which the target device again acknowledges. The controller then ends the I²C transmission through a stop command.

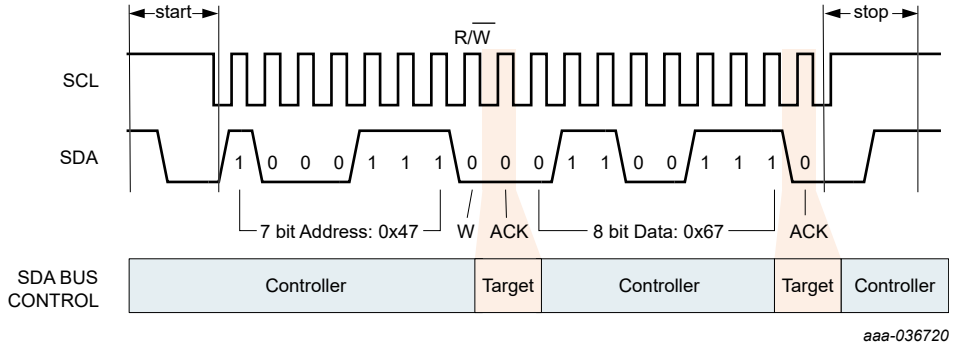


Fig. 10. 8-bit I²C write example

If the target device has multiple registers, I²C communication will include accessing to those Register Bytes.

6.2. Target device register write

Fig. 11 displays an I²C write to a device with multiple internal registers. As a result, there are multiple Register Bytes that the "Controller" device can access.

The controller initiates a start, followed by the 7-bit target address, which refers to a valid target address on the I²C bus. The read/write bit is set to '0', which informs the target device of a requested write. After the target device acknowledges, the controller sends the Register byte that it wants to access, which the target device once again acknowledges. The last 8-bits of data, referred to as the Data byte is the value that the controller writes to the register byte. The target device acknowledges once again, and the controller ends transmission with a stop command.

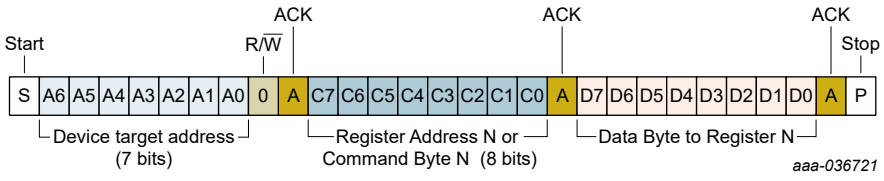


Fig. 11. Target register write

For an I^2C read, the transaction is slightly different as it includes more steps. One of the major differences is the 'repeated start' located at the center of the transaction.

Just as in a write the controller initiates a start, followed by the 7-bit target address with Read/Write bit set to '0'. After the target device acknowledges the controller, the controller then sends the register byte -- this is the register that the controller wants to read. After the target device acknowledges, the controller sends a repeated start followed by the target device address with Read/Write bit set to '1'. The target device again acknowledges, and controller releases SDA bus. The controller still provides SCL clock, however the target device updates SDA so the controller can capture the data. If the controller is expecting multiple bytes of data it will send an ACK, however in this case it only expected 1 byte of data and sent a NACK to target device followed by a STOP command.

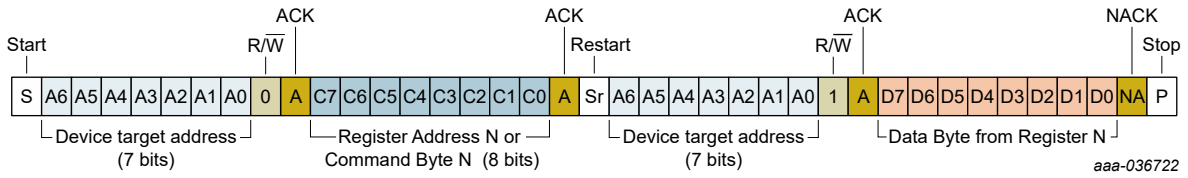
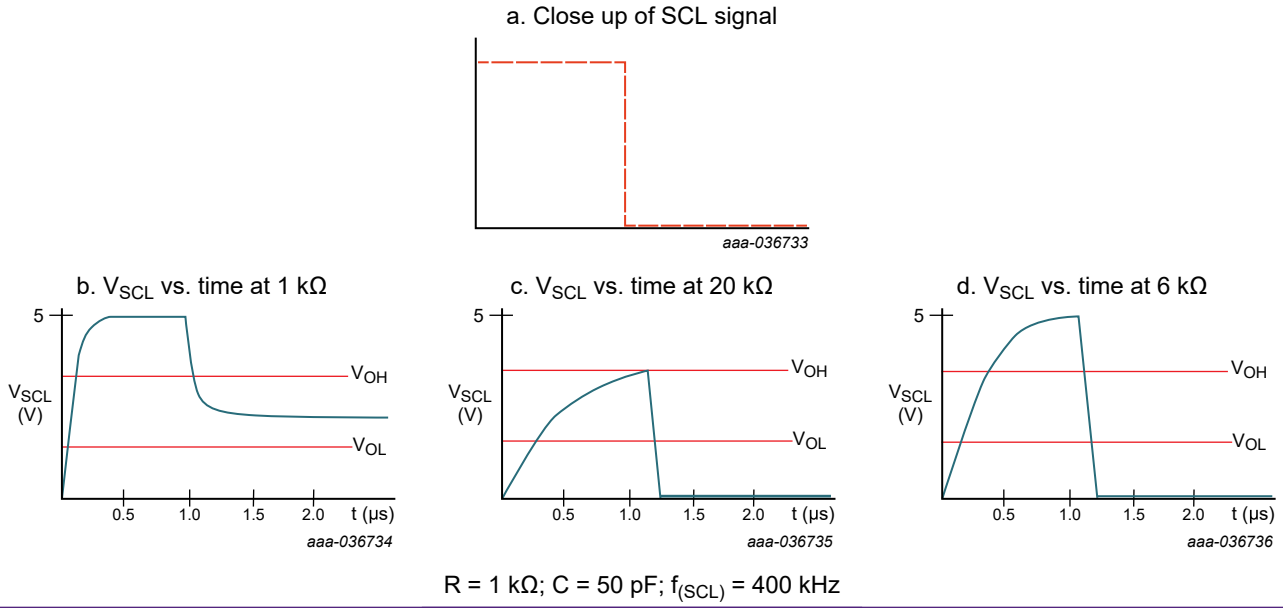


Fig. 12. Target register read

Now that example I^2C transmissions have been explored, this section will touch on the physical design related to the I^2C bus. This includes choosing suitable pullup resistors and understanding how signal capacitance (from PCB parasitics or input capacitance) can affect I^2C performance.

Fig. 13 illustrates a closeup of the SCL signal. The signal period is captured for a 400 kHz square waveform to drive SCL (fast-mode), with a 2.5 µs period.



a. 400 kHz signal (2.5 µs)
b. Displays the use with a 1 kΩ resistor that was sized below recommended guidelines for Fast-mode. When the controller releases the line it displays a fast rise time, however, the drive strength of the internal FET is limited and cannot pull the line below V_{IL} values.
c. The pullup resistors are sized too large, therefore the signal does not have time to produce a valid high.
d. A waveform that will satisfy all requirements for Fast mode.

Fig. 13. Effect of pullup resistor on waveform

The next sections will focus on equations that can be used to find pullup resistor values for a specific bus capacitance and V_{DD} voltage.

7. R_p maximum resistor value: $R_{p(max)}$

Equation 1 represents the procedure on how to calculate $R_{p(max)}$, which is the maximum resistor value for a specific bus capacitance. Bus capacitance is generally calculated as the total capacitance from device pins, PCB parasitics, and cable capacitance.

The equation for charging an RC circuit is used, along with rise time metrics – the equations are rearranged to find the time needed for 30% magnitude and 70% magnitude.

$$V(t) = V_{DD}\left(1 - e^{\frac{-t}{RC}}\right) \tag{1}$$

$$V(t1) = 0.3 \times V_{DD} = V_{DD}\left(1 - e^{\frac{-t1}{RC}}\right) \quad \text{then } t1 = 0.35667 \times RC \tag{2}$$

$$V(t2) = 0.7 \times V_{DD} = V_{DD}\left(1 - e^{\frac{-t2}{RC}}\right) \quad \text{then } t2 = 1.20397 \times RC \tag{3}$$

$$t_r = t_2 - t_1 = 0.8473 \times RC \tag{4}$$

Rearranging the equation:

$$R_{p(max)} = \frac{t_r}{0.8473 \times C_b} \tag{5}$$

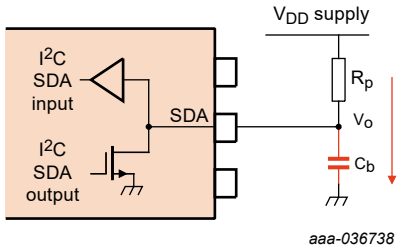
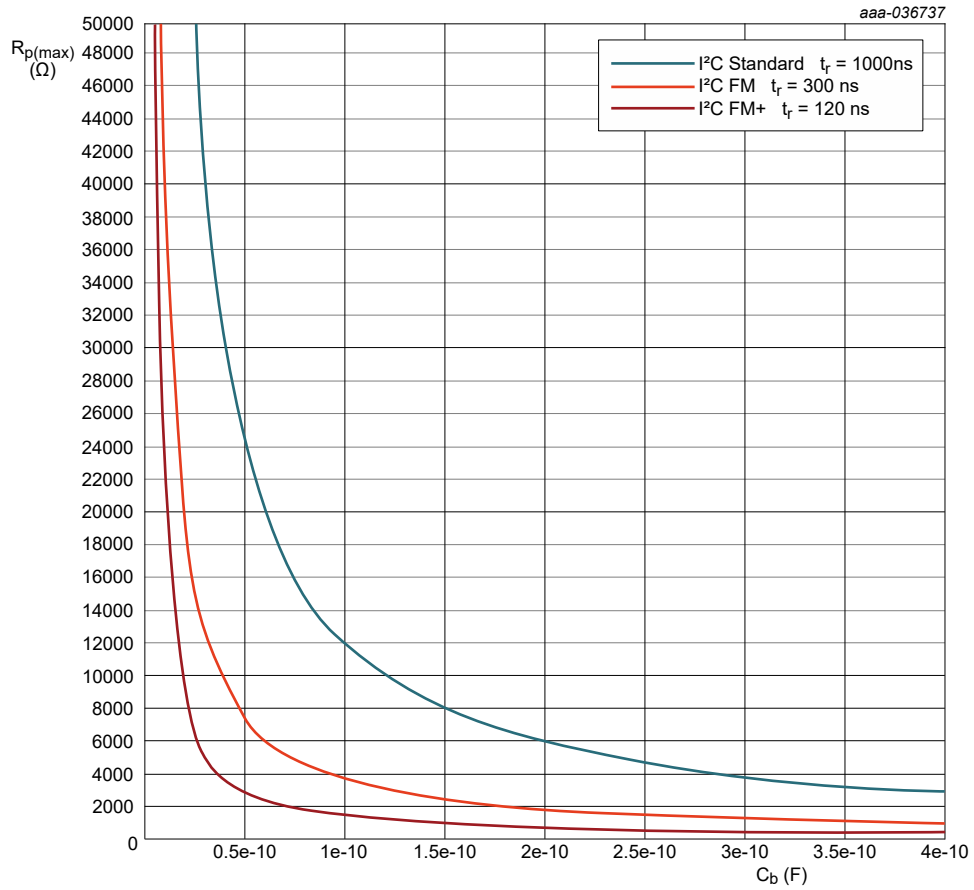


Fig. 14. Circuit



a. Graph

Fig. 15. $R_{p(max)}$ vs. Bus capacitance

8. R_p minimum resistor value: $R_{p(min)}$

To find $R_{p(min)}$, a simple KCL equation can be generated from the circuit shown in Fig. 16. From this equation, it is easy to verify the relationship between MOSFET sink capability, VDD, VOL and the pullup resistance value. As an example, if the MOSFET is only capable of sinking 3mA, and a minimum VOL value of 0.4V is required, the equation would produce a minimum resistor value of 1.533kOhm. For correct operation, the user must ensure that the pullup resistor values are above this value.

The graph shows the minimum resistance that can be used across the different I^2C modes for specific VDD voltages and sink currents.

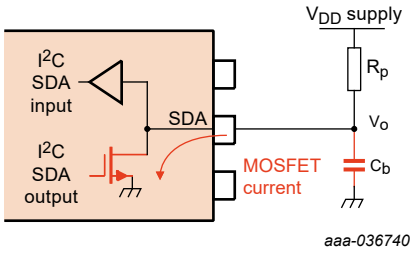
$$\left(V_{DD} - V_{OL(max)}\right) / R_{p(min)} = I_{OL} \tag{6}$$

Rearranging the equation:

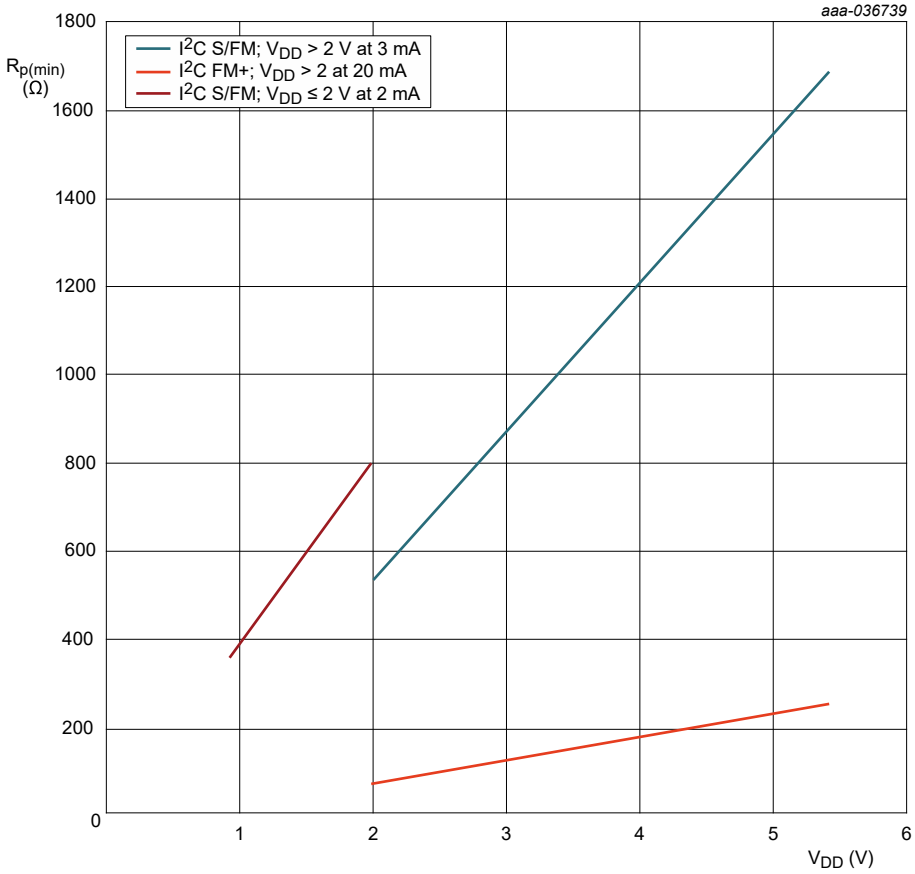
$$R_{p(min)} = \left(V_{DD} - V_{OL(max)}\right) / I_{OL} \tag{7}$$

Table 2. SCL/SDA I/O specifications: V_{OL}, I_{OL}

Symbol	Parameter	Conditions	Standard mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
V_{OL1}	LOW-level output voltage 1	open-drain at 3 mA sink; V_{DD} > 2 V	0	0.4	0	0.4	0	0.4	V
V_{OL2}	LOW-level output voltage 2	open-drain at 3 mA sink; V_{DD} > 2 V	-	-	0	$0.2V_{DD}$	0	$0.2V_{DD}$	V
I_{OL}	LOW-level output current	V_{OL} = 0.4 V	3	-	3	-	20	-	mA



a. Circuit



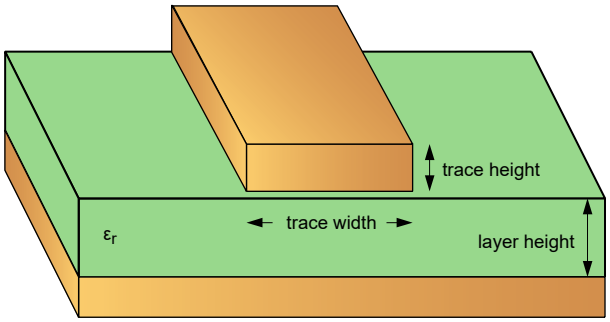
b. Graph

Fig. 16. $R_{p(min)}$ vs. V_{DD} (V)

9. PCB net capacitance: capacitance of a microstrip

As previously mentioned I^2C bus capacitance is limited at 400pF for most modes, therefore important considerations should be taken when designing the PCB board layout. While the equation for parallel plate capacitance may be readily known, C= εA/d, it's important to note an equation that resembles PCB layout. The illustration in Fig. 17, shows a signal net separated from an adjacent plane by internal PCB dielectric.

The equations listed are known as the capacitance of a microstrip, and this equation better resembles actual PCB layout. The resulting value is reported in capacitance-per unit-length. In the Fig. 17 below, both imperial and metric equations are shown for reference.



Metric version:

$$C_L = \frac{0.264\left(1.41 + \epsilon_r\right)}{\ln\frac{5.98 \times h}{0.8 \times w + t}} \quad (8)$$

Where
- C_L = capacitance per unit length (pF/cm);
- w = width of the trace in mm;
- h = height of the trace above the power plane in mm;
- t = thickness of trace in mm;
- ε_r = dielectric constant of material between trace and power plane (FR-4 is ~4.5).

Imperial version:

$$C_L = \frac{0.67\left(1.41 + \epsilon_r\right)}{\ln\frac{5.98 \times h}{0.8 \times w + t}} \quad (9)$$

Where
- C_L = capacitance per unit length (pF/inch);
- w = width of the trace in mil;
- h = height of the trace above the power plane in mil;
- t = thickness of trace in mil;
- ε_r = dielectric constant of material between trace and power plane (FR-4 is ~4.5).

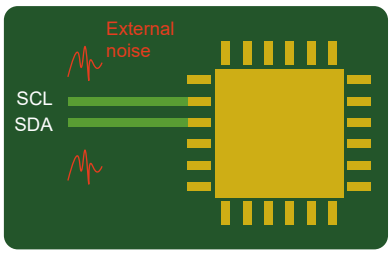
Fig. 17. Capacitance of a microstrip

10. Reducing external noise

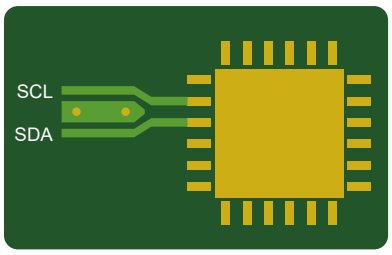
Removing extra copper from the PCB design can reduce net capacitance, however this could leave the signal nets exposed to any possible disturbances, such as noise or inductive transients. Although the slow speeds of the I^2C protocol make it more robust against these types of noise artifacts, it is good practice to minimize the coupling of any unwanted signals.

More specifically, PCBs should be designed to effectively shield the SCL and SDA lines from any inductive transients that could potentially couple onto the signal nets.

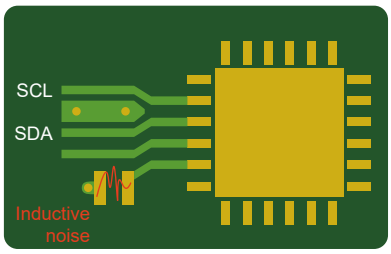
Fig. 18 illustrates 4 different designs with I^2C nets that showcase the tradeoffs between shielding and design area.



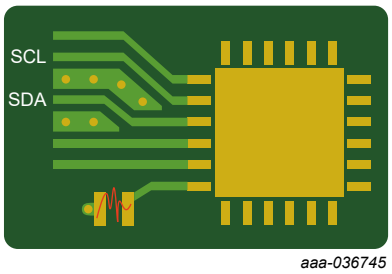
(a) This first device shows no shielding, so any inductive or fast switching nets can couple into the communication lines



(b) This second device shows shielding between SCL and SDA, so that SCL rising/falling edges are not coupled onto SDA.



(c) This third device has a power management device with an inductor connected to one of its pins. The inductive switching could potentially couple into the communication lines, therefore adequate shielding between this element and the lines should be employed



(d) This fourth device shows better shielding between the communication lines and inductive element.

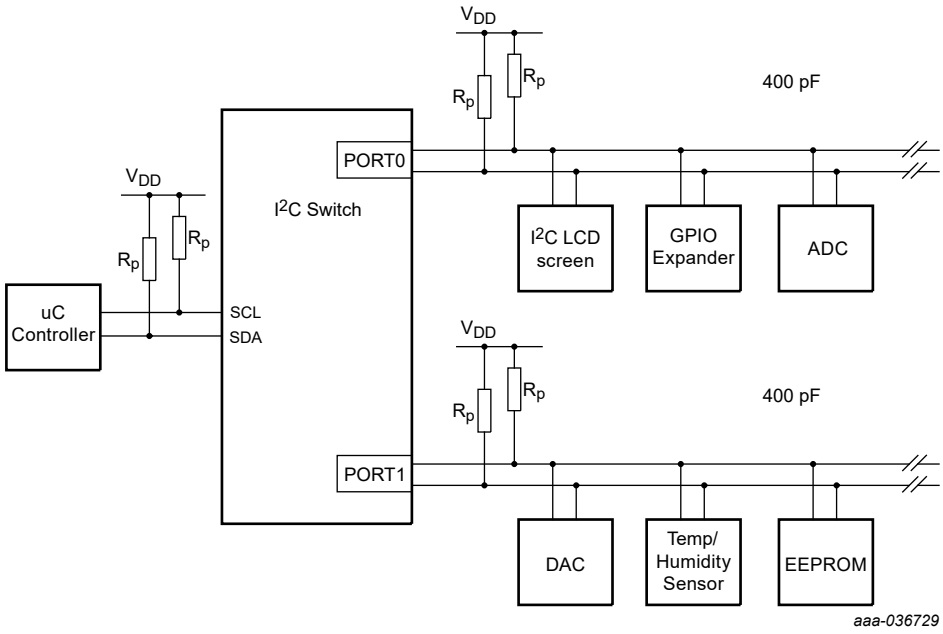
Fig. 18. Effective PCB shielding

It is important to note that noise and capacitance are traded with one another in PCB design. Shielding protects from noise but adds extra capacitance between copper nets. However, there are ways to reduce this capacitance while maintaining transient suppression, such as increasing space from net to shield polygon --this would then tradeoff with board area.

11. Product solutions - exceeding the I²C bus capacitance limit

In practice, when designing systems with large amounts of controllers and target devices, the bus capacitance will be exceeded before reaching the numerical limit of devices allowed on the system bus.

Fig. 19 shows the use of an I²C switch, that allows the separation of target devices through PORT0 and PORT1. The controller can switch to any Port during operation. Each port can support the 400pF requirement, allowing the user to incorporate large amounts of controller or I²C devices. If there are certain target devices with limited unique addresses, this circuit would also allow the user to exceed this limit, as the user can put the maximum amount at each port.



Note: Capacitance load on upstream port (controller side) is considered when determining full capacitance of I²C Bus. Total Bus capacitance is calculated as the sum of upstream capacitance and downstream capacitance (activated PORT).

Fig. 19. I²C switch to increase BUS capacitance

Fig. 20 displays the use of an I²C repeater or buffer. The device is essentially buffering the communication lines that allow the bus to be split into 400 pF capacitance before and after the buffer. This would effectively increase the allowed capacitance to 2X its limited value. Additionally, the device can translate the V_{DD} voltages to different values. This may be needed when bridging two different I²C buses with different V_{DD} values or when communicating with target devices at different supply voltages.

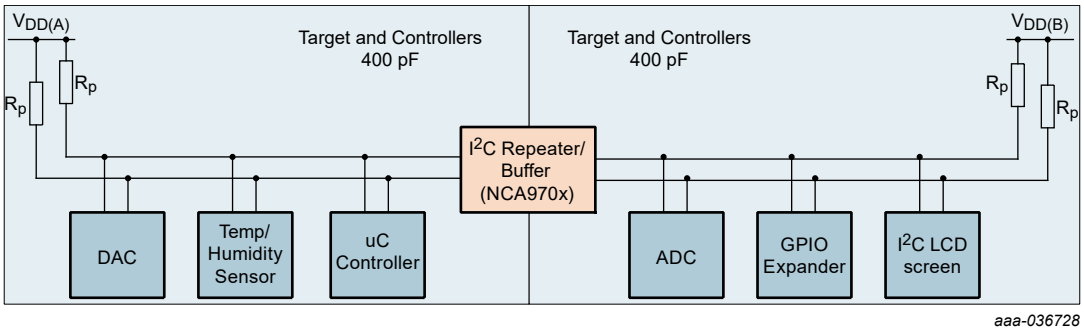


Fig. 20. I²C repeater to increase BUS capacitance

12. Dynamic addressing - exceeding the address pin limit

Target devices with limited number of address pins also create another challenge in I^2C systems. In Fig. 21 a Target device with only 1 address pin is shown. Due to the single address pin, the target device is limited to two unique addresses.

This means at most only two devices can connect to the same I^2C bus. Although limited, there may be scenarios when more of these target devices are needed in the system, such as sensor applications.

In Fig. 21, a GPIO expander is used to dynamically address eight similar target devices with only one address pin. The controller accomplishes this by initially setting all address pins high, and when the controller wants to communicate to a specific device, it would then set the device's address pin to LOW through the GPIO expander. It will then communicate with the target address that corresponds to this LOW address pin state. The microcontroller can use the GPIO expander to sequentially access each target device.

In this example it is assumed that the targeted device can be dynamically addressed, meaning it can change address states while powered. This may be possible for most target devices; however, testing would need to be performed to verify.

If the address values are latched in during power up, the GPIO expander can additionally be used to drive a reset mechanism (reset pin) that would reset the target devices before communication.

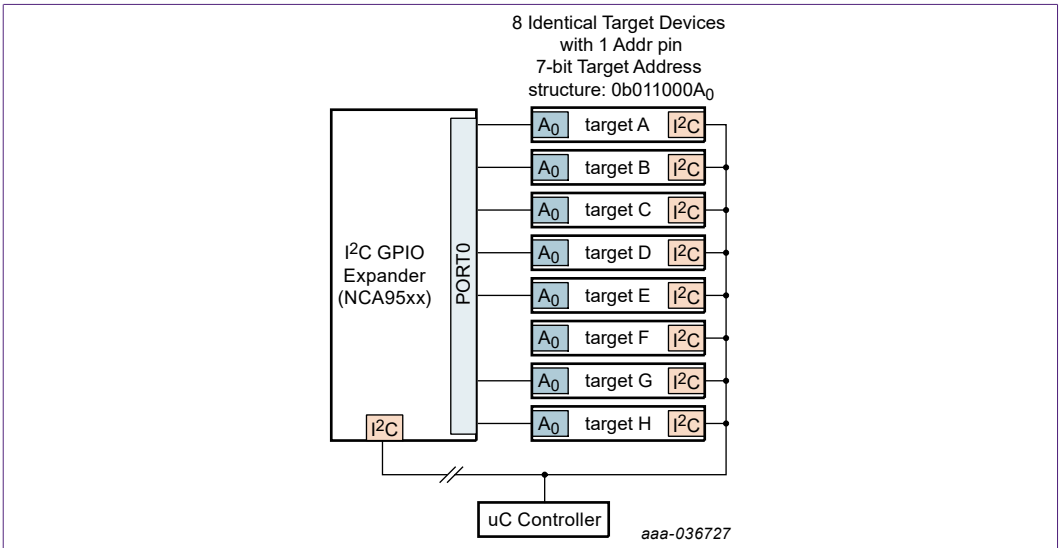


Fig. 21. GPIO expander to dynamically address target devices

13. Revision history

Revision	Date	Description
1	20230615	Application note
2	20230630	Errata: Table 1 corrected; Fig. 17 amended.

14. Legal information

Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. Nexperia does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, Nexperia does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. Nexperia takes no responsibility for the content in this document if provided by an information source outside of Nexperia.

In no event shall Nexperia be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, Nexperia's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of Nexperia.

Right to make changes — Nexperia reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — Nexperia products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an Nexperia product can reasonably be expected to result in personal injury, death or severe property or environmental damage. Nexperia and its suppliers accept no liability for inclusion and/or use of Nexperia products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. Nexperia makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using Nexperia products, and Nexperia accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the Nexperia product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Nexperia does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using Nexperia products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). Nexperia does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

Contents

For more information, please visit: http://www.nexperia.com
For sales office addresses, please send an email to: salesaddresses@nexperia.com
Date of release: 30 June 2023