UM10204

I²C-bus specification and user manual

Rev. 7.0 — 1 October 2021 User manual

1 Introduction

The I²C-bus is a de facto world standard that is now implemented in over 1000 different ICs manufactured by more than 50 companies. Additionally, the versatile I²C-bus is used in various control architectures such as System Management Bus (SMBus), Power Management Bus (PMBus), Intelligent Platform Management Interface (IPMI), Display Data Channel (DDC) and Advanced Telecom Computing Architecture (ATCA).

This document assists device and system designers to understand how the I²C-bus works and implement a working application. Various operating modes are described. It contains a comprehensive introduction to the I²C-bus data transfer, handshaking and bus arbitration schemes. Detailed sections cover the timing and electrical specifications for the I²C-bus in each of its operating modes.

Designers of I²C-compatible chips should use this document as a reference and ensure that new devices meet all limits specified in this document. Designers of systems that include I²C devices should review this document and also refer to individual component data sheets.

Readers looking to develop I²C based solutions may also be interested in I3C, introduced by the MIPI Alliance in 2017, with NXP's involvement and contributions. MIPI I3C offers backward compatibility with I²C, increased speed and low power consumption, and a royalty-free version is available for implementers. More information is included at the end of this document in Section 9.

2 I²C-bus features

In consumer electronics, telecommunications and industrial electronics, there are often many similarities between seemingly unrelated designs. For example, nearly every system includes:

- Some intelligent control, usually a single-chip microcontroller
- General-purpose circuits like LCD and LED drivers, remote I/O ports, RAM, EEPROM, real-time clocks or A/D and D/A converters
- Application-oriented circuits such as digital tuning and signal processing circuits for radio and video systems, temperature sensors, and smart cards

To exploit these similarities to the benefit of both systems designers and equipment manufacturers, as well as to maximize hardware efficiency and circuit simplicity, Philips Semiconductors (now NXP Semiconductors) developed a simple bidirectional 2-wire bus for efficient inter-IC control. This bus is called the Inter IC or I²C-bus. All I²C-bus compatible devices incorporate an on-chip interface which allows them to communicate directly with each other via the I²C-bus. This design concept solves the many interfacing problems encountered when designing digital control circuits.

Here are some of the features of the I²C-bus:

- Only two bus lines are required; a serial data line (SDA) and a serial clock line (SCL).
- Each device connected to the bus is software addressable by a unique address and simple controller/target relationships exist at all times; controllers can operate as controller-transmitters or as controller-receivers.
- It is a true multi-controller bus including collision detection and arbitration to prevent data corruption if two or more controllers simultaneously initiate data transfer.

- Serial, 8-bit oriented, bidirectional data transfers can be made at up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, up to 1 Mbit/s in Fast-mode Plus, or up to 3.4 Mbit/s in the High-speed mode.
- Serial, 8-bit oriented, unidirectional data transfers up to 5 Mbit/s in Ultra Fast-mode
- On-chip filtering rejects spikes on the bus data line to preserve data integrity.
- The number of ICs that can be connected to the same bus is limited only by a maximum bus capacitance. More capacitance may be allowed under some conditions. Refer to Section 7.2.

Figure 1 shows an example of I²C-bus applications.

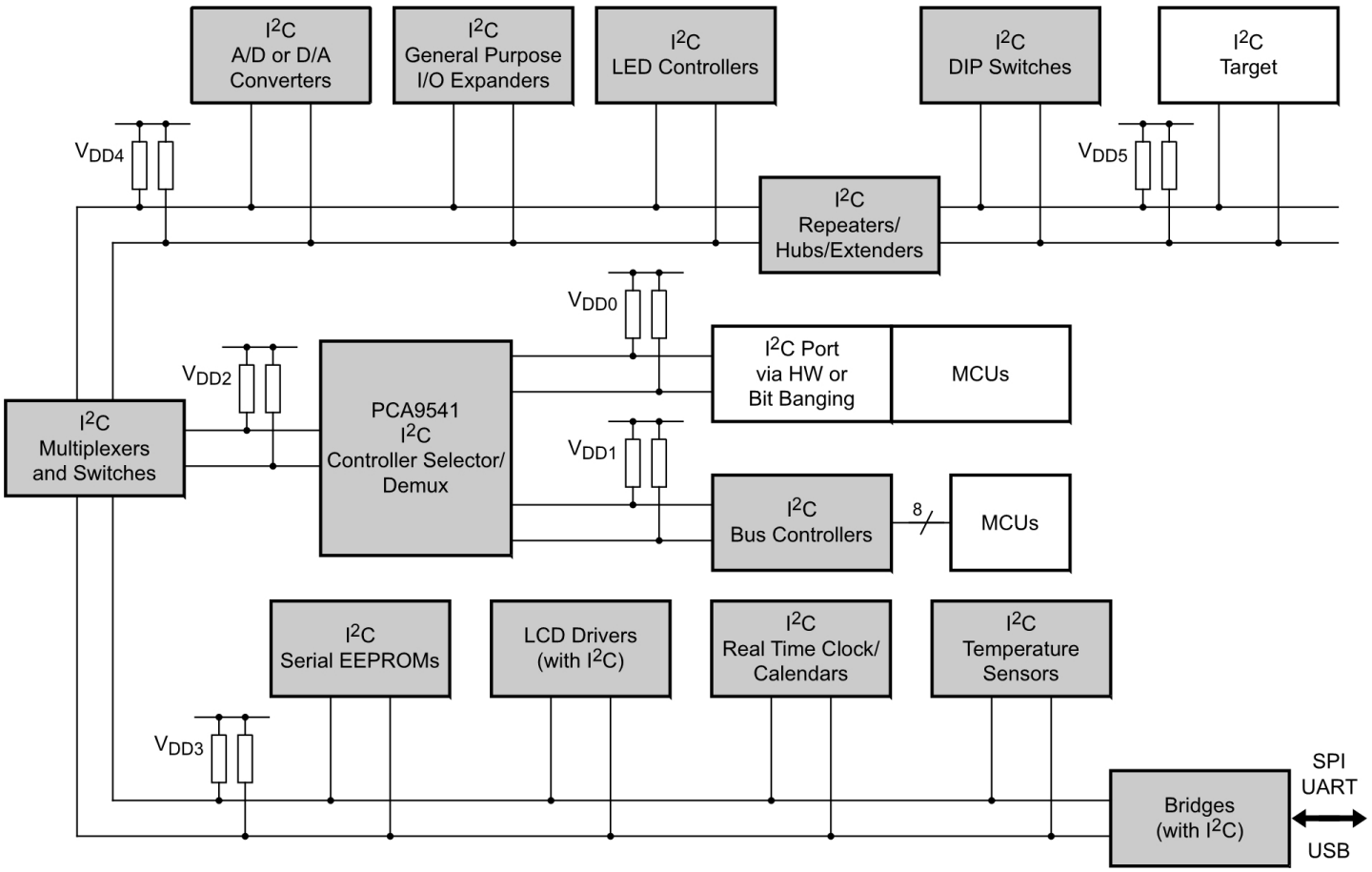


Figure 1. Example of I²C-bus applications

2.1 Designer benefits

I²C-bus compatible ICs allow a system design to progress rapidly directly from a functional block diagram to a prototype. Moreover, since they 'clip' directly onto the I²C-bus without any additional external interfacing, they allow a prototype system to be modified or upgraded simply by 'clipping' or 'unclipping' ICs to or from the bus.

Here are some of the features of I²C-bus compatible ICs that are particularly attractive to designers:

- Functional blocks on the block diagram correspond with the actual ICs; designs proceed rapidly from block diagram to final schematic.
- No need to design bus interfaces because the I²C-bus interface is already integrated on-chip.

- Integrated addressing and data-transfer protocol allow systems to be completely software-defined.
- The same IC types can often be used in many different applications.
- Design-time reduces as designers quickly become familiar with the frequently used functional blocks represented by I^2C-bus compatible ICs.
- ICs can be added to or removed from a system without affecting any other circuits on the bus.
- Fault diagnosis and debugging are simple; malfunctions can be immediately traced.
- Software development time can be reduced by assembling a library of reusable software modules.

In addition to these advantages, the CMOS ICs in the I^2C-bus compatible range offer designers special features which are particularly attractive for portable equipment and battery-backed systems.

They all have:

- Extremely low current consumption
- High noise immunity
- Wide supply voltage range
- Wide operating temperature range.

2.2 Manufacturer benefits

I^2C-bus compatible ICs not only assist designers, they also give a wide range of benefits to equipment manufacturers because:

- The simple 2-wire serial I^2C-bus minimizes interconnections so ICs have fewer pins and there are not so many PCB tracks; result — smaller and less expensive PCBs.
- The completely integrated I^2C-bus protocol eliminates the need for address decoders and other ‘glue logic’.
- The multi-controller capability of the I^2C-bus allows rapid testing and alignment of end-user equipment via external connections to an assembly line.
- The availability of I^2C-bus compatible ICs in various leadless packages reduces space requirements even more.

These are just some of the benefits. In addition, I^2C-bus compatible ICs increase system design flexibility by allowing simple construction of equipment variants and easy upgrading to keep designs up-to-date. In this way, an entire family of equipment can be developed around a basic model. Upgrades for new equipment, or enhanced-feature models (that is, extended memory, remote control, etc.) can then be produced simply by clipping the appropriate ICs onto the bus. If a larger ROM is needed, it is simply a matter of selecting a microcontroller with a larger ROM from our comprehensive range. As new ICs supersede older ones, it is easy to add new features to equipment or to increase its performance by simply unclipping the outdated IC from the bus and clipping on its successor.

2.3 IC designer benefits

Designers of microcontrollers are frequently under pressure to conserve output pins. The I^2C protocol allows connection of a wide variety of peripherals without the need for separate addressing or chip enable signals. Additionally, a microcontroller that includes an I^2C interface is more successful in the marketplace due to the wide variety of existing peripheral devices available.

3 The I²C-bus protocol

3.1 Standard-mode, Fast-mode and Fast-mode Plus I²C-bus protocols

Two wires, serial data (SDA) and serial clock (SCL), carry information between the devices connected to the bus. Each device is recognized by a unique address (whether it is a microcontroller, LCD driver, memory or keyboard interface) and can operate as either a transmitter or receiver, depending on the function of the device. An LCD driver may be only a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as controllers or targets when performing data transfers (see Table 2). A controller is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a target.

Table 2. Definition of I²C-bus terminology

Term	Description
Transmitter	the device which sends data to the bus
Receiver	the device which receives data from the bus
Controller	the device which initiates a transfer, generates clock signals and terminates a transfer
Target	the device addressed by a controller
Multi-controller	more than one controller can attempt to control the bus at the same time without corrupting the message
Arbitration	procedure to ensure that, if more than one controller simultaneously tries to control the bus, only one is allowed to do so and the winning message is not corrupted
Synchronization	procedure to synchronize the clock signals of two or more devices

The I²C-bus is a multi-controller bus. This means that more than one device capable of controlling the bus can be connected to it. As controllers are usually microcontrollers, let us consider the case of a data transfer between two microcontrollers connected to the I²C-bus (see Figure 2).

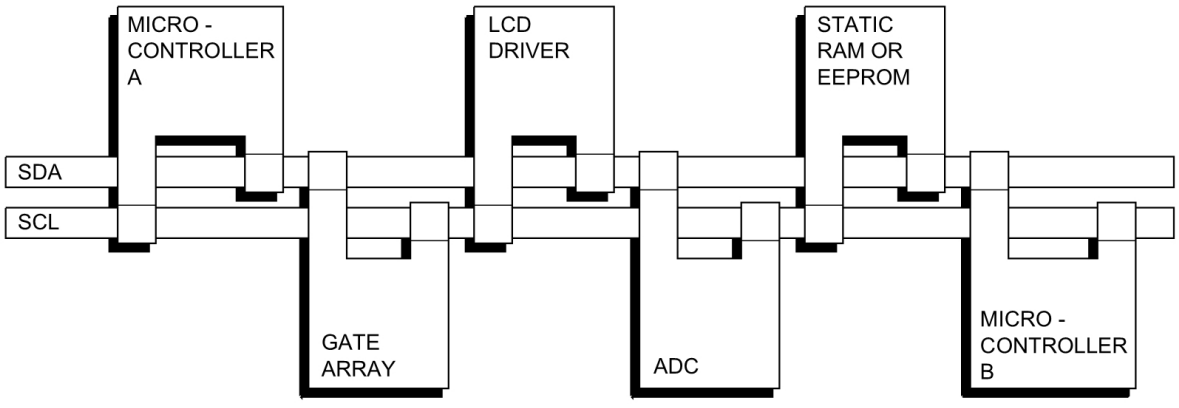


Figure 2. Example of an I²C-bus configuration using two microcontrollers

This example highlights the controller-target and receiver-transmitter relationships found on the I²C-bus. Note that these relationships are not permanent, but only depend on the direction of data transfer at that time. The transfer of data would proceed as follows:

1. Suppose microcontroller A wants to send information to microcontroller B:
   - microcontroller A (controller), addresses microcontroller B (target)
   - microcontroller A (controller-transmitter), sends data to microcontroller B (target-receiver)
   - microcontroller A terminates the transfer.
2. If microcontroller A wants to receive information from microcontroller B:
   - microcontroller A (controller) addresses microcontroller B (target)
   - microcontroller A (controller-receiver) receives data from microcontroller B (target-transmitter)
   - microcontroller A terminates the transfer.

Even in this case, the controller (microcontroller A) generates the timing and terminates the transfer.

The possibility of connecting more than one microcontroller to the I^2C-bus means that more than one controller could try to initiate a data transfer at the same time. To avoid the chaos that might ensue from such an event, an arbitration procedure has been developed. This procedure relies on the wired-AND connection of all I^2C interfaces to the I^2C-bus.

If two or more controllers try to put information onto the bus, the first to produce a 'one' when the other produces a 'zero' loses the arbitration. The clock signals during arbitration are a synchronized combination of the clocks generated by the controllers using the wired-AND connection to the SCL line (for more detailed information concerning arbitration see Section 3.1.8).

Generation of clock signals on the I^2C-bus is always the responsibility of controller devices; each controller generates its own clock signals when transferring data on the bus. Bus clock signals from a controller can only be altered when they are stretched by a slow target device holding down the clock line or by another controller when arbitration occurs.

Table 3 summarizes the use of mandatory and optional portions of the I^2C-bus specification and which system configurations use them.

Table 3. Applicability of I^2C-bus protocol features

M = mandatory; O = optional; n/a = not applicable.

Feature	Configuration		
	Single controller	Multi-controller	Target[1]
START condition	M	M	M
STOP condition	M	M	M
Acknowledge	M	M	M
Synchronization	n/a	M	n/a
Arbitration	n/a	M	n/a
Clock stretching	O[2]	O[2]	O
7-bit target address	M	M	M
10-bit target address	O	O	O
General Call address	O	O	O
Software Reset	O	O	O
START byte	n/a	O[3]	n/a

Table 3. Applicability of I²C-bus protocol features...*continued*

M = mandatory; O = optional; n/a = not applicable.

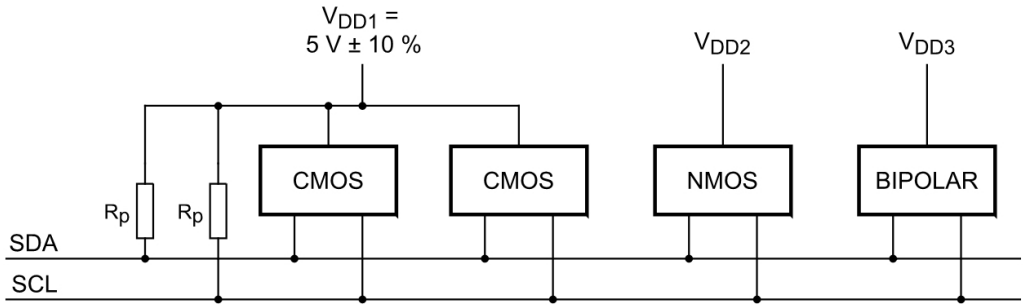
Feature	Configuration		
	Single controller	Multi-controller	Target[1]
Device ID	n/a	n/a	O

[1] Also refers to a controller acting as a target.
[2] Clock stretching is a feature of some targets. If no targets in a system can stretch the clock (hold SCL LOW), the controller need not be designed to handle this procedure.
[3] 'Bit banging' (software emulation) multi-controller systems should consider a START byte. See Section 3.1.15.

3.1.1 SDA and SCL signals

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via a current-source or pull-up resistor (see Figure 3). When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function. Data on the I²C-bus can be transferred at rates of up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, up to 1 Mbit/s in Fast-mode Plus, or up to 3.4 Mbit/s in the High-speed mode. The bus capacitance limits the number of interfaces connected to the bus.

For a single controller application, the controller's SCL output can be a push-pull driver design if there are no devices on the bus which would stretch the clock.



V_{DD2}, V_{DD3} are device-dependent (for example, 12 V).

Figure 3. Devices with various supply voltages sharing the same bus

3.1.2 SDA and SCL logic levels

Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I²C-bus, the levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of V_{DD}. Input reference levels are set as 30 % and 70 % of V_{DD}; V_{IL} is $0.3V_{DD}$ and V_{IH} is $0.7V_{DD}$. See Figure 38, timing diagram. Some legacy device input levels were fixed at V_{IL} = 1.5 V and V_{IH} = 3.0 V, but all new devices require this 30 %/70 % specification. See Section 6 for electrical specifications.

3.1.3 Data validity

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (see Figure 4). One clock pulse is generated for each data bit transferred.

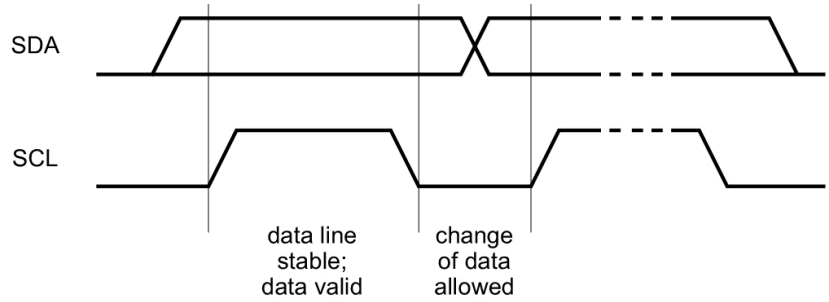


Figure 4. Bit transfer on the I²C-bus

3.1.4 START and STOP conditions

All transactions begin with a START (S) and are terminated by a STOP (P) (see Figure 5). A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition. A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

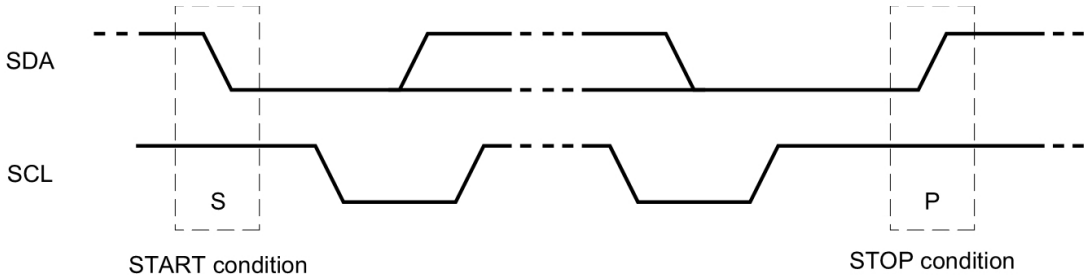


Figure 5. START and STOP conditions

START and STOP conditions are always generated by the controller. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition. This bus free situation is specified in Section 6.

The bus stays busy if a repeated START (Sr) is generated instead of a STOP condition. In this respect, the START (S) and repeated START (Sr) conditions are functionally identical. For the remainder of this document, therefore, the S symbol is used as a generic term to represent both the START and repeated START conditions, unless Sr is particularly relevant.

Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware. However, microcontrollers with no such interface have to sample the SDA line at least twice per clock period to sense the transition.

3.1.5 Byte format

Every byte put on the SDA line must be eight bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte must be followed by an Acknowledge bit. Data is transferred with the Most Significant Bit (MSB) first (see Figure 6). If a target cannot receive or transmit another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can hold the clock line SCL LOW to force the controller into a wait state. Data transfer then continues when the target is ready for another byte of data and releases clock line SCL.

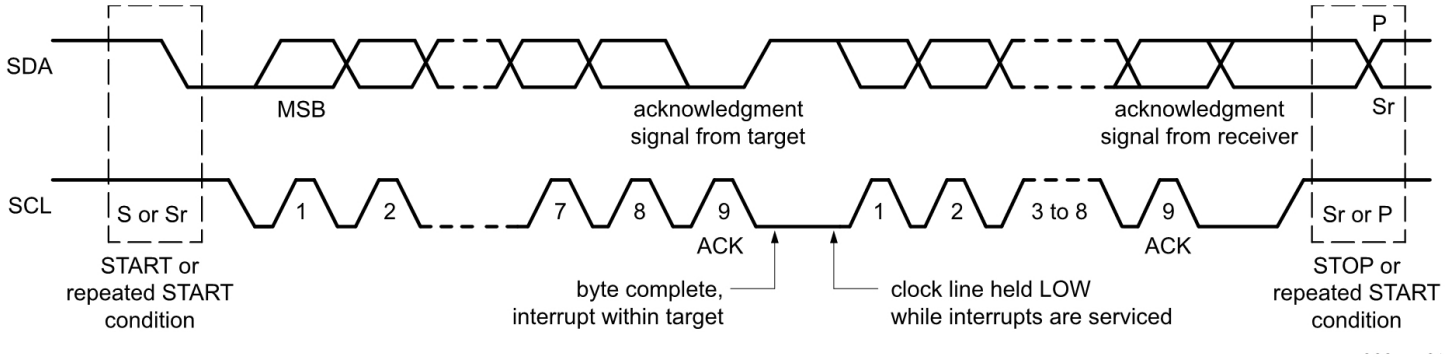


Figure 6. Data transfer on the I²C-bus

3.1.6 Acknowledge (ACK) and Not Acknowledge (NACK)

The acknowledge takes place after every byte. The acknowledge bit allows the receiver to signal the transmitter that the byte was successfully received and another byte may be sent. The controller generates all clock pulses, including the acknowledge ninth clock pulse.

The Acknowledge signal is defined as follows: the transmitter releases the SDA line during the acknowledge clock pulse so the receiver can pull the SDA line LOW and it remains stable LOW during the HIGH period of this clock pulse (see Figure 4). Set-up and hold times (specified in Section 6) must also be taken into account.

When SDA remains HIGH during this ninth clock pulse, this is defined as the Not Acknowledge signal. The controller can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer. There are five conditions that lead to the generation of a NACK:

1. No receiver is present on the bus with the transmitted address so there is no device to respond with an acknowledge.
2. The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the controller.
3. During the transfer, the receiver gets data or commands that it does not understand.
4. During the transfer, the receiver cannot receive any more data bytes.
5. A controller-receiver must signal the end of the transfer to the target transmitter.

3.1.7 Clock synchronization

Two controllers can begin transmitting on a free bus at the same time and there must be a method for deciding which takes control of the bus and complete its transmission. This is done by clock synchronization and arbitration. In single controller systems, clock synchronization and arbitration are not needed.

Clock synchronization is performed using the wired-AND connection of I²C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line causes the controllers concerned to start counting off their LOW period and, once a controller clock has gone LOW, it holds the SCL line in that state until the clock HIGH state is reached (see Figure 7). However, if another clock is still within its LOW period, the LOW to HIGH transition of this clock may not change the state of the SCL line. The SCL line is therefore held LOW by the controller with the longest LOW period. Controllers with shorter LOW periods enter a HIGH wait-state during this time.

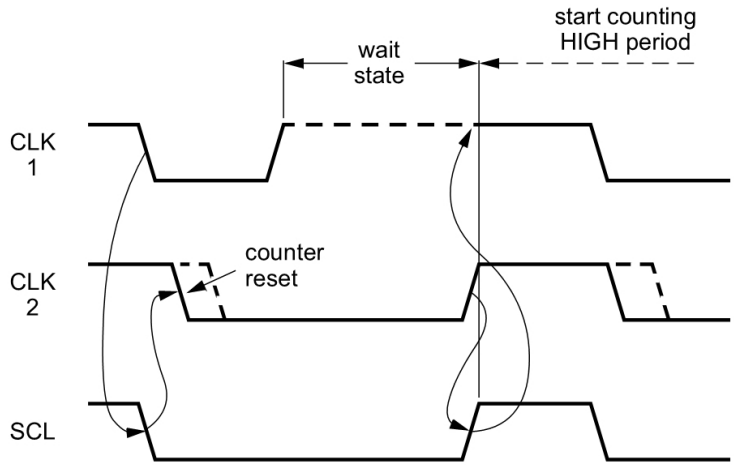


Figure 7. Clock synchronization during the arbitration procedure

When all controllers concerned have counted off their LOW period, the clock line is released and goes HIGH. There is then no difference between the controller clocks and the state of the SCL line, and all the controllers start counting their HIGH periods. The first controller to complete its HIGH period pulls the SCL line LOW again.

In this way, a synchronized SCL clock is generated with its LOW period determined by the controller with the longest clock LOW period, and its HIGH period determined by the one with the shortest clock HIGH period.

3.1.8 Arbitration

Arbitration, like synchronization, refers to a portion of the protocol required only if more than one controller is used in the system. Targets are not involved in the arbitration procedure. A controller may start a transfer only if the bus is free. Two controllers may generate a START condition within the minimum hold time ($t_{HD;STA}$) of the START condition which results in a valid START condition on the bus. Arbitration is then required to determine which controller will complete its transmission.

Arbitration proceeds bit by bit. During every bit, while SCL is HIGH, each controller checks to see if the SDA level matches what it has sent. This process may take many bits. Two controllers can actually complete an entire transaction without error, as long as the transmissions are identical. The first time a controller tries to send a HIGH, but detects that the SDA level is LOW, the controller knows that it has lost the arbitration and turns off its SDA output driver. The other controller goes on to complete its transaction.

No information is lost during the arbitration process. A controller that loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration and must restart its transaction when the bus is free.

If a controller also incorporates a target function and it loses arbitration during the addressing stage, it is possible that the winning controller is trying to address it. The losing controller must therefore switch over immediately to its target mode.

Figure 8 shows the arbitration procedure for two controllers. More may be involved depending on how many controllers are connected to the bus. The moment there is a difference between the internal data level of the controller generating DATA1 and the actual level on the SDA line, the DATA1 output is switched off. This does not affect the data transfer initiated by the winning controller.

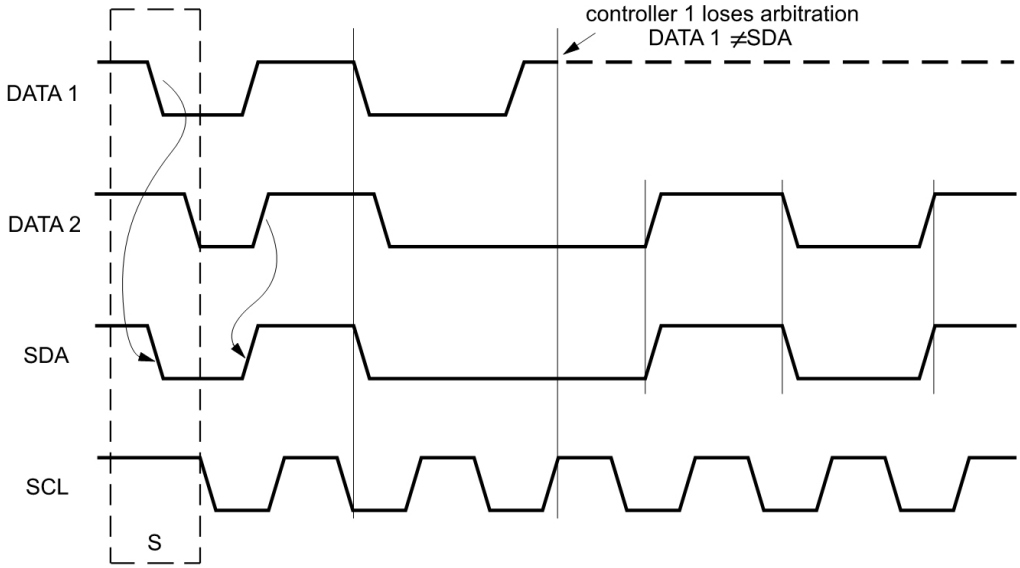


Figure 8. Arbitration procedure of two controllers

Since control of the I^{2}C-bus is decided solely on the address and data sent by competing controllers, there is no central controller, nor any order of priority on the bus.

There is an undefined condition if the arbitration procedure is still in progress at the moment when one controller sends a repeated START or a STOP condition while the other controller is still sending data. In other words, the following combinations result in an undefined condition:

- Controller 1 sends a repeated START condition and controller 2 sends a data bit.
- Controller 1 sends a STOP condition and controller 2 sends a data bit.
- Controller 1 sends a repeated START condition and controller 2 sends a STOP condition.

3.1.9 Clock stretching

Clock stretching pauses a transaction by holding the SCL line LOW. The transaction cannot continue until the line is released HIGH again. Clock stretching is optional and in fact, most target devices do not include an SCL driver so they are unable to stretch the clock.

On the byte level, a device may be able to receive bytes of data at a fast rate, but needs more time to store a received byte or prepare another byte to be transmitted. Targets can then hold the SCL line LOW after reception and acknowledgment of a byte to force the controller into a wait state until the target is ready for the next byte transfer in a type of handshake procedure (see Figure 7).

On the bit level, a device such as a microcontroller with or without limited hardware for the I^{2}C-bus, can slow down the bus clock by extending each clock LOW period. The speed of any controller is adapted to the internal operating rate of this device.

In Hs-mode, this handshake feature can only be used on byte level (see Section 5.3.2).

3.1.10 The target address and R/W bit

Data transfers follow the format shown in Figure 9. After the START condition (S), a target address is sent. This address is seven bits long followed by an eighth bit which is a data direction bit (R/$\overline{W}$) — a ‘zero’ indicates a transmission (WRITE), a ‘one’ indicates a request for data (READ) (refer to Figure 10). A data transfer is always terminated by

a STOP condition (P) generated by the controller. However, if a controller still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another target without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

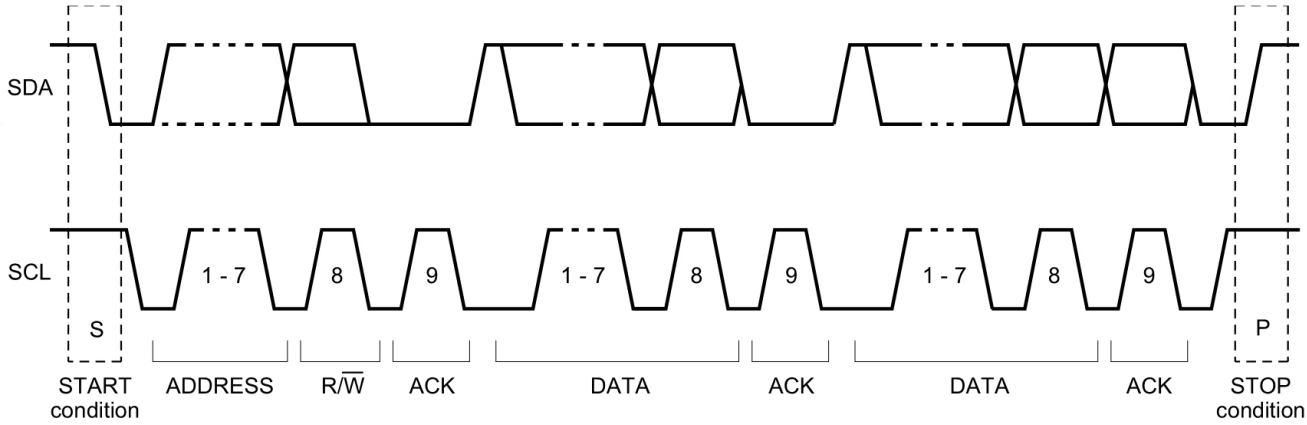


Figure 9. A complete data transfer

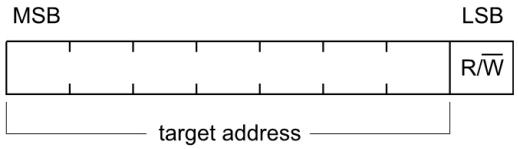


Figure 10. The first byte after the START procedure

Possible data transfer formats are:

- Controller-transmitter transmits to target-receiver. The transfer direction is not changed (see Figure 11). The target receiver acknowledges each byte.
- Controller reads target immediately after first byte (see Figure 12). At the moment of the first acknowledge, the controller-transmitter becomes a controller-receiver and the target-receiver becomes a target-transmitter. This first acknowledge is still generated by the target. The controller generates subsequent acknowledges. The STOP condition is generated by the controller, which sends a not-acknowledge ($\overline{A}$) just before the STOP condition.
- Combined format (see Figure 13). During a change of direction within a transfer, the START condition and the target address are both repeated, but with the R/$\overline{W}$ bit reversed. If a controller-receiver sends a repeated START condition, it sends a not-acknowledge ($\overline{A}$) just before the repeated START condition.

Notes:

1. Combined formats can be used, for example, to control a serial memory. The internal memory location must be written during the first data byte. After the START condition and target address is repeated, data can be transferred.
2. All decisions on auto-increment or decrement of previously accessed memory locations, etc., are taken by the designer of the device.
3. Each byte is followed by an acknowledgment bit as indicated by the A or $\overline{A}$ blocks in the sequence.
4. I^2C-bus compatible devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a target

address, even if these START conditions are not positioned according to the proper format.

5. A START condition immediately followed by a STOP condition (void message) is an illegal format. Many devices however are designed to operate properly under this condition.
6. Each device connected to the bus is addressable by a unique address. Normally a simple controller/target relationship exists, but it is possible to have multiple identical targets that can receive and respond simultaneously, for example in a group broadcast. This technique works best when using bus switching devices like the PCA9546A where all four channels are on and identical devices are configured at the same time, understanding that it is impossible to determine that each target acknowledges, and then turn on one channel at a time to read back each individual device's configuration to confirm the programming. Refer to individual component data sheets.

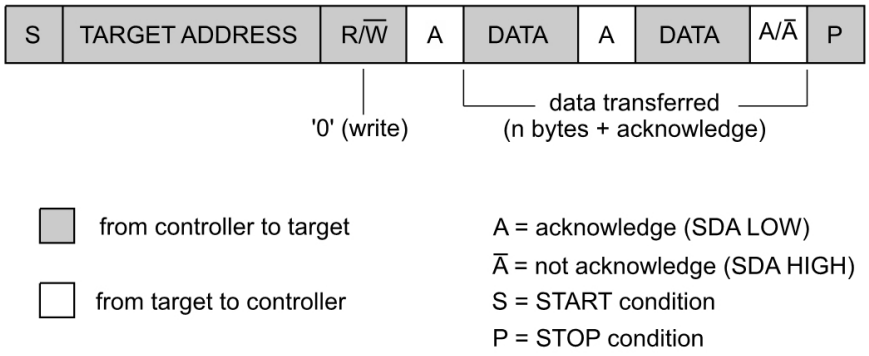


Figure 11. A controller-transmitter addressing a target receiver with a 7-bit address (the transfer direction is not changed)

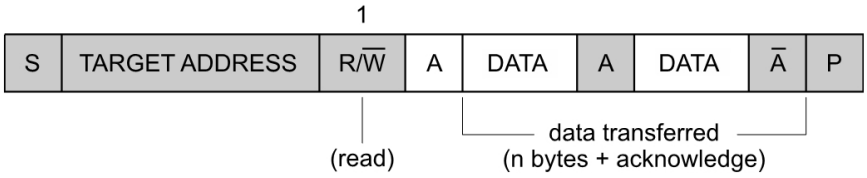


Figure 12. A controller reads a target immediately after the first byte

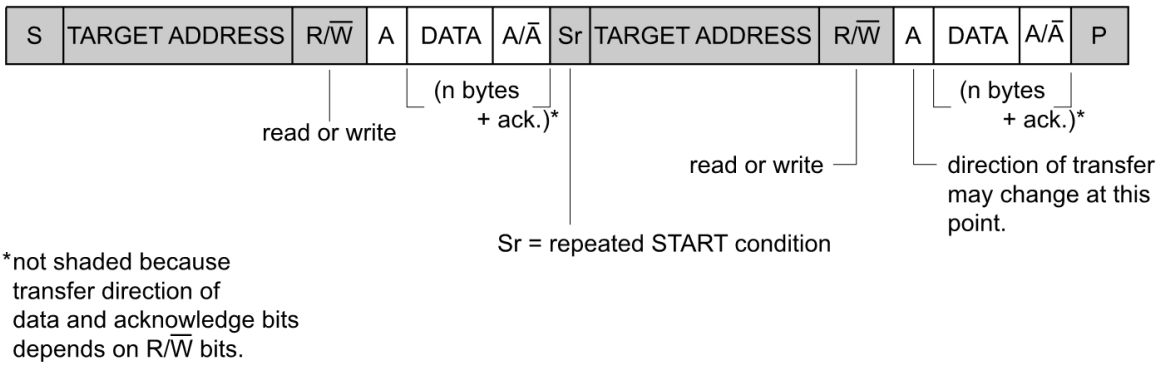


Figure 13. Combined format

3.1.11 10-bit addressing

10-bit addressing expands the number of possible addresses. Devices with 7-bit and 10-bit addresses can be connected to the same I^2C-bus, and both 7-bit and 10-bit addressing can be used in all bus speed modes. Currently, 10-bit addressing is not being widely used.

The 10-bit target address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr).

The first seven bits of the first byte are the combination 1111 0XX of which the last two bits (XX) are the two Most-Significant Bits (MSB) of the 10-bit address; the eighth bit of the first byte is the R/$\overline{W}$ bit that determines the direction of the message.

Although there are eight possible combinations of the reserved address bits 1111 XXX, only the four combinations 1111 0XX are used for 10-bit addressing. The remaining four combinations 1111 1XX are reserved for future I^2C-bus enhancements.

All combinations of read/write formats previously described for 7-bit addressing are possible with 10-bit addressing. Two are detailed here:

- Controller-transmitter transmits to target-receiver with a 10-bit target address. The transfer direction is not changed (see Figure 14). When a 10-bit address follows a START condition, each target compares the first seven bits of the first byte of the target address (1111 0XX) with its own address and tests if the eighth bit (R/$\overline{W}$ direction bit) is 0. It is possible that more than one device finds a match and generate an acknowledge (A1). All targets that found a match compare the eight bits of the second byte of the target address (XXXX XXXX) with their own addresses, but only one target finds a match and generates an acknowledge (A2). The matching target remains addressed by the controller until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different target address.
- Controller-receiver reads target-transmitter with a 10-bit target address. The transfer direction is changed after the second R/$\overline{W}$ bit (Figure 15). Up to and including acknowledge bit A2, the procedure is the same as that described for a controller-transmitter addressing a target-receiver. After the repeated START condition (Sr), a matching target remembers that it was addressed before. This target then checks if the first seven bits of the first byte of the target address following Sr are the same as they were after the START condition (S), and tests if the eighth (R/$\overline{W}$) bit is 1. If there is a match, the target considers that it has been addressed as a transmitter and generates acknowledge A3. The target-transmitter remains addressed until it receives a STOP condition (P) or until it receives another repeated START condition (Sr) followed by a different target address. After a repeated START condition (Sr), all the other target devices will also compare the first seven bits of the first byte of the target address (1111 0XX) with their own addresses and test the eighth (R/$\overline{W}$) bit. However, none of them will be addressed because R/$\overline{W}$ = 1 (for 10-bit devices), or the 1111 0XX target address (for 7-bit devices) does not match.

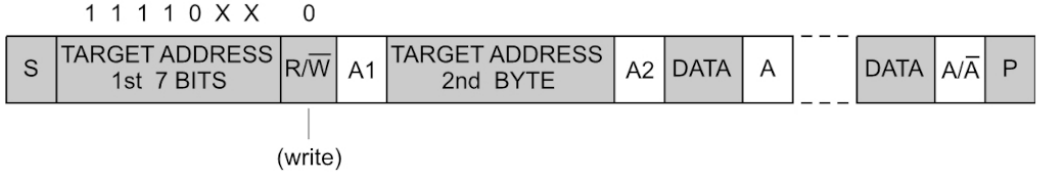


Figure 14. A controller-transmitter addresses a target-receiver with a 10-bit address

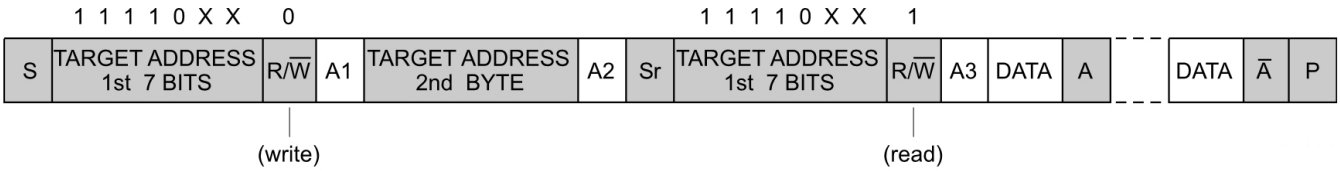


Figure 15. A controller-receiver addresses a target-transmitter with a 10-bit address

Target devices with 10-bit addressing react to a ‘general call’ in the same way as target devices with 7-bit addressing. Hardware controllers can transmit their 10-bit address after a ‘general call’. In this case, the ‘general call’ address byte is followed by two successive bytes containing the 10-bit address of the controller-transmitter. The format is as shown in Figure 15 where the first DATA byte contains the eight least-significant bits of the controller address.

The START byte 0000 0001 (01h) can precede the 10-bit addressing in the same way as for 7-bit addressing (see Section 3.1.15).

3.1.12 Reserved addresses

Two groups of eight addresses (0000 XXX and 1111 XXX) are reserved for the purposes shown in Table 4.

Table 4. Reserved addresses

X = don’t care; 1 = HIGH; 0 = LOW.

Target address	R/$\overline{W}$ bit	Description
0000 000	0	general call address[1]
0000 000	1	START byte[2]
0000 001	X	CBUS address[3]
0000 010	X	reserved for different bus format[4]
0000 011	X	reserved for future purposes
0000 1XX	X	Hs-mode controller code
1111 1XX	1	device ID
1111 0XX	X	10-bit target addressing

[1] The general call address is used for several functions including software reset.
[2] No device is allowed to acknowledge at the reception of the START byte.
[3] The CBUS address has been reserved to enable the inter-mixing of CBUS compatible and I^2C-bus compatible devices in the same system. I^2C-bus compatible devices are not allowed to respond on reception of this address.
[4] The address reserved for a different bus format is included to enable I^2C and other protocols to be mixed. Only I^2C-bus compatible devices that can work with such formats and protocols are allowed to respond to this address.

Assignment of addresses within a local system is up to the system architect who must take into account the devices being used on the bus and any future interaction with other conventional I^2C-buses. For example, a device with seven user-assignable address pins allows all 128 addresses to be assigned. If it is known that the reserved address is never going to be used for its intended purpose, a reserved address can be used for a target address.

3.1.13 General call address

The general call address is for addressing every device connected to the I^2C-bus at the same time. However, if a device does not need any of the data supplied within the

general call structure, it can ignore this address by not issuing an acknowledgment. If a device does require data from a general call address, it acknowledges this address and behave as a target-receiver. The controller does not actually know how many devices acknowledged if one or more devices respond. The second and following bytes are acknowledged by every target-receiver capable of handling this data. A target who cannot process one of these bytes must ignore it by not-acknowledging. Again, if one or more targets acknowledge, the not-acknowledge will not be seen by the controller. The meaning of the general call address is always specified in the second byte (see Figure 16).

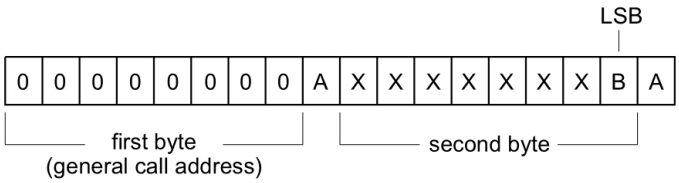


Figure 16. General call address format

There are two cases to consider:

- When the least significant bit B is a ‘zero’.
- When the least significant bit B is a ‘one’.

When bit B is a ‘zero’, the second byte has the following definition:

- **0000 0110 (06h): Reset and write programmable part of target address by hardware.** On receiving this 2-byte sequence, all devices designed to respond to the general call address reset and take in the programmable part of their address. Precautions must be taken to ensure that a device is not pulling down the SDA or SCL line after applying the supply voltage, since these low levels would block the bus.
- **0000 0100 (04h): Write programmable part of target address by hardware.** Behaves as above, but the device does not reset.
- **0000 0000 (00h): This code is not allowed to be used as the second byte.**

Sequences of programming procedure are published in the appropriate device data sheets. The remaining codes have not been fixed and devices must ignore them.

When bit B is a ‘one’, the 2-byte sequence is a ‘hardware general call’. This means that the sequence is transmitted by a hardware controller device, such as a keyboard scanner, which can be programmed to transmit a desired target address. Since a hardware controller does not know in advance to which device the message has to be transferred, it can only generate this hardware general call and its own address — identifying itself to the system (see Figure 17).

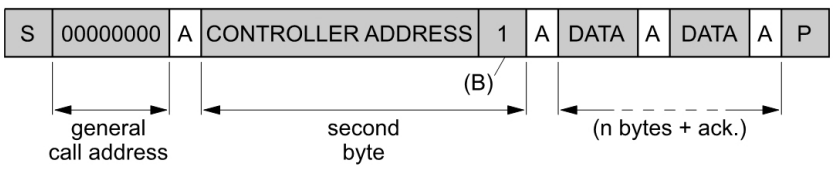
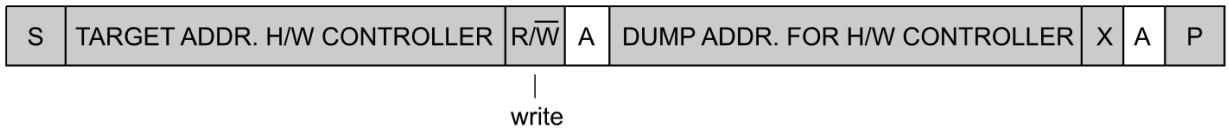


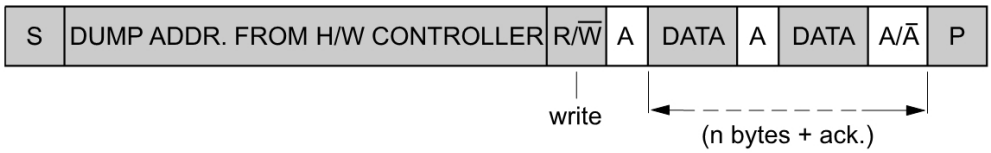
Figure 17. Data transfer from a hardware controller-transmitter

The seven bits remaining in the second byte contain the address of the hardware controller. This address is recognized by an intelligent device (for example, a microcontroller) connected to the bus which then accepts the information from the hardware controller. If the hardware controller can also act as a target, the target address is identical to the controller address.

In some systems, an alternative could be that the hardware controller transmitter is set in the target-receiver mode after the system reset. In this way, a system configuring controller can tell the hardware controller-transmitter (which is now in target-receiver mode) to which address data must be sent (see Figure 18). After this programming procedure, the hardware controller remains in the controller-transmitter mode.



a. Configuring controller sends dump address to hardware controller



b. Hardware controller dumps data to selected target

Figure 18. Data transfer by a hardware-transmitter capable of dumping data directly to target devices

3.1.14 Software reset

Following a General Call, (0000 0000), sending 0000 0110 (06h) as the second byte causes a software reset. This feature is optional and not all devices respond to this command. On receiving this 2-byte sequence, all devices designed to respond to the general call address reset and take in the programmable part of their address. Precautions must be taken to ensure that a device is not pulling down the SDA or SCL line after applying the supply voltage, since these low levels would block the bus.

3.1.15 START byte

Microcontrollers can be connected to the I^2C-bus in two ways. A microcontroller with an on-chip hardware I^2C-bus interface can be programmed to be only interrupted by requests from the bus. When the device does not have such an interface, it must constantly monitor the bus via software. Obviously, the more times the microcontroller monitors, or polls the bus, the less time it can spend carrying out its intended function.

There is therefore a speed difference between fast hardware devices and a relatively slow microcontroller which relies on software polling.

In this case, data transfer can be preceded by a start procedure which is much longer than normal (see Figure 19). The start procedure consists of:

- A START condition (S)
- A START byte (0000 0001)
- An acknowledge clock pulse (ACK)
- A repeated START condition (Sr).

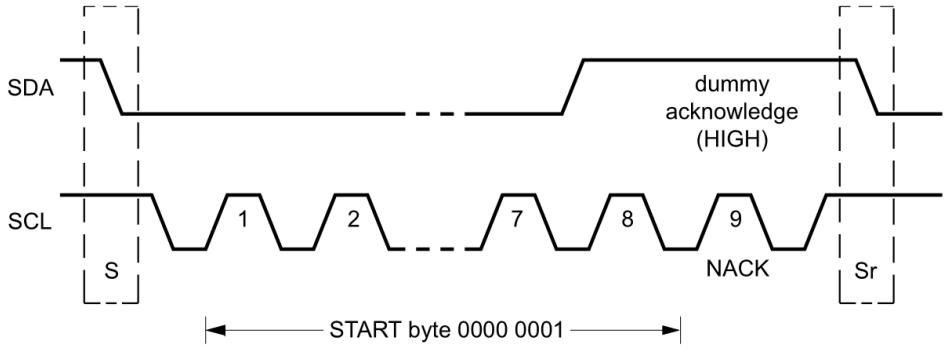


Figure 19. START byte procedure

After the START condition S has been transmitted by a controller which requires bus access, the START byte (0000 0001) is transmitted. Another microcontroller can therefore sample the SDA line at a low sampling rate until one of the seven zeros in the START byte is detected. After detection of this LOW level on the SDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition Sr which is then used for synchronization.

A hardware receiver resets upon receipt of the repeated START condition Sr and therefore ignores the START byte.

An acknowledge-related clock pulse is generated after the START byte. This is present only to conform with the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

3.1.16 Bus clear

In the unlikely event where the clock (SCL) is stuck LOW, the preferential procedure is to reset the bus using the HW reset signal if your I^2C devices have HW reset inputs. If the I^2C devices do not have HW reset inputs, cycle power to the devices to activate the mandatory internal Power-On Reset (POR) circuit.

If the data line (SDA) is stuck LOW, the controller should send nine clock pulses. The device that held the bus LOW should release it sometime within those nine clocks. If not, then use the HW reset or cycle power to clear the bus.

3.1.17 Device ID

The Device ID field (see Figure 20) is an optional 3-byte read-only (24 bits) word giving the following information:

- Twelve bits with the manufacturer name, unique per manufacturer (for example, NXP)
- Nine bits with the part identification, assigned by manufacturer (for example, PCA9698)
- Three bits with the die revision, assigned by manufacturer (for example, RevX)

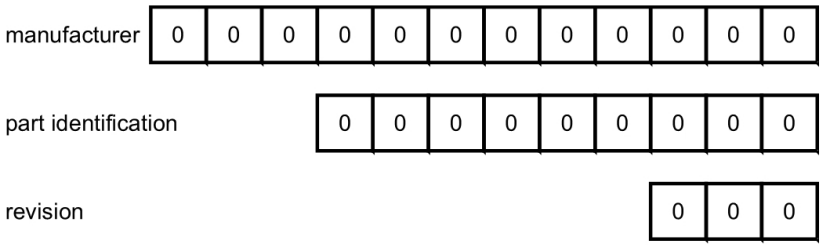


Figure 20. Device ID field

The Device ID is read-only, hard-wired in the device and can be accessed as follows:

1. START condition
2. The controller sends the Reserved Device ID I^2C-bus address followed by the R/$\overline{W}$ bit set to '0' (write): '1111 1000'.
3. The controller sends the I^2C-bus target address of the target device it must identify. The LSB is a 'Don't care' value. Only one device must acknowledge this byte (the one that has the I^2C-bus target address).
4. The controller sends a Re-START condition.
   Remark: A STOP condition followed by a START condition resets the target state machine and the Device ID Read cannot be performed. Also, a STOP condition or a Re-START condition followed by an access to another target device resets the target state machine and the Device ID Read cannot be performed.
5. The controller sends the Reserved Device ID I^2C-bus address followed by the R/$\overline{W}$ bit set to '1' (read): '1111 1001'.
6. The Device ID Read can be done, starting with the 12 manufacturer bits (first byte + four MSBs of the second byte), followed by the nine part identification bits (four LSBs of the second byte + five MSBs of the third byte), and then the three die revision bits (three LSBs of the third byte).
7. The controller ends the reading sequence by NACKing the last byte, thus resetting the target device state machine and allowing the controller to send the STOP condition.
   Remark: The reading of the Device ID can be stopped anytime by sending a NACK.

If the controller continues to ACK the bytes after the third byte, the target rolls back to the first byte and keeps sending the Device ID sequence until a NACK has been detected.

Table 5. Assigned manufacturer IDs

Manufacturer bits												Company
11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	0	0	0	0	0	NXP Semiconductors
0	0	0	0	0	0	0	0	0	0	0	1	NXP Semiconductors (reserved)
0	0	0	0	0	0	0	0	0	0	1	0	NXP Semiconductors (reserved)
0	0	0	0	0	0	0	0	0	0	1	1	NXP Semiconductors (reserved)
0	0	0	0	0	0	0	0	0	1	0	0	Ramtron International
0	0	0	0	0	0	0	0	0	1	0	1	Analog Devices
0	0	0	0	0	0	0	0	0	1	1	0	STMicroelectronics
0	0	0	0	0	0	0	0	0	1	1	1	ON Semiconductor
0	0	0	0	0	0	0	0	1	0	0	0	Sprintek Corporation
0	0	0	0	0	0	0	0	1	0	0	1	ESPROS Photonics AG
0	0	0	0	0	0	0	0	1	0	1	0	Fujitsu Semiconductor
0	0	0	0	0	0	0	0	1	0	1	1	Flir
0	0	0	0	0	0	0	0	1	1	0	0	O_2Micro
0	0	0	0	0	0	0	0	1	1	0	1	Atmel
0	0	0	0	0	0	0	0	1	1	1	0	DIODES Incorporated
0	0	0	0	0	0	0	0	1	1	1	1	Pericom
0	0	0	0	0	0	0	1	0	0	0	0	Marvell Semiconductor Inc

Table 5. Assigned manufacturer IDs...*continued*

Manufacturer bits												Company
11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	1	0	0	0	1	ForteMedia
0	0	0	0	0	0	0	1	0	0	1	0	Snaju LLC
0	0	0	0	0	0	0	1	0	0	1	1	Intel
0	0	0	0	0	0	0	1	0	1	0	0	Pericom
0	0	0	0	0	0	0	1	0	1	0	1	Arctic Sand Technologies
0	0	0	0	0	0	0	1	0	1	1	0	Micron Technology
0	0	0	0	0	0	0	1	0	1	1	1	Semtech Corporation
0	0	0	0	0	0	0	1	1	0	0	0	IDT
0	0	0	0	0	0	0	1	1	0	0	1	TT Electronics
0	0	0	0	0	0	0	1	1	0	1	0	Alien Technology
0	0	0	0	0	0	0	1	1	0	1	1	LAPIS semiconductor
0	0	0	0	0	0	0	1	1	1	0	0	Qorvo
0	0	0	0	0	0	0	1	1	1	0	1	Wuxi Chipown Micro-electronics limited (Chipown)
0	0	0	0	0	0	0	1	1	1	1	0	KOA CORPORATION
0	0	0	0	0	0	0	1	1	1	1	1	Prevo Technologies, Inc.

3.2 Ultra Fast-mode I^2C-bus protocol

The UFm I^2C-bus is a 2-wire push-pull serial bus that operates from DC to 5 MHz transmitting data in one direction. It is most useful for speeds greater than 1 MHz to drive LED controllers and other devices that do not need feedback. The UFm I^2C-bus protocol is based on the standard I^2C-bus protocol that consists of a START, target address, command bit, ninth clock, and a STOP bit. The command bit is a ‘write’ only, and the data bit on the ninth clock is driven HIGH, ignoring the ACK cycle due to the unidirectional nature of the bus. The 2-wire push-pull driver consists of a UFm serial clock (USCL) and serial data (USDA).

Target devices contain a unique address (whether it is a microcontroller, LCD driver, LED controller, GPO) and operate only as receivers. An LED driver may be only a receiver and can be supported by UFm, whereas a memory can both receive and transmit data and is not supported by UFm.

Since UFm I^2C-bus uses push-pull drivers, it does not have the multi-controller capability of the wired-AND open-drain Sm, Fm, and Fm+ I^2C-buses. In UFm, a controller is the only device that initiates a data transfer on the bus and generates the clock signals to permit that transfer. All other devices addressed are considered targets.

Table 6. Definition of UFm I^2C-bus terminology

Term	Description
Transmitter	the device that sends data to the bus
Receiver	the device that receives data from the bus

Table 6. Definition of UFm I²C-bus terminology *...continued*

Term	Description
Controller	the device that initiates a transfer, generates clock signals and terminates a transfer
Target	the device addressed by a controller

Let us consider the case of a data transfer between a controller and multiple targets connected to the UFm I²C-bus (see Figure 21).

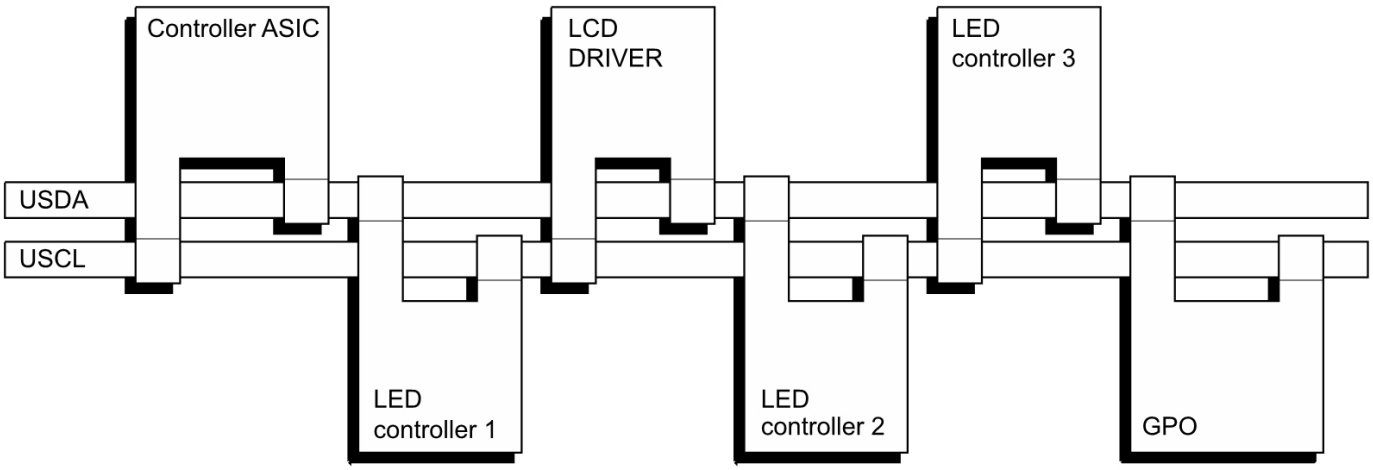


Figure 21. Example of UFm I²C-bus configuration

This highlights the controller/transmitter-target/receiver relationship found on the UFm I²C-bus. Note that these relationships are permanent, as data transfer is only permitted in one direction. The transfer of data would proceed as follows:

Suppose that the controller ASIC wants to send information to the LED controller 2:

- ASIC A (controller-transmitter), addresses LED controller 2 (target-receiver) by sending the address on the USDA and generating the clock on USCL.
- ASIC A (controller-transmitter), sends data to LED controller 2 (target-receiver) on the USDA and generates the clock on USCL.
- ASIC A terminates the transfer.

The possibility of connecting more than one UFm controller to the UFm I²C-bus is not allowed due to bus contention on the push-pull outputs. If an additional controller is required in the system, it must be fully isolated from the other controller (that is, with a true ‘one hot’ MUX) as only one controller is allowed on the bus at a time.

Generation of clock signals on the UFm I²C-bus is always the responsibility of the controller device, that is, the controller generates the clock signals when transferring data on the bus. Bus clock signals from a controller cannot be altered by a target device with clock stretching and the process of arbitration and clock synchronization does not exist within the UFm I²C-bus.

Table 7 summarizes the use of mandatory and optional portions of the UFm I²C-bus specification.

Table 7. Applicability of I²C-bus features to UFm

M = mandatory; O = optional; n/p = not possible

Feature	Configuration
	Single controller
START condition	M
STOP condition	M
Acknowledge	n/p
Synchronization	n/p
Arbitration	n/p
Clock stretching	n/p
7-bit target address	M
10-bit target address	O
General Call address	O
Software Reset	O
START byte	O
Device ID	n/p

3.2.1 USDA and USCL signals

Both USDA and USCL are unidirectional lines, with push-pull outputs. When the bus is free, both lines are pulled HIGH by the upper transistor of the output stage. Data on the I²C-bus can be transferred at rates of up to 5000 kbit/s in the Ultra Fast-mode. The number of interfaces connected to the bus is limited by the bus loading, reflections from cable ends, connectors, and stubs.

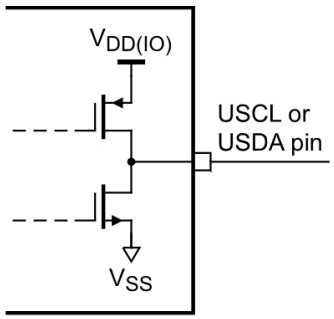


Figure 22. Simplified schematic of USCL, USDA outputs

3.2.2 USDA and USCL logic levels

Due to the variety of different technology devices (CMOS, NMOS, bipolar) that can be connected to the I²C-bus, the levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of V_{DD}. Input reference levels are set as 30 % and 70 % of V_{DD}; V_{IL} is $0.3V_{DD}$ and V_{IH} is $0.7V_{DD}$. See Figure 40, timing diagram. See Section 6 for electrical specifications.

3.2.3 Data validity

The data on the USDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the USCL line is LOW (see Figure 23). One clock pulse is generated for each data bit transferred.

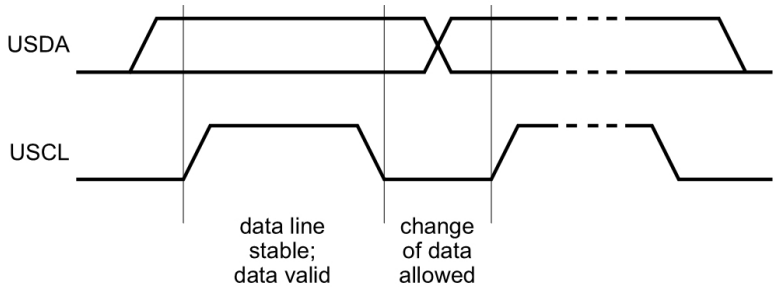


Figure 23. Bit transfer on the UFm I²C-bus

3.2.4 START and STOP conditions

Both data and clock lines remain HIGH when the bus is not busy. All transactions begin with a START (S) and can be terminated by a STOP (P) (see Figure 24). A HIGH to LOW transition on the USDA line while USCL is HIGH defines a START condition. A LOW to HIGH transition on the USDA line while USCL is HIGH defines a STOP condition.

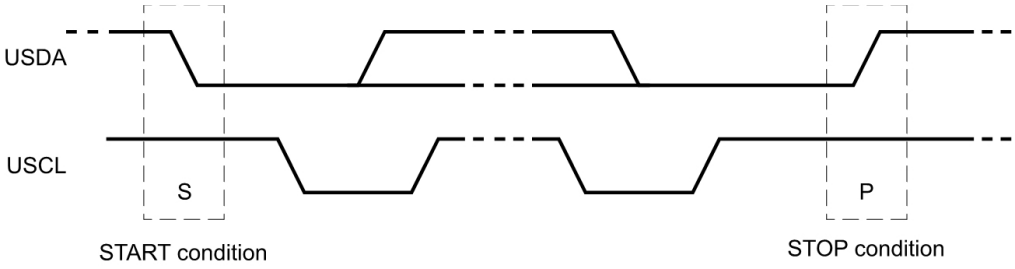


Figure 24. Definition of START and STOP conditions for UFm I²C-bus

START and STOP conditions are always generated by the controller. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition. This bus free situation is specified in Section 6. The bus stays busy if a repeated START (Sr) is generated instead of a STOP condition. In this respect, the START (S) and repeated START (Sr) conditions are functionally identical. For the remainder of this document, therefore, the S symbol is used as a generic term to represent both the START and repeated START conditions, unless Sr is particularly relevant.

Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware. However, microcontrollers with no such interface have to sample the USDA line at least twice per clock period to sense the transition.

3.2.5 Byte format

Every byte put on the USDA line must be eight bits long. The number of bytes that can be transmitted per transfer is unrestricted. The controller drives the USDA HIGH after each byte during the Acknowledge cycle. Data is transferred with the Most Significant Bit (MSB) first (see Figure 25). A target is not allowed to hold the clock LOW if it cannot

receive another complete byte of data or while it is performing some other function, for example servicing an internal interrupt.

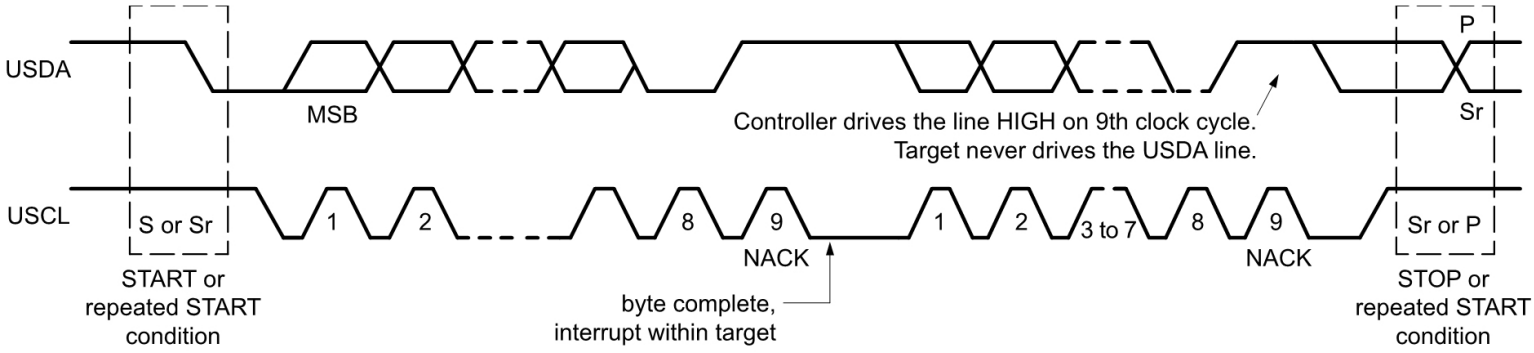


Figure 25. Data transfer on the UFm I^2C-bus

3.2.6 Acknowledge (ACK) and Not Acknowledge (NACK)

Since the targets are not able to respond the ninth clock cycle, the ACK and NACK are not required. However, the clock cycle is preserved in the UFm to be compatible with the I^2C-bus protocol. The ACK and NACK are also referred to as the ninth clock cycle. The controller generates all clock pulses, including the ninth clock pulse. The ninth data bit is always driven HIGH (‘1’). Target devices are not allowed to drive the SDA line at any time.

3.2.7 The target address and R/W bit

Data transfers follow the format shown in Figure 26. After the START condition (S), a target address is sent. This address is seven bits long followed by an eighth bit which is a data direction bit ($\overline{W}$) — a ‘zero’ indicates a transmission (WRITE); a ‘one’ indicates a request for data (READ) and is not supported by UFm (except for the START byte, Section 3.2.12) since the communication is unidirectional (refer to Figure 27). A data transfer is always terminated by a STOP condition (P) generated by the controller. However, if a controller still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another target without first generating a STOP condition.

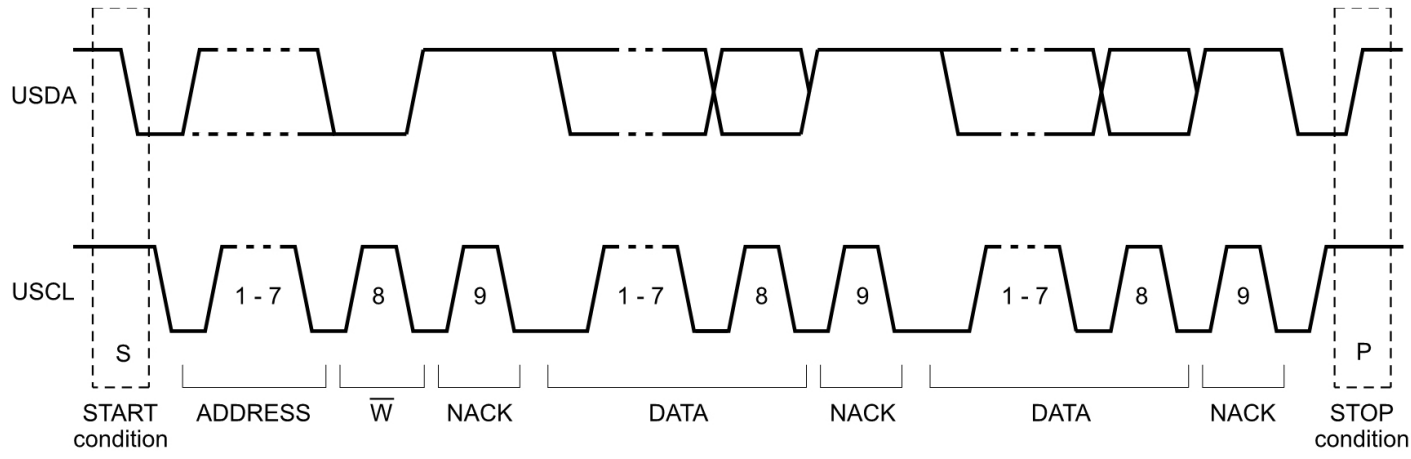


Figure 26. A complete UFm data transfer

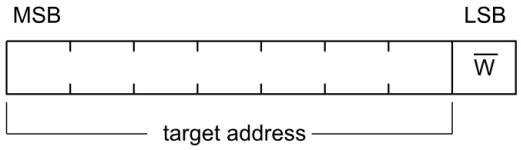


Figure 27. The first byte after the START procedure

The UFm data transfer format is:

- Controller-transmitter transmits to target-receiver. The transfer direction is not changed (see Figure 28). The controller never acknowledges because it never receives any data but generates the '1' on the ninth bit for the target to conform to the I^2C-bus protocol.

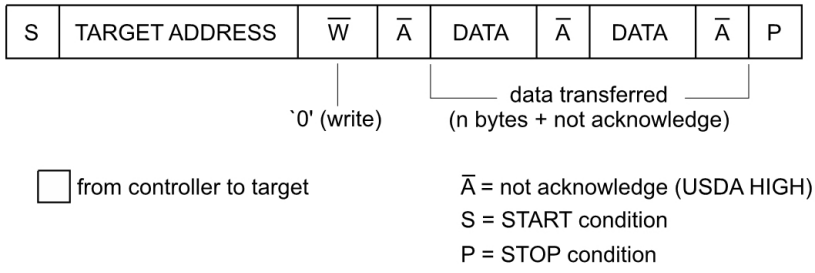


Figure 28. A controller-transmitter addressing a target receiver with a 7-bit address

Notes:

1. Individual transaction or repeated START formats addressing multiple targets in one transaction can be used. After the START condition and target address is repeated, data can be transferred.
2. All decisions on auto-increment or decrement of previously accessed memory locations, etc., are taken by the designer of the device.
3. Each byte is followed by a Not-Acknowledgment bit as indicated by the $\overline{A}$ blocks in the sequence.
4. I^2C-bus compatible devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a target address, even if these START conditions are not positioned according to the proper format.
5. A START condition immediately followed by a STOP condition (void message) is an illegal format. Many devices however are designed to operate properly under this condition.
6. Each device connected to the bus is addressable by a unique address. A simple controller/target relationship exists, but it is possible to have multiple identical targets that can receive and respond simultaneously, for example, in a group broadcast where all identical devices are configured at the same time, understanding that it is impossible to determine that each target is responsive. Refer to individual component data sheets.

3.2.8 10-bit addressing

10-bit addressing expands the number of possible addresses. Devices with 7-bit and 10-bit addresses can be connected to the same I^2C-bus, and both 7-bit and 10-bit addressing can be used in all bus speed modes.

The 10-bit target address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr). The first seven bits of the first byte are the

combination 1111 0XX of which the last two bits (XX) are the two Most Significant Bits (MSBs) of the 10-bit address; the eighth bit of the first byte is the R/$\overline{W}$ bit that determines the direction of the message.

Although there are eight possible combinations of the reserved address bits 1111 XXX, only the four combinations 1111 0XX are used for 10-bit addressing. The remaining four combinations 1111 1XX are reserved for future I^2C-bus enhancements.

Only the write format previously described for 7-bit addressing is possible with 10-bit addressing. Detailed here:

- Controller-transmitter transmits to target-receiver with a 10-bit target address. The transfer direction is not changed (see Figure 29). When a 10-bit address follows a START condition, each target compares the first seven bits of the first byte of the target address (1111 0XX) with its own address and tests if the eighth bit (R/$\overline{W}$ direction bit) is 0 ($\overline{W}$). All targets that found a match compare the eight bits of the second byte of the target address (XXXX XXXX) with their own addresses, but only one target finds a match. The matching target remains addressed by the controller until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different target address.

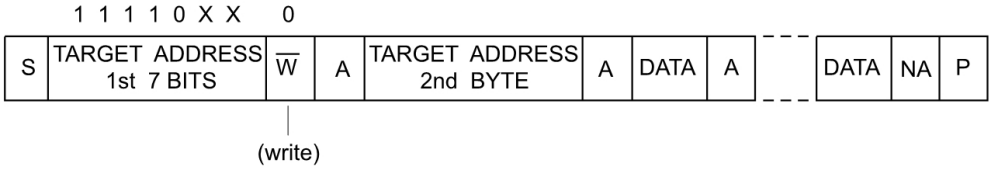


Figure 29. A controller-transmitter addresses a target-receiver with a 10-bit address

The START byte 0000 0001 (01h) can precede the 10-bit addressing in the same way as for 7-bit addressing (see Section 3.2.12).

3.2.9 Reserved addresses in UFm

The UFm I^2C-bus has a different physical layer than the other I^2C-bus modes. Therefore the available target address range is different. Two groups of eight addresses (0000 XXX and 1111 XXX) are reserved for the purposes shown in Table 8.

Table 8. Reserved addresses

X = don't care; 1 = HIGH; 0 = LOW.

Target address	R/$\overline{W}$ bit	Description
0000 000	0	general call address[1]
0000 000	1	START byte[2]
0000 001	X	reserved for future purposes
0000 010	X	reserved for future purposes
0000 011	X	reserved for future purposes
0000 1XX	X	reserved for future purposes
1111 1XX	X	reserved for future purposes
1111 0XX	X	10-bit target addressing

[1] The general call address is used for several functions including software reset.
[2] No UFm device is allowed to acknowledge at the reception of the START byte.

Assignment of addresses within a local system is up to the system architect who must take into account the devices being used on the bus and any future interaction with reserved addresses. For example, a device with seven user-assignable address pins allows all 128 addresses to be assigned. If it is known that the reserved address is never going to be used for its intended purpose, then a reserved address can be used for a target address.

3.2.10 General call address

The general call address is for addressing every device connected to the I^2C-bus at the same time. However, if a device does not need any of the data supplied within the general call structure, it can ignore this address. If a device does require data from a general call address, it behaves as a target-receiver. The controller does not actually know how many devices are responsive to the general call. The second and following bytes are received by every target-receiver capable of handling this data. A target that cannot process one of these bytes must ignore it. The meaning of the general call address is always specified in the second byte (see Figure 30).

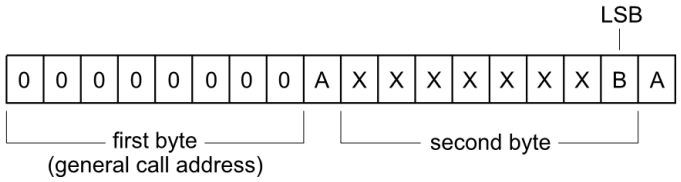


Figure 30. General call address format

There are two cases to consider:

- When the least significant bit B is a ‘zero’
- When the least significant bit B is a ‘one’

When bit B is a ‘zero’, the second byte has the following definition:

0000 0110 (06h)

Reset and write programmable part of target address by hardware. On receiving this 2-byte sequence, all devices designed to respond to the general call address reset and take in the programmable part of their address.

0000 0100 (04h)

Write programmable part of target address by hardware. Behaves as above, but the device does not reset.

0000 0000 (00h)

This code is not allowed to be used as the second byte. Sequences of programming procedure are published in the appropriate device data sheets. The remaining codes have not been fixed and devices must ignore them.

When bit B is a ‘one’, the 2-byte sequence is ignored.

3.2.11 Software reset

Following a General Call, (0000 0000), sending 0000 0110 (06h) as the second byte causes a software reset. This feature is optional and not all devices respond to this command. On receiving this 2-byte sequence, all devices designed to respond to the general call address reset and take in the programmable part of their address.

3.2.12 START byte

Microcontrollers can be connected to the I^2C-bus in two ways. A microcontroller with an on-chip hardware I^2C-bus interface can be programmed to be only interrupted by requests from the bus. When the device does not have such an interface, it must constantly monitor the bus via software. Obviously, the more times the microcontroller monitors, or polls the bus, the less time it can spend carrying out its intended function.

There is therefore a speed difference between fast hardware devices and a relatively slow microcontroller which relies on software polling.

In this case, data transfer can be preceded by a start procedure which is much longer than normal (see Figure 31). The start procedure consists of:

- A START condition (S)
- A START byte (0000 0001)
- A Not Acknowledge clock pulse (NACK)
- A repeated START condition (Sr)

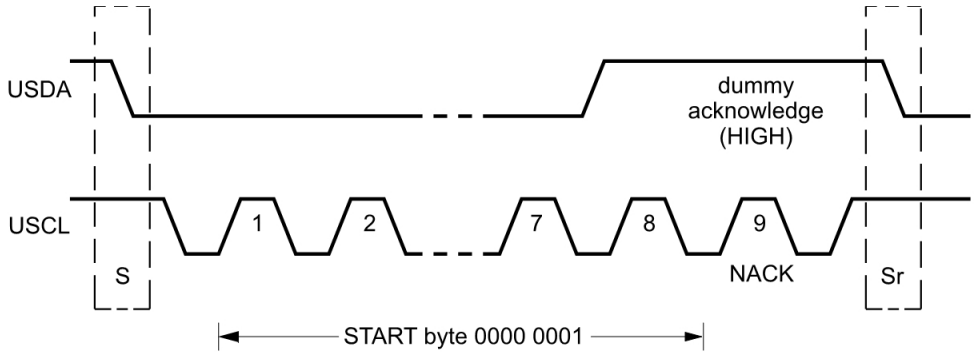


Figure 31. START byte procedure

After the START condition S has been transmitted by a controller which requires bus access, the START byte (0000 0001) is transmitted. Another microcontroller can therefore sample the USDA line at a low sampling rate until one of the seven zeros in the START byte is detected. After detection of this LOW level on the USDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition Sr, which is then used for synchronization. A hardware receiver resets upon receipt of the repeated START condition Sr and therefore ignores the START byte. An acknowledge-related clock pulse is generated after the START byte. This is present only to conform with the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

3.2.13 Unresponsive target reset

In the unlikely event where the target becomes unresponsive (for example, determined through external feedback, not through UFm I^2C-bus), the preferential procedure is to reset the target by using the software reset command or the hardware reset signal. If the targets do not support these features, then cycle power to the devices to activate the mandatory internal Power-On Reset (POR) circuit.

3.2.14 Device ID

The Device ID field is not supported in UFm.

4 Other uses of the I²C-bus communications protocol

The I²C-bus is used as the communications protocol for several system architectures. These architectures have added command sets and application-specific extensions in addition to the base I²C specification.

In general, simple I²C-bus devices such as I/O extenders could be used in any one of these architectures since the protocol and physical interfaces are the same.

4.1 CBUS compatibility

CBUS receivers can be connected to the Standard-mode I²C-bus. However, a third bus line called DLEN must then be connected and the acknowledge bit omitted. Normally, I²C transmissions are sequences of 8-bit bytes; CBUS compatible devices have different formats.

In a mixed bus structure, I²C-bus devices must not respond to the CBUS message. For this reason, a special CBUS address (0000 001X) to which no I²C-bus compatible device responds has been reserved. After transmission of the CBUS address, the DLEN line can be made active and a CBUS-format transmission sent. After the STOP condition, all devices are again ready to accept data.

Controller-transmitters can send CBUS formats after sending the CBUS address. The transmission is ended by a STOP condition, recognized by all devices.

Remark: If the CBUS configuration is known, and expansion with CBUS compatible devices is not foreseen, the designer is allowed to adapt the hold time to the specific requirements of the device(s) used.

4.2 SMBus - System Management Bus

The SMBus uses I²C hardware and I²C hardware addressing, but adds second-level software for building special systems. In particular, its specifications include an Address Resolution Protocol that can make dynamic address allocations.

Dynamic reconfiguration of the hardware and software allow bus devices to be ‘hot-plugged’ and used immediately, without restarting the system. The devices are recognized automatically and assigned unique addresses. This advantage results in a plug-and-play user interface. In both those protocols, there is a very useful distinction made between a System Host and all the other devices in the system that can have the names and functions of controllers or targets.

SMBus is used today as a system management bus in most PCs. Developed by Intel and others in 1995, it modified some I²C electrical and software characteristics for better compatibility with the quickly decreasing power supply budget of portable equipment. SMBus also has a ‘High Power’ version 2.0 that includes a 4 mA sink current that cannot be driven by I²C chips unless the pull-up resistor is sized to I²C-bus levels.

4.2.1 I²C/SMBus compliancy

SMBus and I²C protocols are basically the same: A SMBus controller is able to control I²C devices and vice versa at the protocol level. The SMBus clock is defined from 10 kHz to 100 kHz while I²C can be 0 Hz to 100 kHz, 0 Hz to 400 kHz, 0 Hz to 1 MHz and 0 Hz to 3.4 MHz, depending on the mode. This means that an I²C-bus running at less than 10 kHz is not SMBus compliant since the SMBus devices may time-out.

Logic levels are slightly different also: TTL for SMBus: LOW = 0.8 V and HIGH = 2.1 V, versus the 30 %/70 % V_{DD} CMOS level for I^2C. This is not a problem if V_{DD} > 3.0 V. If the I^2C device is below 3.0 V, then there could be a problem if the logic HIGH/LOW levels are not properly recognized.

4.2.2 Time-out feature

SMBus has a time-out feature which resets devices if a communication takes too long. This explains the minimum clock frequency of 10 kHz to prevent locking up the bus. I^2C can be a 'DC' bus, meaning that a target device stretches the controller clock when performing some routine while the controller is accessing it. This notifies the controller that the target is busy but does not want to lose the communication. The target device will allow continuation after its task is complete. There is no limit in the I^2C-bus protocol as to how long this delay can be, whereas for a SMBus system, it would be limited to 35 ms.

SMBus protocol just assumes that if something takes too long, then it means that there is a problem on the bus and that all devices must reset in order to clear this mode. Target devices are not then allowed to hold the clock LOW too long.

4.2.3 Differences between SMBus 1.0 and SMBus 2.0

The SMBus specification defines two classes of electrical characteristics: low power and high power. The first class, originally defined in the SMBus 1.0 and 1.1 specifications, was designed primarily with Smart Batteries in mind, but could be used with other low-power devices.

The 2.0 version introduces an alternative higher power set of electrical characteristics. This class is appropriate for use when higher drive capability is required, for example with SMBus devices on PCI add-in cards and for connecting such cards across the PCI connector between each other and to SMBus devices on the system board.

Devices may be powered by the bus V_{DD} or by another power source, V_{Bus} (as with, for example, Smart Batteries), and will inter-operate as long as they adhere to the SMBus electrical specifications for their class.

NXP devices have a higher power set of electrical characteristics than SMBus 1.0. The main difference is the current sink capability with V_{OL} = 0.4 V.

- SMBus low power = 350 µA
- SMBus high power = 4 mA
- I^2C-bus = 3 mA

SMBus 'high power' devices and I^2C-bus devices will work together if the pull-up resistor is sized for 3 mA.

For more information, refer to: http://www.smbus.org/.

4.3 PMBus - Power Management Bus

PMBus is a standard way to communicate between power converters and a system host over the SMBus to provide more intelligent control of the power converters. The PMBus specification defines a standard set of device commands so that devices from multiple sources function identically. PMBus devices use the SMBus Version 1.1 plus extensions for transport.

For more information, refer to: https://pmbus.org/.

4.4 Intelligent Platform Management Interface (IPMI)

Intelligent Platform Management Interface (IPMI) defines a standardized, abstracted, message-based interface for intelligent platform management hardware. IPMI also defines standardized records for describing platform management devices and their characteristics. IPMI increases reliability of systems by monitoring parameters such as temperatures, voltages, fans and chassis intrusion.

IPMI provides general system management functions such as automatic alerting, automatic system shutdown and restart, remote restart and power control. The standardized interface to intelligent platform management hardware aids in prediction and early monitoring of hardware failures as well as diagnosis of hardware problems.

This standardized bus and protocol for extending management control, monitoring, and event delivery within the chassis:

- I^2C based
- Multi-controller
- Simple Request/Response Protocol
- Uses IPMI Command sets
- Supports non-IPMI devices
- Physically I^2C but write-only (controller capable devices); hot swap not required
- Enables the Baseboard Management Controller (BMC) to accept IPMI request messages from other management controllers in the system
- Allows non-intelligent devices as well as management controllers on the bus
- BMC serves as a controller to give system software access to IPMB.

Hardware implementation is isolated from software implementation so that new sensors and events can then be added without any software changes.

For more information, refer to: https://www.intel.com/content/www/us/en/products/docs/servers/ipmi/ipmi-home.html.

4.5 Advanced Telecom Computing Architecture (ATCA)

Advanced Telecom Computing Architecture (ATCA) is a follow-on to Compact PCI (cPCI), providing a standardized form-factor with larger card area, larger pitch and larger power supply for use in advanced rack-mounted telecom hardware. It includes a fault-tolerant scheme for thermal management that uses I^2C-bus communications between boards.

Advanced Telecom Computing Architecture (ATCA) is backed by more than 100 companies including many of the large players such as Intel, Lucent, and Motorola.

There are two general compliant approaches to an ATCA-compliant fan control: the first is an Intelligent FRU (Field Replaceable Unit) which means that the fan control would be directly connected to the IPMB (Intelligent Platform Management Bus); the second is a Managed or Non-intelligent FRU.

One requirement is the inclusion of hardware and software to manage the dual I^2C-buses. This requires an on-board isolated supply to power the circuitry, a buffered dual I^2C-bus with rise time accelerators, and 3-state capability. The I^2C controller must be able to support a multi-controller I^2C dual bus and handle the standard set of fan commands outlined in the protocol. In addition, on-board temperature reporting, tray capability reporting, fan turn-off capabilities, and non-volatile storage are required.

For more information, refer to: https://www.picmg.org/openstandards/advancedtca/.

4.6 Display Data Channel (DDC)

The Display Data Channel (DDC) allows a monitor or display to inform the host about its identity and capabilities. The specification for DDC version 2 calls for compliance with the I^2C-bus standard mode specification. It allows bidirectional communication between the display and the host, enabling control of monitor functions such as how images are displayed and communication with other devices attached to the I^2C-bus.

For more information, refer to: https://vesa.org/.

5 Bus speeds

Originally, the I^2C-bus was limited to 100 kbit/s operation. Over time there have been several additions to the specification so that there are now five operating speed categories. Standard-mode, Fast-mode (Fm), Fast-mode Plus (Fm+), and High-speed mode (Hs-mode) devices are downward-compatible — any device may be operated at a lower bus speed. Ultra Fast-mode devices are not compatible with previous versions since the bus is unidirectional.

- Bidirectional bus:
  - **Standard-mode (Sm)**, with a bit rate up to 100 kbit/s
  - **Fast-mode (Fm)**, with a bit rate up to 400 kbit/s
  - **Fast-mode Plus (Fm+)**, with a bit rate up to 1 Mbit/s
  - **High-speed mode (Hs-mode)**, with a bit rate up to 3.4 Mbit/s.
- Unidirectional bus:
  - **Ultra Fast-mode (UFm)**, with a bit rate up to 5 Mbit/s

5.1 Fast-mode

Fast-mode devices can receive and transmit at up to 400 kbit/s. The minimum requirement is that they can synchronize with a 400 kbit/s transfer; they can then prolong the LOW period of the SCL signal to slow down the transfer. The protocol, format, logic levels and maximum capacitive load for the SDA and SCL lines are the same as the Standard-mode I^2C-bus specification. Fast-mode devices are downward-compatible and can communicate with Standard-mode devices in a 0 to 100 kbit/s I^2C-bus system. As Standard-mode devices, however, are not upward compatible; they should not be incorporated in a Fast-mode I^2C-bus system as they cannot follow the higher transfer rate and unpredictable states would occur.

The Fast-mode I^2C-bus specification has the following additional features compared with the Standard-mode:

- The maximum bit rate is increased to 400 kbit/s.
- Timing of the serial data (SDA) and serial clock (SCL) signals has been adapted. There is no need for compatibility with other bus systems such as CBUS because they cannot operate at the increased bit rate.
- The inputs of Fast-mode devices incorporate spike suppression and a Schmitt trigger at the SDA and SCL inputs.
- The output buffers of Fast-mode devices incorporate slope control of the falling edges of the SDA and SCL signals.
- If the power supply to a Fast-mode device is switched off, the SDA and SCL I/O pins must be floating so that they do not obstruct the bus lines.

The external pull-up devices connected to the bus lines must be adapted to accommodate the shorter maximum permissible rise time for the Fast-mode I^2C-bus. For bus loads up to 200 pF, the pull-up device for each bus line can be a resistor; for bus loads between 200 pF and 400 pF, the pull-up device can be a current source (3 mA max.) or a switched resistor circuit (see Section 7.2.4).

5.2 Fast-mode Plus

Fast-mode Plus (Fm+) devices offer an increase in I^2C-bus transfer speeds and total bus capacitance. Fm+ devices can transfer information at bit rates of up to 1 Mbit/s, yet they remain fully downward compatible with Fast- or Standard-mode devices for bidirectional communication in a mixed-speed bus system. The same serial bus protocol and data format is maintained as with the Fast- or Standard-mode system. Fm+ devices also offer increased drive current over Fast- or Standard-mode devices allowing them to drive longer and/or more heavily loaded buses so that bus buffers do not need to be used.

The drivers in Fast-mode Plus parts are strong enough to satisfy the Fast-mode Plus timing specification with the same 400 pF load as Standard-mode parts. To be backward compatible with Standard-mode, they are also tolerant of the 1 µs rise time of Standard-mode parts. In applications where only Fast-mode Plus parts are present, the high drive strength and tolerance for slow rise and fall times allow the use of larger bus capacitance as long as set-up, minimum LOW time and minimum HIGH time for Fast-mode Plus are all satisfied and the fall time and rise time do not exceed the 300 ns t_f and 1 µs t_r specifications of Standard-mode. Bus speed can be traded against load capacitance to increase the maximum capacitance by about a factor of ten.

5.3 Hs-mode

High-speed mode (Hs-mode) devices offer a quantum leap in I^2C-bus transfer speeds. Hs-mode devices can transfer information at bit rates of up to 3.4 Mbit/s, yet they remain fully downward compatible with Fast-mode Plus, Fast- or Standard-mode (F/S) devices for bidirectional communication in a mixed-speed bus system. With the exception that arbitration and clock synchronization is not performed during the Hs-mode transfer, the same serial bus protocol and data format is maintained as with the F/S-mode system.

5.3.1 High speed transfer

To achieve a bit transfer of up to 3.4 Mbit/s, the following improvements have been made to the regular I^2C-bus specification:

- Hs-mode controller devices have an open-drain output buffer for the SDAH signal and a combination of an open-drain pull-down and current-source pull-up circuit on the SCLH output. This current-source circuit shortens the rise time of the SCLH signal. Only the current-source of one controller is enabled at any one time, and only during Hs-mode.
- No arbitration or clock synchronization is performed during Hs-mode transfer in multi-controller systems, which speeds-up bit handling capabilities. The arbitration procedure always finishes after a preceding controller code transmission in F/S-mode.
- Hs-mode controller devices generate a serial clock signal with a HIGH to LOW ratio of 1 to 2. This relieves the timing requirements for set-up and hold times.
- As an option, Hs-mode controller devices can have a built-in bridge. During Hs-mode transfer, the high-speed data (SDAH) and high-speed serial clock (SCLH) lines of Hs-mode devices are separated by this bridge from the SDA and SCL lines of F/S-mode

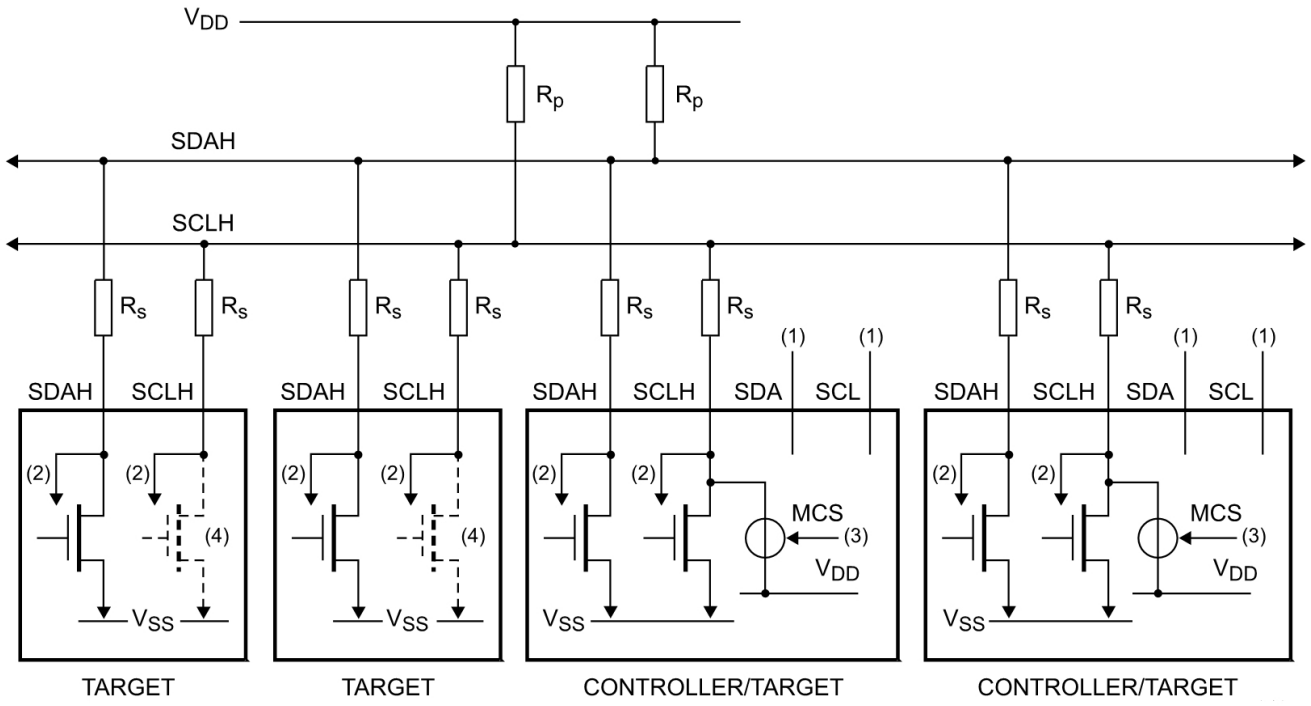
devices. This reduces the capacitive load of the SDAH and SCLH lines resulting in faster rise and fall times.

- The only difference between Hs-mode target devices and F/S-mode target devices is the speed at which they operate. Hs-mode targets have open-drain output buffers on the SCLH and SDAH outputs. Optional pull-down transistors on the SCLH pin can be used to stretch the LOW level of the SCLH signal, although this is only allowed after the acknowledge bit in Hs-mode transfers.
- The inputs of Hs-mode devices incorporate spike suppression and a Schmitt trigger at the SDAH and SCLH inputs.
- The output buffers of Hs-mode devices incorporate slope control of the falling edges of the SDAH and SCLH signals.

Figure 32 shows the physical I^2C-bus configuration in a system with only Hs-mode devices. Pins SDA and SCL on the controller devices are only used in mixed-speed bus systems and are not connected in an Hs-mode only system. In such cases, these pins can be used for other functions.

Optional series resistors R_s protect the I/O stages of the I^2C-bus devices from high-voltage spikes on the bus lines and minimize ringing and interference.

Pull-up resistors R_p maintain the SDAH and SCLH lines at a HIGH level when the bus is free and ensure that the signals are pulled up from a LOW to a HIGH level within the required rise time. For higher capacitive bus-line loads (>100 pF), the resistor R_p can be replaced by external current source pull-ups to meet the rise time requirements. Unless proceeded by an acknowledge bit, the rise time of the SCLH clock pulses in Hs-mode transfers is shortened by the internal current-source pull-up circuit MCS of the active controller.



1. SDA and SCL are not used here but may be used for other functions.
2. To input filter.
3. Only the active controller can enable its current-source pull-up circuit.
4. Dotted transistors are optional open-drain outputs which can stretch the serial clock signal SCLH.

Figure 32. I^2C-bus configuration with Hs-mode devices only

5.3.2 Serial data format in Hs-mode

Serial data transfer format in Hs-mode meets the Standard-mode I^2C-bus specification. Hs-mode can only commence after the following conditions (all of which are in F/S-mode):

1. START condition (S)
2. 8-bit controller code (0000 1XXX)
3. Not-acknowledge bit ($\overline{A}$)

Figure 33 and Figure 34 show this in more detail. This controller code has two main functions:

- It allows arbitration and synchronization between competing controllers at F/S-mode speeds, resulting in one winning controller.
- It indicates the beginning of an Hs-mode transfer.

Hs-mode controller codes are reserved 8-bit codes, which are not used for target addressing or other purposes. Furthermore, as each controller has its own unique controller code, up to eight Hs-mode controllers can be present on the one I^2C-bus system (although controller code 0000 1000 should be reserved for test and diagnostic purposes). The controller code for an Hs-mode controller device is software programmable and is chosen by the System Designer.

Arbitration and clock synchronization only take place during the transmission of the controller code and not-acknowledge bit ($\overline{A}$), after which one winning controller remains active. The controller code indicates to other devices that an Hs-mode transfer is to begin and the connected devices must meet the Hs-mode specification. As no device is allowed to acknowledge the controller code, the controller code is followed by a not-acknowledge ($\overline{A}$).

After the not-acknowledge bit ($\overline{A}$), and the SCLH line has been pulled-up to a HIGH level, the active controller switches to Hs-mode and enables (at time t_H, see Figure 34) the current-source pull-up circuit for the SCLH signal. As other devices can delay the serial transfer before t_H by stretching the LOW period of the SCLH signal, the active controller enables its current-source pull-up circuit when all devices have released the SCLH line and the SCLH signal has reached a HIGH level, thus speeding up the last part of the rise time of the SCLH signal.

The active controller then sends a repeated START condition (Sr) followed by a 7-bit target address (or 10-bit target address, see Section 3.1.11) with a R/$\overline{W}$ bit address, and receives an acknowledge bit (A) from the selected target.

After a repeated START condition and after each acknowledge bit (A) or not-acknowledge bit ($\overline{A}$), the active controller disables its current-source pull-up circuit. This enables other devices to delay the serial transfer by stretching the LOW period of the SCLH signal. The active controller re-enables its current-source pull-up circuit again when all devices have released and the SCLH signal reaches a HIGH level, and so speeds up the last part of the SCLH signal's rise time.

Data transfer continues in Hs-mode after the next repeated START (Sr), and only switches back to F/S-mode after a STOP condition (P). To reduce the overhead of the controller code, it is possible that a controller links a number of Hs-mode transfers, separated by repeated START conditions (Sr).

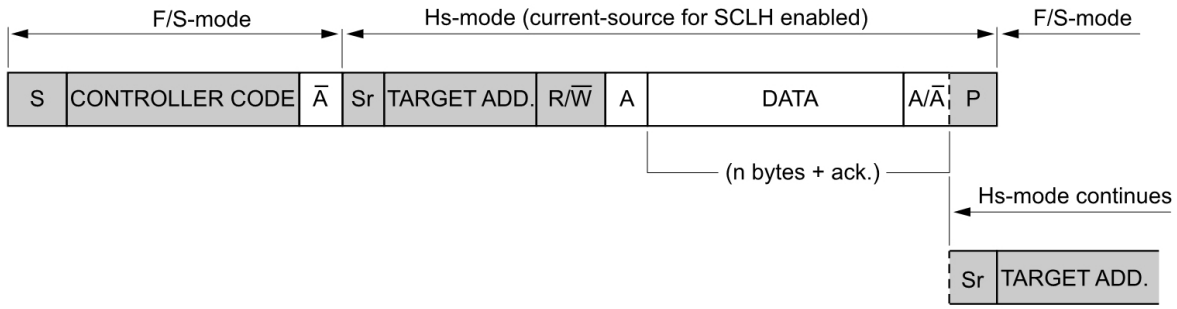


Figure 33. Data transfer format in Hs-mode

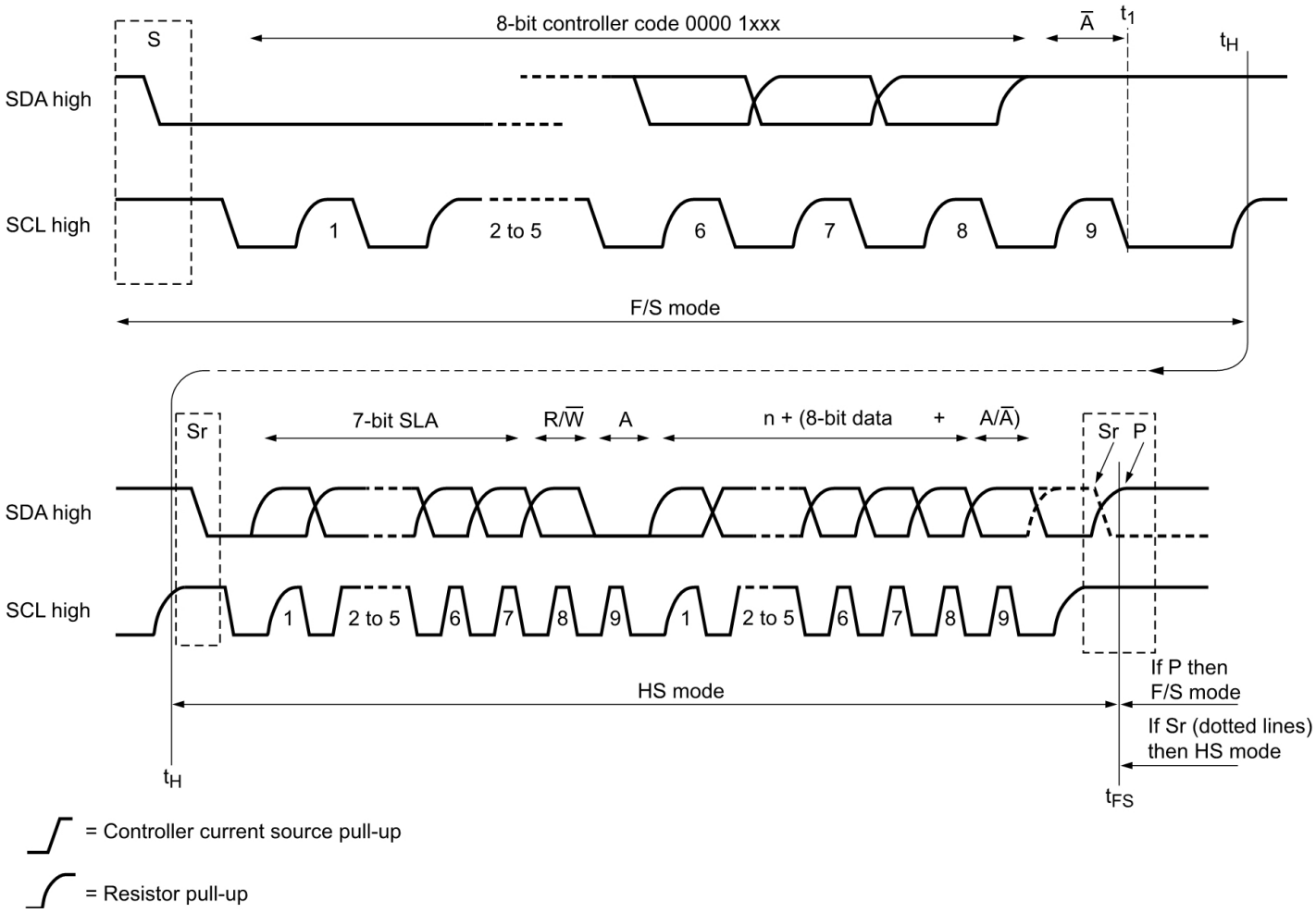


Figure 34. A complete Hs-mode transfer

5.3.3 Switching from F/S-mode to Hs-mode and back

After reset and initialization, Hs-mode devices must be in Fast-mode (which is in effect F/S-mode, as Fast-mode is downward compatible with Standard-mode). Each Hs-mode device can switch from Fast-mode to Hs-mode and back and is controlled by the serial transfer on the I^2C-bus.

Before time t_1 in Figure 34, each connected device operates in Fast-mode. Between times t_1 and t_H (this time interval can be stretched by any device) each connected device must recognize the ‘S 00001XXX A’ sequence and has to switch its internal circuit from the Fast-mode setting to the Hs-mode setting. Between times t_1 and t_H, the connected controller and target devices perform this switching by the following actions.

The active (winning) controller:

1. Adapts its SDAH and SCLH input filters according to the spike suppression requirement in Hs-mode.
2. Adapts the set-up and hold times according to the Hs-mode requirements.
3. Adapts the slope control of its SDAH and SCLH output stages according to the Hs-mode requirement.
4. Switches to the Hs-mode bit-rate, which is required after time t_H.
5. Enables the current source pull-up circuit of its SCLH output stage at time t_H.

The non-active, or losing controllers:

1. Adapt their SDAH and SCLH input filters according to the spike suppression requirement in Hs-mode.
2. Wait for a STOP condition to detect when the bus is free again.

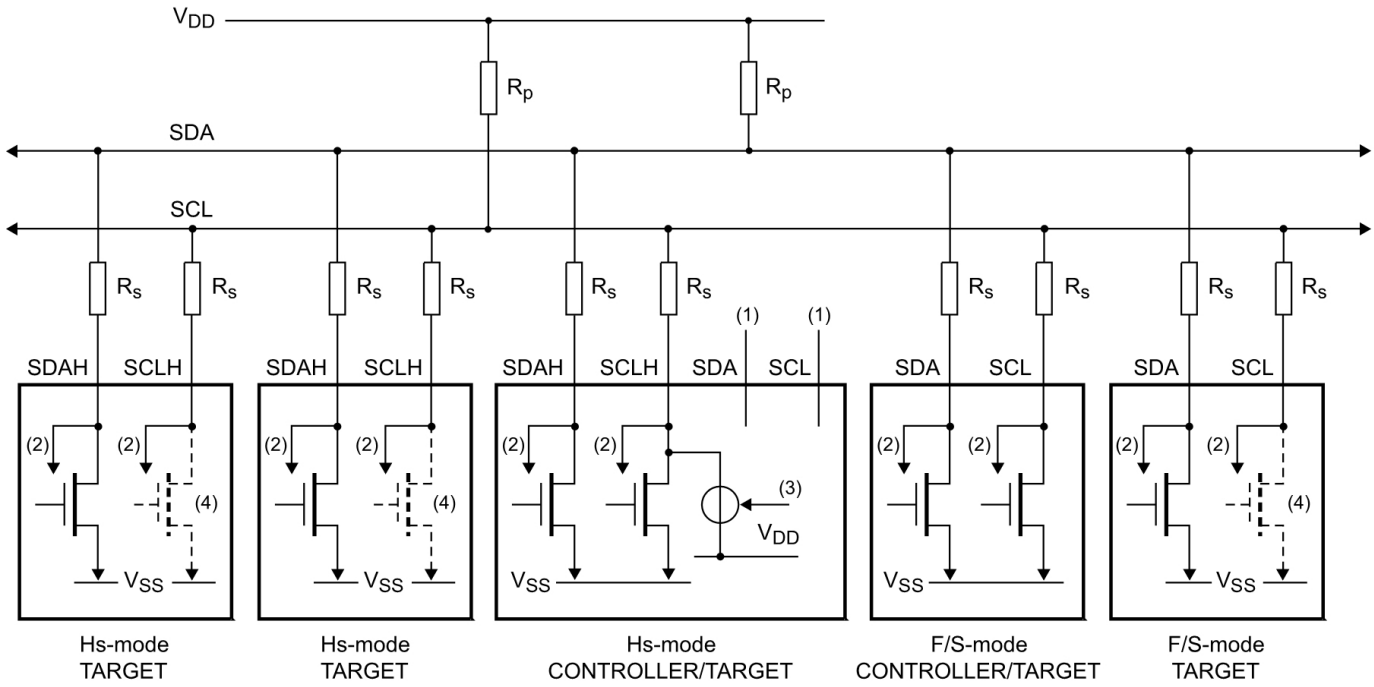
All targets:

1. Adapt their SDAH and SCLH input filters according to the spike suppression requirement in Hs-mode.
2. Adapt the set-up and hold times according to the Hs-mode requirements. This requirement may already be fulfilled by the adaptation of the input filters.
3. Adapt the slope control of their SDAH output stages, if necessary. For target devices, slope control is applicable for the SDAH output stage only and, depending on circuit tolerances, both the Fast-mode and Hs-mode requirements may be fulfilled without switching its internal circuit.

At time t_{FS} in Figure 34, each connected device must recognize the STOP condition (P) and switch its internal circuit from the Hs-mode setting back to the Fast-mode setting as present before time t_1. This must be completed within the minimum bus free time as specified in Table 10 according to the Fast-mode specification.

5.3.4 Hs-mode devices at lower speed modes

Hs-mode devices are fully downwards compatible, and can be connected to an F/S-mode I^2C-bus system (see Figure 35). As no controller code is transmitted in such a configuration, all Hs-mode controller devices stay in F/S-mode and communicate at F/S-mode speeds with their current-source disabled. The SDAH and SCLH pins are used to connect to the F/S-mode bus system, allowing the SDA and SCL pins (if present) on the Hs-mode controller device to be used for other functions.



Bridge not used. SDA and SCL may have an alternative function.
To input filter.
The current-source pull-up circuit stays disabled.
Dotted transistors are optional open-drain outputs which can stretch the serial clock signal SCL.

Figure 35. Hs-mode devices at F/S-mode speed

5.3.5 Mixed speed modes on one serial bus system

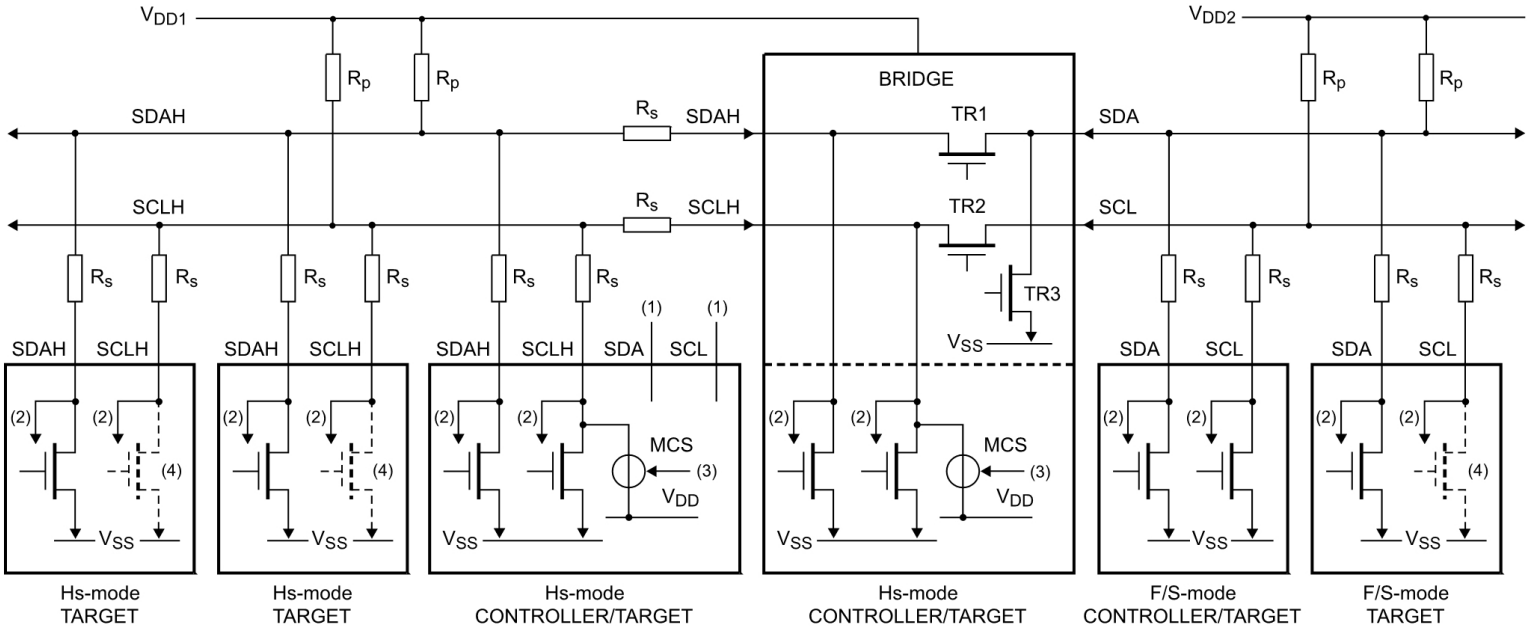
If a system has a combination of Hs-mode, Fast-mode and/or Standard-mode devices, it is possible, by using an interconnection bridge, to have different bit rates between different devices (see Figure 36 and Figure 37).

One bridge is required to connect/disconnect an Hs-mode section to/from an F/S-mode section at the appropriate time. This bridge includes a level shift function that allows devices with different supply voltages to be connected. For example F/S-mode devices with a V_{DD2} of 5 V can be connected to Hs-mode devices with a V_{DD1} of 3 V or less (that is, where $V_{DD2} \geq V_{DD1}$), provided SDA and SCL pins are 5 V tolerant. This bridge is incorporated in Hs-mode controller devices and is completely controlled by the serial signals SDAH, SCLH, SDA and SCL. Such a bridge can be implemented in any IC as an autonomous circuit.

TR1, TR2 and TR3 are N-channel transistors. TR1 and TR2 have a transfer gate function, and TR3 is an open-drain pull-down stage. If TR1 or TR2 are switched on they transfer a LOW level in both directions, otherwise when both the drain and source rise to a HIGH level there is a high-impedance between the drain and source of each switched-on transistor. In the latter case, the transistors act as a level shifter as SDAH and SCLH are pulled-up to V_{DD1} and SDA and SCL are pulled-up to V_{DD2}.

During F/S-mode speed, a bridge on one of the Hs-mode controllers connects the SDAH and SCLH lines to the corresponding SDA and SCL lines thus permitting Hs-mode devices to communicate with F/S-mode devices at slower speeds. Arbitration and synchronization are possible during the total F/S-mode transfer between all connected devices as described in Section 3.1.7. During Hs-mode transfer, however, the bridge opens to separate the two bus sections and allows Hs-mode devices to communicate with each other at 3.4 Mbit/s. Arbitration between Hs-mode devices and F/S-mode devices is only performed during the controller code (0000 1XXX), and normally won by

one Hs-mode controller as no target address has four leading zeros. Other controllers can win the arbitration only if they send a reserved 8-bit code (0000 0XXX). In such cases, the bridge remains closed and the transfer proceeds in F/S-mode. Table 9 gives the possible communication speeds in such a system.



1. Bridge not used. SDA and SCL may have an alternative function.
2. To input filter.
3. Only the active controller can enable its current-source pull-up circuit.
4. Dotted transistors are optional open-drain outputs which can stretch the serial clock signal SCL or SCLH.

Figure 36. Bus system with transfer at Hs-mode and F/S-mode speeds

Table 9. Communication bit rates in a mixed-speed bus system

Transfer between	Serial bus system configuration			
	Hs + Fast + Standard	Hs + Fast	Hs + Standard	Fast + Standard
Hs ↔ Hs	0 to 3.4 Mbit/s	0 to 3.4 Mbit/s	0 to 3.4 Mbit/s	-
Hs ↔ Fast	0 to 100 kbit/s	0 to 400 kbit/s	-	-
Hs ↔ Standard	0 to 100 kbit/s	-	0 to 100 kbit/s	-
Fast ↔ Standard	0 to 100 kbit/s	-	-	0 to 100 kbit/s
Fast ↔ Fast	0 to 100 kbit/s	0 to 400 kbit/s	-	0 to 100 kbit/s
Standard ↔ Standard	0 to 100 kbit/s	-	0 to 100 kbit/s	0 to 100 kbit/s

Remark: Table 9 assumes that the Hs devices are isolated from the Fm and Sm devices when operating at 3.4 Mbit/s. The bus speed is always constrained to the maximum communication rate of the slowest device attached to the bus.

5.3.6 Standard, Fast-mode and Fast-mode Plus transfer in a mixed-speed bus system

The bridge shown in Figure 36 interconnects corresponding serial bus lines, forming one serial bus system. As no controller code (0000 1XXX) is transmitted, the current-source

pull-up circuits stay disabled and all output stages are open-drain. All devices, including Hs-mode devices, communicate with each other according to the protocol, format and speed of the F/S-mode I^2C-bus specification.

5.3.7 Hs-mode transfer in a mixed-speed bus system

Figure 37 shows the timing diagram of a complete Hs-mode transfer, which is invoked by a START condition, a controller code, and a not-acknowledge $\overline{A}$ (at F/S-mode speed). Although this timing diagram is split in two parts, it should be viewed as one timing diagram were time point t_H is a common point for both parts.

The controller code is recognized by the bridge in the active or non-active controller (see Figure 36). The bridge performs the following actions:

1. Between t_1 and t_H (see Figure 37), transistor TR1 opens to separate the SDAH and SDA lines, after which transistor TR3 closes to pull-down the SDA line to V_{SS}.
2. When both SCLH and SCL become HIGH (t_H in Figure 37), transistor TR2 opens to separate the SCLH and SCL lines. TR2 must be opened before SCLH goes LOW after Sr.

Hs-mode transfer starts after t_H with a repeated START condition (Sr). During Hs-mode transfer, the SCL line stays at a HIGH and the SDA line at a LOW steady-state level, and so is prepared for the transfer of a STOP condition (P).

After each acknowledge (A) or not-acknowledge bit ($\overline{A}$), the active controller disables its current-source pull-up circuit. This enables other devices to delay the serial transfer by stretching the LOW period of the SCLH signal. The active controller re-enables its current-source pull-up circuit again when all devices are released and the SCLH signal reaches a HIGH level, and so speeds up the last part of the SCLH signal rise time. In irregular situations, F/S-mode devices can close the bridge (TR1 and TR2 closed, TR3 open) at any time by pulling down the SCL line for at least 1 µs, for example, to recover from a bus hang-up.

Hs-mode finishes with a STOP condition and brings the bus system back into the F/S-mode. The active controller disables its current-source MCS when the STOP condition (P) at SDAH is detected (t_{FS} in Figure 37). The bridge also recognizes this STOP condition and takes the following actions:

1. Transistor TR2 closes after t_{FS} to connect SCLH with SCL; both of which are HIGH at this time. Transistor TR3 opens after t_{FS}, which releases the SDA line and allows it to be pulled HIGH by the pull-up resistor R_p. This is the STOP condition for the F/S-mode devices. TR3 must open fast enough to ensure the bus free time between the STOP condition and the earliest next START condition is according to the Fast-mode specification (see t_{BUF} in Table 10).
2. When SDA reaches a HIGH (t_2 in Figure 37), transistor TR1 closes to connect SDAH with SDA. (Note: interconnections are made when all lines are HIGH, thus preventing spikes on the bus lines.) TR1 and TR2 must be closed within the minimum bus free time according to the Fast-mode specification (see t_{BUF} in Table 10).

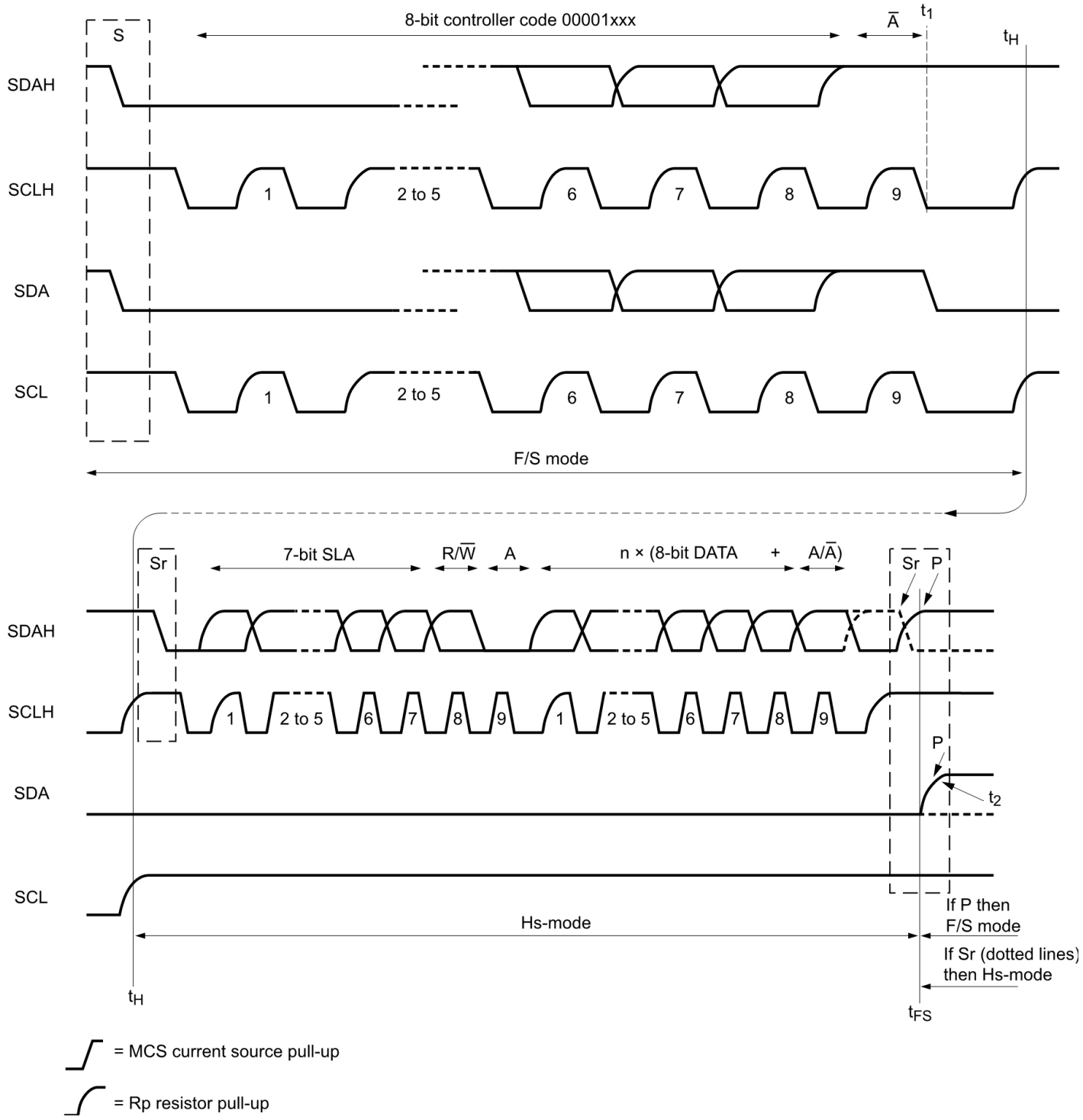


Figure 37. A complete Hs-mode transfer in a mixed-speed bus system

5.3.8 Timing requirements for the bridge in a mixed-speed bus system

It can be seen from Figure 37 that the actions of the bridge at t_1, t_H and t_{FS} must be so fast that it does not affect the SDAH and SCLH lines. Furthermore the bridge must meet the related timing requirements of the Fast-mode specification for the SDA and SCL lines.

5.4 Ultra Fast-mode

Ultra Fast-mode (UFm) devices offer an increase in I^2C-bus transfer speeds. UFm devices can transfer information at bit rates of up to 5 Mbit/s. UFm devices offer push-pull drivers, eliminating the pull-up resistors, allowing higher transfer rates. The same serial

bus protocol and data format is maintained as with the Sm, Fm, or Fm+ system. UFm bus devices are not compatible with bidirectional I^2C-bus devices.

6 Electrical specifications and timing for I/O stages and bus lines

6.1 Standard-, Fast-, and Fast-mode Plus devices

The I/O levels, I/O current, spike suppression, output slope control and pin capacitance are given in Table 10. The I^2C-bus timing characteristics, bus-line capacitance and noise margin are given in Table 10. Figure 38 shows the timing definitions for the I^2C-bus.

The minimum HIGH and LOW periods of the SCL clock specified in Table 10 determine the maximum bit transfer rates of 100 kbit/s for Standard-mode devices, 400 kbit/s for Fast-mode devices, and 1000 kbit/s for Fast-mode Plus. Devices must be able to follow transfers at their own maximum bit rates, either by being able to transmit or receive at that speed or by applying the clock synchronization procedure described in Section 3.1.7 which forces the controller into a wait state and stretch the LOW period of the SCL signal. In the latter case, the bit transfer rate is reduced.

Table 10. Characteristics of the SDA and SCL I/O stages

n/a = not applicable.

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
V_{IL}	LOW-level input voltage [1]		-0.5	$0.3V_{DD}$	-0.5	$0.3V_{DD}$	-0.5	$0.3V_{DD}$	V
V_{IH}	HIGH-level input voltage [1]		$0.7V_{DD}$	[2]	$0.7V_{DD}$	[2]	$0.7V_{DD}$ [1]	[2]	V
V_{hys}	hysteresis of Schmitt trigger inputs		-	-	$0.05V_{DD}$	-	$0.05V_{DD}$	-	V
V_{OL1}	LOW-level output voltage 1	(open-drain or open-collector) at 3 mA sink current; V_{DD} > 2 V	0	0.4	0	0.4	0	0.4	V
V_{OL2}	LOW-level output voltage 2	(open-drain or open-collector) at 2 mA sink current [3] ; 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
		V_{OL} = 0.6 V [4]	-	-	6	-	-	-	mA
t_{of}	output fall time from V_{IHmin} to V_{ILmax}		-	250 [5]	20 × (V_{DD} / 5.5 V) [6]	250 [5]	20 × (V_{DD} / 5.5 V) [6]	120 [7]	ns
t_{SP}	pulse width of spikes that must be suppressed by the input filter		-	-	0	50 [8]	0	50 [8]	ns
I_i	input current each I/O pin	$0.1V_{DD} < V_I < 0.9V_{DDmax}$	-10	+10	-10 [9]	+10 [9]	-10 [9]	+10 [9]	µA
C_i	capacitance for each I/O pin [10]		-	10	-	10	-	10	pF

[1] Some legacy Standard-mode devices had fixed input levels of V_{IL} = 1.5 V and V_{IH} = 3.0 V. Refer to component data sheets.

[2] Maximum $V_{IH} = V_{DD(max)}$ + 0.5 V or 5.5 V, which ever is lower. See component data sheets.

[3] The same resistor value to drive 3 mA at 3.0 V V_{DD} provides the same RC time constant when using <2 V V_{DD} with a smaller current draw.

[4] In order to drive full bus load at 400 kHz, 6 mA I_{OL} is required at 0.6 V V_{OL}. Parts not meeting this specification can still function, but not at 400 kHz and 400 pF.

[5] The maximum t_f for the SDA and SCL bus lines quoted in Table 10 (300 ns) is longer than the specified maximum t_{of} for the output stages (250 ns). This allows series protection resistors (R_s) to be connected between the SDA/SCL pins and the SDA/SCL bus lines as shown in Figure 45 without exceeding the maximum specified t_f.

[6] Necessary to be backwards compatible with Fast-mode.

[7] In Fast-mode Plus, fall time is specified the same for both output stage and bus timing. If series resistors are used, designers should allow for this when considering bus timing.
[8] Input filters on the SDA and SCL inputs suppress noise spikes of less than 50 ns.
[9] If V_{DD} is switched off, I/O pins of Fast-mode and Fast-mode Plus devices must not obstruct the SDA and SCL lines.
[10] Special purpose devices such as multiplexers and switches may exceed this capacitance because they connect multiple paths together.

Table 11. Characteristics of the SDA and SCL bus lines for Standard, Fast, and Fast-mode Plus I^2C-bus devices

All values referred to $V_{IH(min)}$ ($0.7V_{DD}$) and $V_{IL(max)}$ ($0.3V_{DD}$) levels (see Table 10).

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency		0	100	0	400	0	1000	kHz
$t_{HD;STA}$	hold time (repeated) START condition	After this period, the first clock pulse is generated.	4.0	-	0.6	-	0.26	-	µs
t_{LOW}	LOW period of the SCL clock		4.7	-	1.3	-	0.5	-	µs
t_{HIGH}	HIGH period of the SCL clock		4.0	-	0.6	-	0.26	-	µs
$t_{SU;STA}$	set-up time for a repeated START condition		4.7	-	0.6	-	0.26	-	µs
$t_{HD;DAT}$	data hold time [1]	CBUS compatible controllers (see Remark in Section 4.1)	5.0	-	-	-	-	-	µs
		I^2C-bus devices	0 [2]	- [3]	0 [2]	- [3]	0	-	µs
$t_{SU;DAT}$	data set-up time		250	-	100 [4]	-	50	-	ns
t_r	rise time of both SDA and SCL signals		-	1000	20	300	-	120	ns
t_f	fall time of both SDA and SCL signals [2] [5] [6] [7]		-	300	20 × (V_{DD} / 5.5 V)	300	20 × (V_{DD} / 5.5 V) [8]	120 [7]	ns
$t_{SU;STO}$	set-up time for STOP condition		4.0	-	0.6	-	0.26	-	µs
t_{BUF}	bus free time between a STOP and START condition		4.7	-	1.3	-	0.5	-	µs
C_b	capacitive load for each bus line [9]		-	400	-	400	-	550	pF
$t_{VD;DAT}$	data valid time [10]		-	3.45 [3]	-	0.9 [3]	-	0.45 [3]	µs
$t_{VD;ACK}$	data valid acknowledge time [11]		-	3.45 [3]	-	0.9 [3]	-	0.45 [3]	µs
V_{nL}	noise margin at the LOW level	for each connected device (including hysteresis)	$0.1V_{DD}$	-	$0.1V_{DD}$	-	$0.1V_{DD}$	-	V
V_{nH}	noise margin at the HIGH level	for each connected device (including hysteresis)	$0.2V_{DD}$	-	$0.2V_{DD}$	-	$0.2V_{DD}$	-	V

[1] $t_{HD;DAT}$ is the data hold time that is measured from the falling edge of SCL, applies to data in transmission and the acknowledge.
[2] Ensure SCL drops below $0.3V_{DD}$ on falling edge before SDA crosses into the indeterminate range of 0.3 V_{DD} to 0.7 V_{DD}.
NOTE: For controllers that cannot observe the SCL falling edge then independent measurement of the time for the SCL transition from static high (V_{DD}) to 0.3 V_{DD} should be used to insert a delay of the SDA transition with respect to SCL.
[3] The maximum $t_{HD;DAT}$ could be 3.45 µs and 0.9 µs for Standard-mode and Fast-mode, but must be less than the maximum of $t_{VD;DAT}$ or $t_{VD;ACK}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. If the clock stretches the SCL, the data must be valid by the set-up time before it releases the clock.
[4] A Fast-mode I^2C-bus device can be used in a Standard-mode I^2C-bus system, but the requirement $t_{SU;DAT}$ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line $t_{r(max)}$ + $t_{SU;DAT}$ = 1000 + 250 = 1250 ns (according to the Standard-mode I^2C-bus specification) before the SCL line is released. Also the acknowledge timing must meet this set-up time.
[5] If mixed with Hs-mode devices, faster fall times according to Table 10 are allowed.

[6] The maximum t_f for the SDA and SCL bus lines is specified at 300 ns. The maximum fall time for the SDA output stage t_f is specified at 250 ns. This allows series protection resistors to be connected in between the SDA and the SCL pins and the SDA/SCL bus lines without exceeding the maximum specified t_f.

[7] In Fast-mode Plus, fall time is specified the same for both output stage and bus timing. If series resistors are used, designers should allow for this when considering bus timing.

[8] Necessary to be backwards compatible to Fast-mode.

[9] The maximum bus capacitance allowable may vary from this value depending on the actual operating voltage and frequency of the application. Section 7.2 discusses techniques for coping with higher bus capacitances.

[10] $t_{VD;DAT}$ = time for data signal from SCL LOW to SDA output (HIGH or LOW, depending on which one is worse).

[11] $t_{VD;ACK}$ = time for Acknowledgement signal from SCL LOW to SDA output (HIGH or LOW, depending on which one is worse).

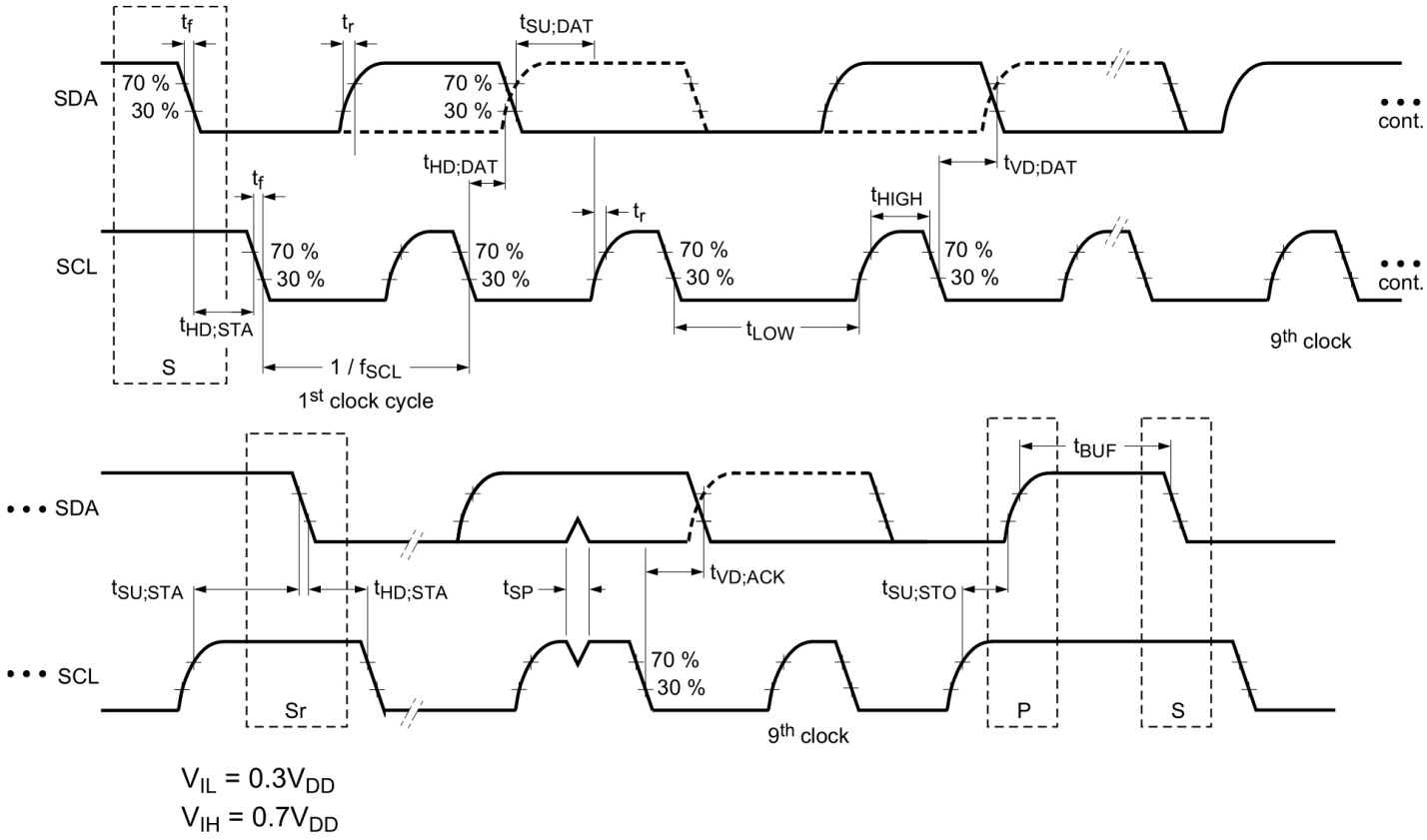


Figure 38. Definition of timing for F/S-mode devices on the I²C-bus

6.2 Hs-mode devices

The I/O levels, I/O current, spike suppression, output slope control and pin capacitance for I²C-bus Hs-mode devices are given in Table 12. The noise margin for HIGH and LOW levels on the bus lines are the same as specified for F/S-mode I²C-bus devices.

Figure 39 shows all timing parameters for the Hs-mode timing. The 'normal' START condition S does not exist in Hs-mode. Timing parameters for Address bits, R/$\overline{W}$ bit, Acknowledge bit and DATA bits are all the same. Only the rising edge of the first SCLH clock signal after an acknowledge bit has a larger value because the external R_p has to pull up SCLH without the help of the internal current-source.

The Hs-mode timing parameters for the bus lines are specified in Table 12. The minimum HIGH and LOW periods and the maximum rise and fall times of the SCLH clock signal determine the highest bit rate.

With an internally generated SCLH signal with LOW and HIGH level periods of 200 ns and 100 ns respectively, an Hs-mode controller fulfills the timing requirements for the external SCLH clock pulses (taking the rise and fall times into account) for the maximum bit rate of 3.4 Mbit/s. So a basic frequency of 10 MHz, or a multiple of 10 MHz, can be used by an Hs-mode controller to generate the SCLH signal. There are no limits for

maximum HIGH and LOW periods of the SCLH clock, and there is no limit for a lowest bit rate.

Timing parameters are independent for capacitive load up to 100 pF for each bus line allowing the maximum possible bit rate of 3.4 Mbit/s. At a higher capacitive load on the bus lines, the bit rate decreases gradually. The timing parameters for a capacitive bus load of 400 pF are specified in Table 12, allowing a maximum bit rate of 1.7 Mbit/s. For capacitive bus loads between 100 pF and 400 pF, the timing parameters must be interpolated linearly. Rise and fall times are in accordance with the maximum propagation time of the transmission lines SDAH and SCLH to prevent reflections of the open ends.

Table 12. Characteristics of the SDAH, SCLH, SDA and SCL I/O stages for Hs-mode I^2C-bus devices

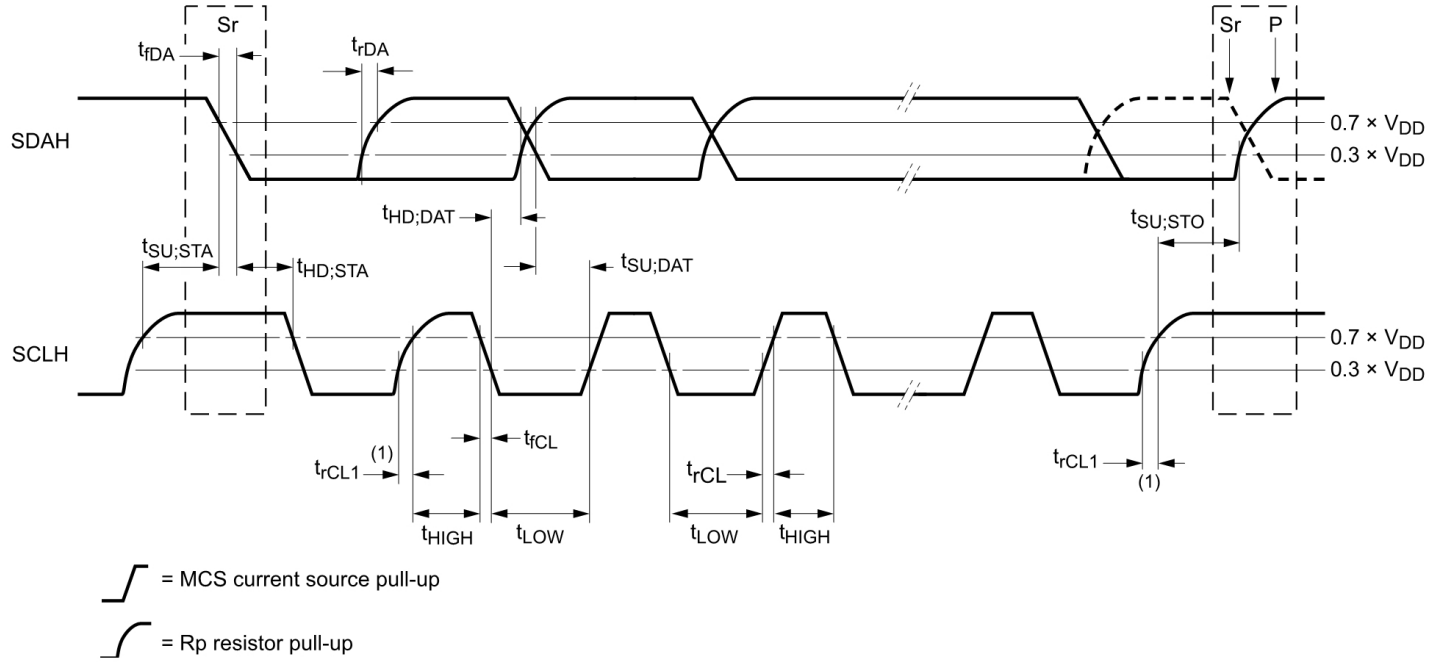
Symbol	Parameter	Conditions	Hs-mode		Unit
			Min	Max	
V_{IL}	LOW-level input voltage		-0.5	$0.3V_{DD}$[1]	V
V_{IH}	HIGH-level input voltage		$0.7V_{DD}$[1]	$V_{DD} + 0.5$[2]	V
V_{hys}	hysteresis of Schmitt trigger inputs		$0.1V_{DD}$[1]	-	V
V_{OL}	LOW-level output voltage	(open-drain) at 3 mA sink current at SDAH, SDA and SCLH			
		$V_{DD} > 2$ V	0	0.4	V
		$V_{DD} \leq 2$ V	0	$0.2V_{DD}$	V
R_{onL}	transfer gate on resistance for currents between SDA and SDAH or SCL and SCLH	V_{OL} level; I_{OL} = 3 mA	-	50	Ω
R_{onH}[2]	transfer gate on resistance between SDA and SDAH or SCL and SCLH	both signals (SDA and SDAH, or SCL and SCLH) at V_{DD} level	50	-	kΩ
I_{CS}	pull-up current of the SCLH current-source	SCLH output levels between $0.3V_{DD}$ and $0.7V_{DD}$	3	12	mA
t_{rCL}	rise time of SCLH signal	output rise time (current-source enabled) with an external pull-up current source of 3 mA			
		capacitive load from 10 pF to 100 pF	10	40	ns
		capacitive load of 400 pF[3]	20	80	ns
t_{fCL}	fall time of SCLH signal	output fall time (current-source enabled) with an external pull-up current source of 3 mA			
		capacitive load from 10 pF to 100 pF	10	40	ns
		capacitive load of 400 pF[3]	20	80	ns
t_{fDA}	fall time of SDAH signal	capacitive load from 10 pF to 100 pF	10	80	ns
		capacitive load of 400 pF[3]	20	160	ns
t_{SP}	pulse width of spikes that must be suppressed by the input filter	SDAH and SCLH	0	10	ns
I_i[4]	input current each I/O pin	input voltage between $0.1V_{DD}$ and $0.9V_{DD}$	-	10	µA
C_i	capacitance for each I/O pin[5]		-	10	pF

[1] Devices that use non-standard supply voltages which do not conform to the intended I^2C-bus system levels must relate their input levels to the V_{DD} voltage to which the pull-up resistors R_p are connected.
[2] Devices that offer the level shift function must tolerate a maximum input voltage of 5.5 V at SDA and SCL.
[3] For capacitive bus loads between 100 pF and 400 pF, the rise and fall time values must be linearly interpolated.
[4] If their supply voltage has been switched off, SDAH and SCLH I/O stages of Hs-mode target devices must have floating outputs. Due to the current-source output circuit, which normally has a clipping diode to V_{DD}, this requirement is not mandatory for the SCLH or the SDAH I/O stage of Hs-mode controller devices. This means that the supply voltage of Hs-mode controller devices cannot be switched off without affecting the SDAH and SCLH lines.
[5] Special purpose devices such as multiplexers and switches may exceed this capacitance because they connect multiple paths together.

Table 13. Characteristics of the SDAH, SCLH, SDA and SCL bus lines for Hs-mode I^2C-bus devices[1]

Symbol	Parameter	Conditions	C_b = 100 pF (max)		C_b = 400 pF[2]		Unit
			Min	Max	Min	Max	
f_{SCLH}	SCLH clock frequency		0	3.4	0	1.7	MHz
$t_{SU;STA}$	set-up time for a repeated START condition		160	-	160	-	ns
$t_{HD;STA}$	hold time (repeated) START condition		160	-	160	-	ns
t_{LOW}	LOW period of the SCL clock		160	-	320	-	ns
t_{HIGH}	HIGH period of the SCL clock		60	-	120	-	ns
$t_{SU;DAT}$	data set-up time		10	-	10	-	ns
$t_{HD;DAT}$	data hold time		0[3]	70	0[3]	150	ns
t_{rCL}	rise time of SCLH signal		10	40	20	80	ns
t_{rCL1}	rise time of SCLH signal after a repeated START condition and after an acknowledge bit		10	80	20	160	ns
t_{fCL}	fall time of SCLH signal		10	40	20	80	ns
t_{rDA}	rise time of SDAH signal		10	80	20	160	ns
t_{fDA}	fall time of SDAH signal		10	80	20	160	ns
$t_{SU;STO}$	set-up time for STOP condition		160	-	160	-	ns
C_b[2]	capacitive load for each bus line	SDAH and SCLH lines	-	100	-	400	pF
		SDAH + SDA line and SCLH + SCL line	-	400	-	400	pF
V_{nL}	noise margin at the LOW level	for each connected device (including hysteresis)	$0.1V_{DD}$	-	$0.1V_{DD}$	-	V
V_{nH}	noise margin at the HIGH level	for each connected device (including hysteresis)	$0.2V_{DD}$	-	$0.2V_{DD}$	-	V

[1] All values referred to $V_{IH(min)}$ and $V_{IL(max)}$ levels (see Table 12).
[2] For bus line loads C_b between 100 pF and 400 pF the timing parameters must be linearly interpolated.
[3] A device must internally provide a data hold time to bridge the undefined part between V_{IH} and V_{IL} of the falling edge of the SCLH signal. An input circuit with a threshold as low as possible for the falling edge of the SCLH signal minimizes this hold time.



1. First rising edge of the SCLH signal after Sr and after each acknowledge bit.

Figure 39. Definition of timing for Hs-mode devices on the I²C-bus

6.3 Ultra Fast-mode devices

The I/O levels, I/O current, spike suppression, output slope control and pin capacitance are given in Table 14. The UFm I²C-bus timing characteristics are given in Table 15. Figure 40 shows the timing definitions for the I²C-bus. The minimum HIGH and LOW periods of the SCL clock specified in Table 15 determine the maximum bit transfer rates of 5000 kbit/s for Ultra Fast-mode. Devices must be able to follow transfers at their own maximum bit rates, either by being able to transmit or receive at that speed.

Table 14. Characteristics of the USDA and USCL I/O stages

n/a = not applicable.

Symbol	Parameter	Conditions		Ultra Fast-mode		Unit
				Min	Max	
V_{IL}	LOW-level input voltage[1]			-0.5	$+0.3V_{DD}$	V
V_{IH}	HIGH-level input voltage[1]			$0.7V_{DD}$[1]	-[2]	V
V_{hys}	hysteresis of Schmitt trigger inputs			$0.05V_{DD}$	-	V
V_{OL}	LOW-level output voltage	at 4 mA sink current; V_{DD} > 2 V		0	0.4	V
V_{OH}	HIGH-level output voltage	at 4 mA source current; V_{DD} > 2 V		V_{DD} - 0.4	-	V
I_L	leakage current	V_{DD} = 3.6 V		-1	+1	µA
		V_{DD} = 5.5 V		-10	+10	µA
C_i	input capacitance		[3]	-	10	pF
t_{SP}	pulse width of spikes that must be suppressed by the input filter		[4]	-	10	ns

[1] Refer to component data sheets for actual switching points.
[2] Maximum $V_{IH} = V_{DD(max)}$ + 0.5 V or 5.5 V, whichever is lower. See component data sheets.
[3] Special purpose devices such as multiplexers and switches may exceed this capacitance because they connect multiple paths together.
[4] Input filters on the USDA and USCL target inputs suppress noise spikes of less than 10 ns.

Table 15. UFm I^2C-bus frequency and timing specifications

Symbol	Parameter	Conditions		Ultra Fast-mode		Unit
				Min	Max	
f_{USCL}	USCL clock frequency			0	5000	kHz
t_{BUF}	bus free time between a STOP and START condition			80	-	ns
$t_{HD;STA}$	hold time (repeated) START condition			50	-	ns
$t_{SU;STA}$	set-up time for a repeated START condition			50	-	ns
$t_{SU;STO}$	set-up time for STOP condition			50	-	ns
$t_{HD;DAT}$	data hold time			10	-	ns
$t_{VD;DAT}$	data valid time		[1]	10	-	ns
$t_{SU;DAT}$	data set-up time			30	-	ns
t_{LOW}	LOW period of the USCL clock			50	-	ns
t_{HIGH}	HIGH period of the USCL clock			50	-	ns
t_f	fall time of both USDA and USCL signals			-[2]	50	ns
t_r	rise time of both USDA and USCL signals			-[2]	50	ns

[1] $t_{VD;DAT}$ = minimum time for USDA data out to be valid following USCL LOW.
[2] Typical rise time or fall time for UFm signals is 25 ns measured from the 30 % level to the 70 % level (rise time) or from the 70 % level to the 30 % level (fall time).

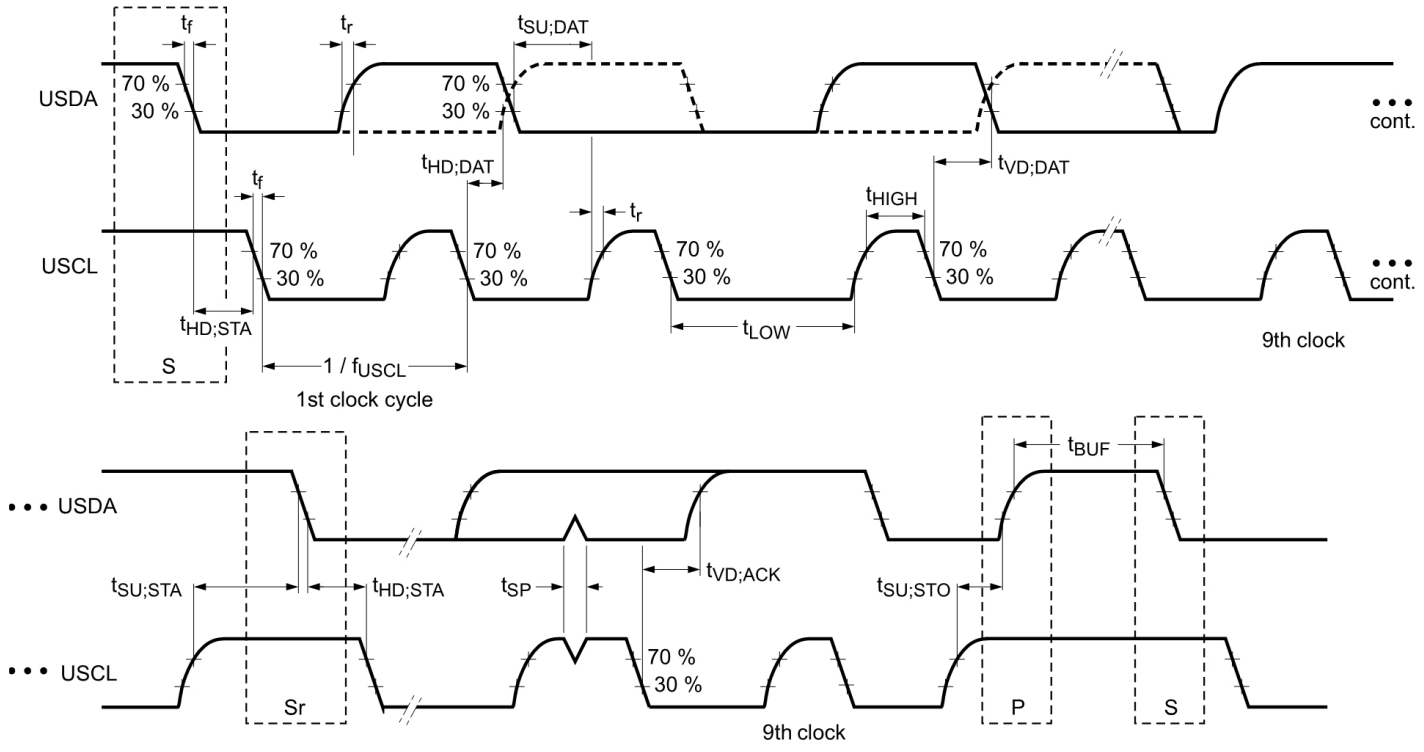


Figure 40. Definition of timing for Ultra Fast-mode devices on the I^2C-bus

7 Electrical connections of I²C-bus devices to the bus lines

7.1 Pull-up resistor sizing

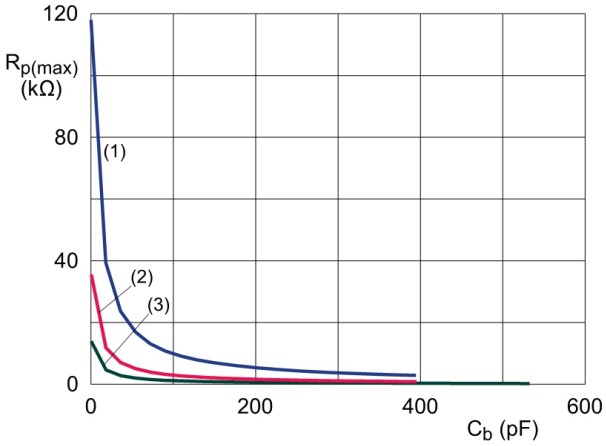
The bus capacitance is the total capacitance of wire, connections and pins. This capacitance limits the maximum value of R_p due to the specified rise time. Figure 41 shows $R_{p(max)}$ as a function of bus capacitance.

Consider the V_{DD} related input threshold of $V_{IH} = 0.7V_{DD}$ and $V_{IL} = 0.3V_{DD}$ for the purposes of RC time constant calculation. Then $V(t) = V_{DD} (1 - e^{-t / RC})$, where t is the time since the charging started and RC is the time constant.

$V(t1) = 0.3 \times V_{DD} = V_{DD} (1 - e^{-t1 / RC})$; then $t1 = 0.3566749 \times RC$
$V(t2) = 0.7 \times V_{DD} = V_{DD} (1 - e^{-t2 / RC})$; then $t2 = 1.2039729 \times RC$
$T = t2 - t1 = 0.8473 \times RC$

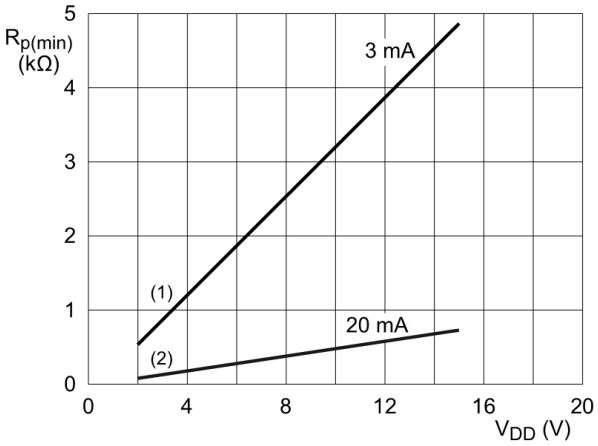
Figure 41 and Equation 1 shows maximum R_p as a function of bus capacitance for Standard-, Fast- and Fast-mode Plus. For each mode, the $R_{p(max)}$ is a function of the rise time maximum (t_r) from Table 10 and the estimated bus capacitance (C_b):

$$R_{p(max)} = \frac{t_r}{0.8473 \times C_b} \quad (1)$$



1. Standard-mode
2. Fast-mode
3. Fast-mode Plus

Figure 41. $R_{p(max)}$ as a function of bus capacitance



1. Fast-mode and Standard-mode
2. Fast-mode Plus

Figure 42. $R_{p(min)}$ as a function of V_{DD}

The supply voltage limits the minimum value of resistor R_p due to the specified minimum sink current of 3 mA for Standard-mode and Fast-mode, or 20 mA for Fast-mode Plus. $R_{p(min)}$ as a function of V_{DD} is shown in Figure 42. The traces are calculated using Equation 2:

$$R_{p(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL}} \quad (2)$$

The designer now has the minimum and maximum value of R_p that is required to meet the timing specification. Portable designs with sensitivity to supply current consumption can use a value toward the higher end of the range in order to limit I_{DD}.

7.2 Operating above the maximum allowable bus capacitance

Bus capacitance limit is specified to limit rise time reductions and allow operating at the rated frequency. While most designs can easily stay within this limit, some applications may exceed it. There are several strategies available to system designers to cope with excess bus capacitance.

- Reduced f_{SCL} (Section 7.2.1): The bus may be operated at a lower speed (lower f_{SCL}).
- Higher drive outputs (Section 7.2.2): Devices with higher drive current such as those rated for Fast-mode Plus can be used (PCA96xx).
- Bus buffers (Section 7.2.3): There are a number of bus buffer devices available that can divide the bus into segments so that each segment has a capacitance below the allowable limit, such as the PCA9517 bus buffer or the PCA9546A switch.
- Switched pull-up circuit (Section 7.2.4): A switched pull-up circuit can be used to accelerate rising edges by switching a low value pull-up alternately in and out when needed.

7.2.1 Reduced f_{SCL}

To determine a lower allowable bus operating frequency, begin by finding the t_{LOW} and t_{HIGH} of the most limiting device on the bus. Refer to individual component data sheets for these values. Actual rise time (t_r) depends on the RC time constant. The most limiting fall time (t_f) depends on the lowest output drive on the bus. Be sure to allow for any devices that have a minimum t_r or t_f. Refer to Equation 3 for the resulting f_{max}.

$$f_{max} = \frac{1}{t_{LOW(min)} + t_{HIGH(min)} + t_{r(actual)} + t_{f(actual)}} \quad (3)$$

Remark: Very long buses must also account for time of flight of signals.

Actual results are slower, as real parts do not tend to control t_{LOW} and t_{HIGH} to the minimum from 30 % to 70 %, or 70 % to 30 %, respectively.

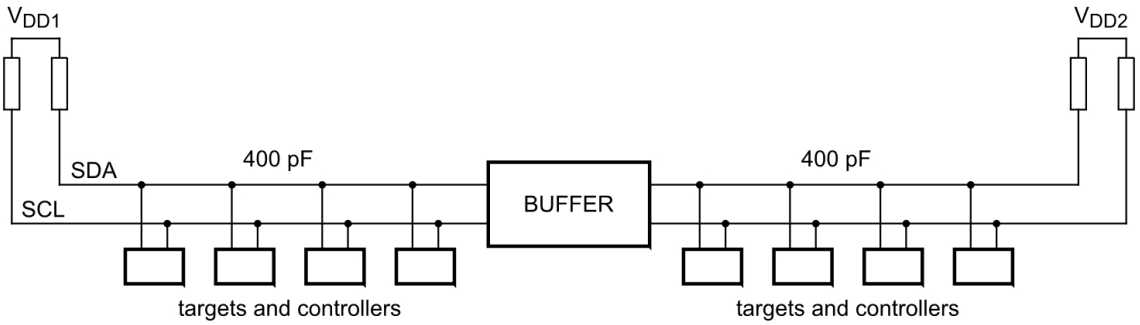
7.2.2 Higher drive outputs

If higher drive devices like the PCA96xx Fast-mode Plus or the P82B bus buffers are used, the higher strength output drivers sink more current which results in considerably faster edge rates, or, looked at another way, allows a higher bus capacitance. Refer to individual component data sheets for actual output drive capability. Repeat the calculation above using the new values of C_b, R_p, t_r and t_f to determine maximum frequency. Bear in mind that the maximum rating for f_{SCL} as specified in Table 10 (100 kHz, 400 kHz and 1000 kHz) may become limiting.

7.2.3 Bus buffers, multiplexers and switches

Another approach to coping with excess bus capacitance is to divide the bus into smaller segments using bus buffers, multiplexers or switches. Figure 43 shows an example of a bus that uses a PCA9515 buffer to deal with high bus capacitance. Each segment is then allowed to have the maximum capacitance so the total bus can have twice the maximum capacitance. Keep in mind that adding a buffer always adds delays — a buffer delay plus an additional transition time to each edge, which reduces the maximum operating frequency and may also introduce special V_{IL} and V_{OL} considerations.

Refer to application notes *AN255, I^2C / SMBus Repeaters, Hubs and Expanders* and *AN262, PCA954x Family of I^2C / SMBus Multiplexers and Switches* for more details on this subject and the devices available from NXP Semiconductors.



Remark: Some buffers allow V_{DD1} and V_{DD2} to be different levels.

Figure 43. Using a buffer to divide bus capacitance

7.2.4 Switched pull-up circuit

The supply voltage (V_{DD}) and the maximum output LOW level determine the minimum value of pull-up resistor R_p (see Section 7.1). For example, with a supply voltage of V_{DD} = 5 V ± 10 % and $V_{OL(max)}$ = 0.4 V at 3 mA, $R_{p(min)}$ = (5.5 - 0.4) / 0.003 = 1.7 kΩ. As shown in Figure 42, this value of R_p limits the maximum bus capacitance to about 200 pF to meet the maximum t_r requirement of 300 ns. If the bus has a higher capacitance than this, a switched pull-up circuit (as shown in Figure 44) can be used.

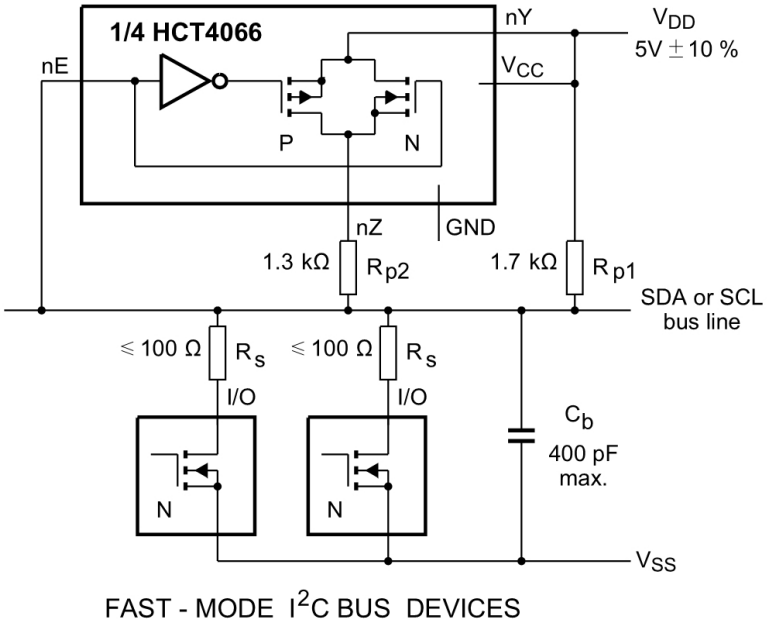


Figure 44. Switched pull-up circuit

The switched pull-up circuit in Figure 44 is for a supply voltage of V_{DD} = 5 V ± 10 % and a maximum capacitive load of 400 pF. Since it is controlled by the bus levels, it needs no additional switching control signals. During the rising/falling edges, the bilateral switch in the HCT4066 switches pull-up resistor R_{p2} on/off at bus levels between 0.8 V and 2.0 V. Combined resistors R_{p1} and R_{p2} can pull up the bus line within the maximum specified rise time (t_r) of 300 ns.

Series resistors R_s are optional. They protect the I/O stages of the I^2C-bus devices from high-voltage spikes on the bus lines, and minimize crosstalk and undershoot of the bus line signals. The maximum value of R_s is determined by the maximum permitted voltage drop across this resistor when the bus line is switched to the LOW level in order to switch off R_{p2}.

Additionally, some bus buffers contain integral rise time accelerators. Stand-alone rise time accelerators are also available.

7.3 Series protection resistors

As shown in Figure 45, series resistors (R_s) of, for example, 300 Ω can be used for protection against high-voltage spikes on the SDA and SCL lines (resulting from the flash-over of a TV picture tube, for example). If series resistors are used, designers must add the additional resistance into their calculations for R_p and allowable bus capacitance.

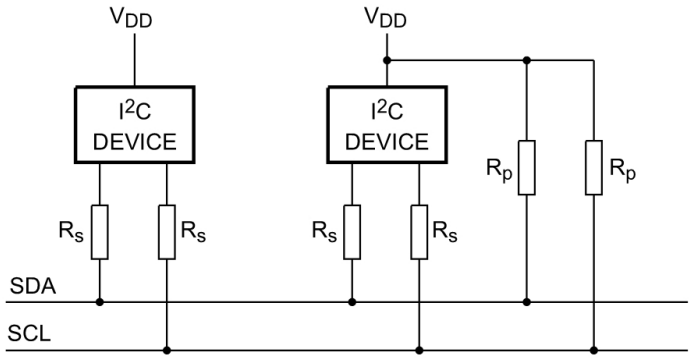


Figure 45. Series resistors (R_s) for protection against high-voltage spikes

The required noise margin of $0.1V_{DD}$ for the LOW level, limits the maximum value of R_s. $R_{s(max)}$ as a function of R_p is shown in Figure 46. Note that series resistors affect the output fall time.

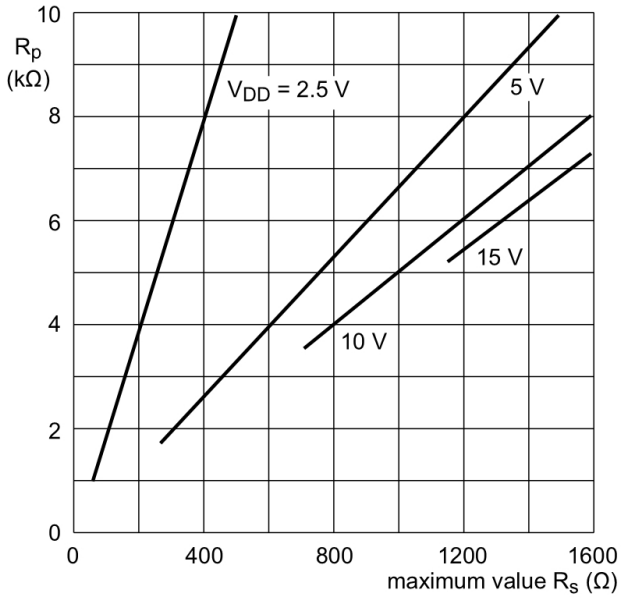


Figure 46. Maximum value of R_s as a function of the value of R_p with supply voltage as a parameter

7.4 Input leakage

The maximum HIGH level input current of each input/output connection has a specified maximum value of 10 µA. Due to the required noise margin of $0.2V_{DD}$ for the HIGH level, this input current limits the maximum value of R_p. This limit depends on V_{DD}. The total HIGH-level input current is shown as a function of $R_{p(max)}$ in Figure 47.

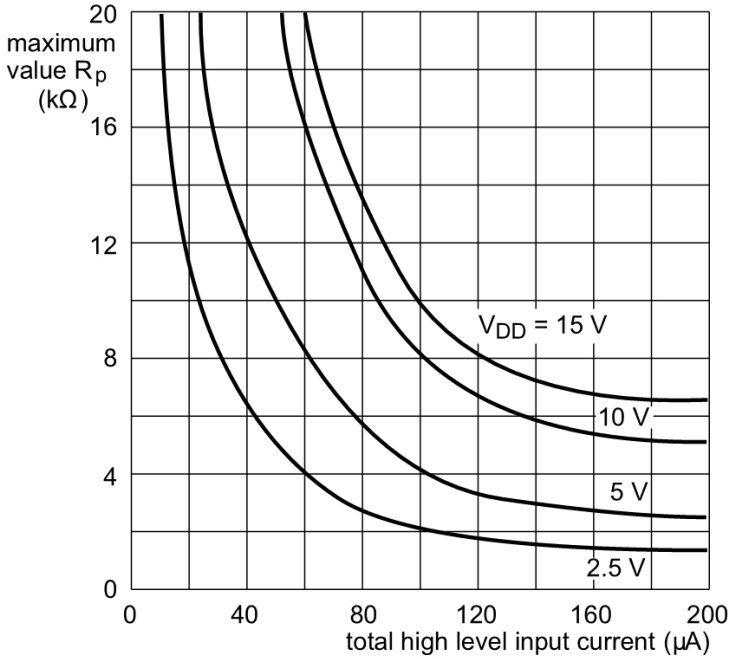


Figure 47. Total HIGH-level input current as a function of the maximum value of R_p with supply voltage as a parameter

7.5 Wiring pattern of the bus lines

In general, the wiring must be chosen so that crosstalk and interference to/from the bus lines is minimized. The bus lines are most susceptible to crosstalk and interference at the HIGH level because of the relatively high impedance of the pull-up devices.

If the length of the bus lines on a PCB or ribbon cable exceeds 10 cm and includes the V_{DD} and V_{SS} lines, the wiring pattern should be:

SDA _____
V_{DD} _____
V_{SS} _____
SCL _____

If only the V_{SS} line is included, the wiring pattern should be:

SDA _____
V_{SS} _____
SCL _____

These wiring patterns also result in identical capacitive loads for the SDA and SCL lines. If a PCB with a V_{SS} and/or V_{DD} layer is used, the V_{SS} and V_{DD} lines can be omitted.

If the bus lines are twisted-pairs, each bus line must be twisted with a V_{SS} return. Alternatively, the SCL line can be twisted with a V_{SS} return, and the SDA line twisted with a V_{DD} return. In the latter case, capacitors must be used to decouple the V_{DD} line to the V_{SS} line at both ends of the twisted pairs.

If the bus lines are shielded (shield connected to V_{SS}), interference is minimized. However, the shielded cable must have low capacitive coupling between the SDA and SCL lines to minimize crosstalk.