AN11084

Very large I²C-bus systems and long buses

Rev. 1 — 14 October 2011 Application note

Document information

Info	Content
Keywords	I2C-bus, twisted pair cables, very long buses, communication cables, Cat5e, Cat6, Fast-mode, Fast-mode Plus, Fm+, inter-IC, SDA, SCL, P82B96, PCA9600, PCA9605, RS-485, CAN bus
Abstract	The availability of powerful I²C-bus buffers that drive their I/Os on both sides to a nominal ground or ‘zero offset’ logic level allows the removal of noise introduced into one section of a very large bus system. That ‘regeneration’ of clean I²C signals enables building very long I²C buses by combining together relatively short bus sections, each say less than 15 meters, using such buffers or multiplexers. Conventional twisted-pair communication cabling with its convenient connectors, and a ‘modular’ I²C-bus system approach, make very large I²C systems such as for solar panel array control or multi-storey car parks to indicate free spaces by overhead red/green indicators very simple to construct. Together with the clever PCA9674/A GPIOs with their >100 simple addresses this arrangement could provide competition for RS-485 or other bus systems that cannot equal the number of simply addressable nodes possible in the system as described in the application note. It can be thousands of nodes. The described bus system is ‘multi-drop’ and slave devices may be connected anywhere on the system branches. Each drop point or node can be individually selected for bidirectional data communication with the master just by using normal I²C software addressing and/or switch control even with over 100 nodes on a single branch. The simplicity and flexibility of this approach makes it attractive to consider as an alternative to other bus systems such as RS-485 or CAN bus. Because very large numbers of buffers, and the wiring, will introduce signal delays that can ‘skew’ the timing of the I²C-bus clock and data signals it becomes necessary to modify the timing margins generated by the conventional I²C-bus components used in small systems. This application note describes two timing delay modules that ensure all devices in a very large system will still receive compliant I²C-bus timing signals. A single ‘master’ delay module is used in combination with ‘slave’ delay modules fitted at each slave in the system. The delay timing is programmable, according to the size of the I²C-bus system, and a corresponding data bit rate for the system can be calculated. As a guide, a system with multiple branches, each over 300 meters long, will be capable of around 60 kbit/s. Details of the timing considerations for I²C buses and designing for the delays caused by buffers, cables, or opto-couplers can be found in AN11075, “Driving I²C-bus signals over twisted pair cables with PCA9605” (Ref. 1) at the NXP web site www.nxp.com/i2c or at www.nxp.com/interface.

Revision history

Rev	Date	Description
v.1	20111014	application note; initial release

Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

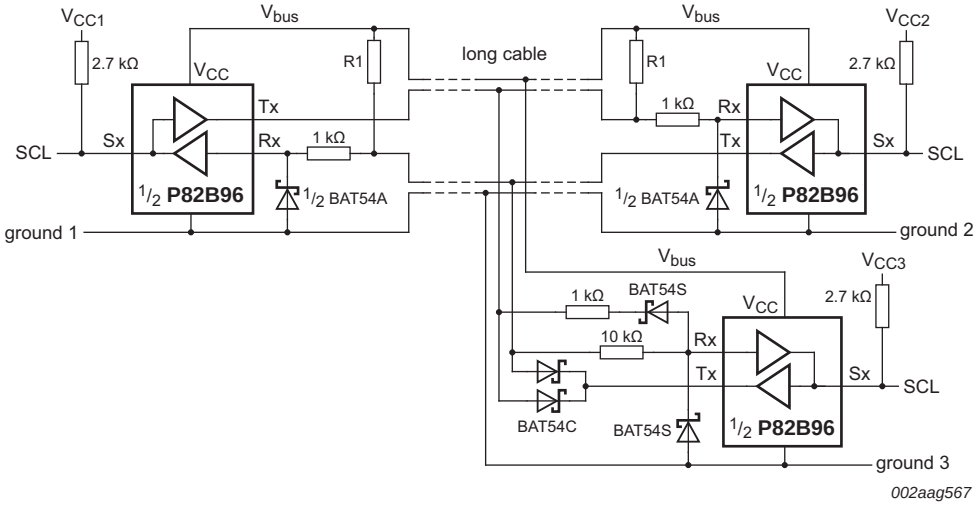
1. Introduction

Because the original I²C-bus applications were internal to a piece of equipment, for example in a PC or radio/TV/audio equipment, I²C-bus is rarely considered for systems when long distances with large numbers of drop-off points are required. It has been widely adopted 'outside the box', as the bus (DDC) used to control PC monitors or in HDMI links of entertainment equipment, but its potential for application over even longer distances is still generally not recognized.

The availability of bus multiplexing and buffering devices with high current drive capability, together with slave devices such as I/O expanders that allow direct addressing of over 100 different devices on a single bus branch, means it becomes feasible to consider I²C based monitoring and control systems with thousands of nodes and total bus wiring measured in kilometers/miles. Conventional communication cabling (e.g., Cat5 or Cat6) is well suited to I²C-bus signaling and its spare conductors can provide a convenient power supply for the system nodes.

Traditional designs for very long buses have used a higher bus voltage, 12 V or 15 V, to increase its tolerance to external interference. They retain bidirectional signaling on two signal wires. The higher voltage does improve noise voltage margins, and this arrangement has been successfully applied to buses up to at least 500 m, but the theoretical ground voltage difference between connected devices is limited to just 0.5 V.

Figure 1 shows a variation on the conventional arrangement. Each logic signal (SDA or SCL) has been split into two separate Send and Receive components as described in the '4-signal' arrangement discussed in application note *AN10658, "Sending I²C-bus signals via long communications cables"* (Ref. 2). This allows a theoretical ground potential difference nearly half of V_{bus}. Typical communication cables have four twisted-pairs, allowing a separate pair for each logic signal, but more care is needed if power is to be distributed on the 'grounded' wires of the twisted pairs. Full multi-master operation is retained but any noise induced anywhere in the bus wiring will appear at the Tx/Rx side of every connected buffer.



Only SCL is shown; arrangement is identical for SDA. Total eight cores of the cable are used.

Fig 1. 4-signal, multidrop, I²C uses a single long bus operating at higher voltage

An alternative arrangement, made possible by the release of ‘zero offset’ bus buffers, breaks the long bus into shorter segments making each no longer than, say, 5 m to 15 m. Buffers are then fitted between each relatively short cable section to remove any noise that has entered that short section. The whole bus system is operated at 5 V using lower resistance bus pull-ups, closer to the cable’s characteristic impedance, to ensure faster, cleaner bus transitions through the important logic switching thresholds.

The lower voltage has less tolerance to induced noise, but its shorter segments can provide very reliable operation. The buffers ‘regenerate’ clean, noise free, logic signals to drive each adjacent length of cable enabling a total bus length limited only by the bus speed necessary to accommodate the signal propagation delays or, possibly, provision of power to each buffered node and associated I/O devices.

Figure 2 shows how a long bus can be split into shorter, buffered, segments. Slaves can be connected on either side of any buffer or at intermediate points on each of the short cables. Figure 3 shows example waveforms at a buffer’s input and output. Provided the peak level of the input noise does not exceed the V_{IH} threshold, about 1.4 V, the output remains noise-free.

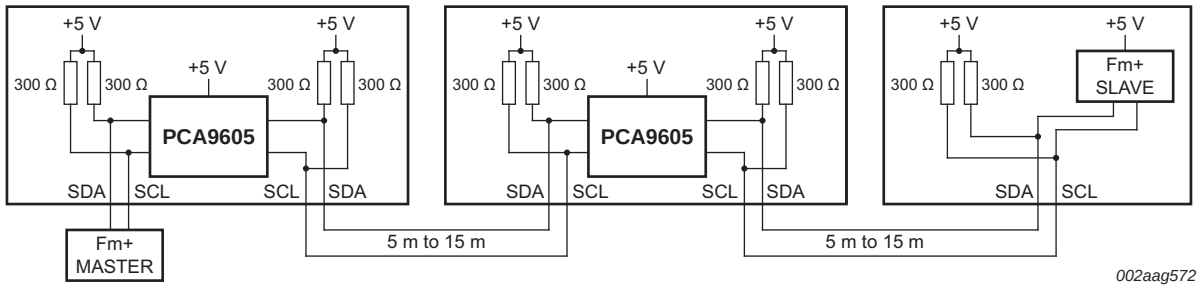
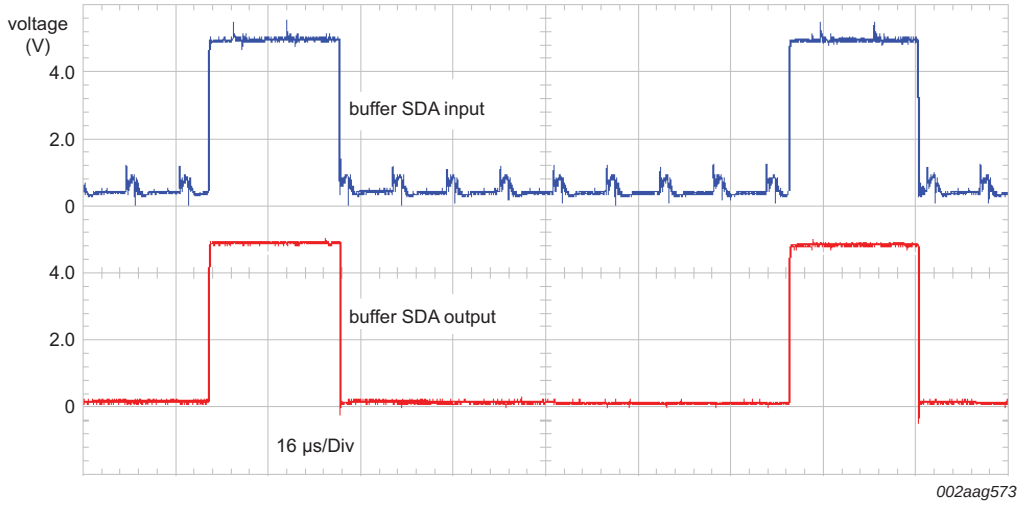


Fig 2. ‘Regenerating’ clean logic signals every 5 m to 15 m removes restrictions on total length



A DC offset and transformer-coupled noise were added to the input cable’s ground wire.

Fig 3. Noise on a buffer input (top) is removed and its output is clean

This arrangement uses just two of the cable pairs leaving the remaining two available for more convenient power distribution, for example at higher voltage such as the 48 V as used in Power over the Ethernet systems. At convenient places the 48 V is converted to 5 V that is then carried (short distances) using the I²C-bus signal pairs.

A disadvantage of this arrangement is that the slew rate control on the buffer's driven edges (i.e., falling edges) introduces a signal delay on falling edges while the buffer cannot influence the bus rising edges. In large systems the falling edge delays can accumulate until they are large enough to cause illegal bus signal timings.

This application note provides a 'hardware' solution that delays the bus signals to compensate the buffer's delays. While a 'software' solution could be equally effective at the Master, it would require the system designer to have control over the precise I²C-bus timing of, for example, a START condition or the t_{VD} timing. That control is possible if 'bit-banging' the I²C-bus signals, but mostly not in integrated I²C interfaces. With the possible exception of microcontrollers, most I²C slaves will require the hardware solution described.

2. Managing the delays in large systems

The delays introduced at the Master and Slave need to be adjusted to avoid the following possible conditions:

1. At the START, the Master drives SDA LOW while SCL is HIGH and then, after a required set-up time, drives SCL LOW. If the Master is Fast-mode Plus compatible then the set-up time generated could be just 260 ns. In a system with, say, 100 series buffers it is possible that there will be some small 'skew' between the delays through the SDA and the SCL buffers. A skew of just 3 ns per buffer could mean that the Master SCL falling edge reaches a distant Slave before the SDA falling edge and it will not receive the START condition. If a hardware delay of around 10 ns per series buffer is added to the SCL falling edge at the Master, relative to the Master SDA falling edge during the START, that will ensure, after allowing for 10 ns skew/buffer, a correct START is delivered to even the most remote Slave. (As will be shown, the actual delays need to be slightly different because the hardware adds its own minimum delays).
2. When the Master addresses a Slave it is allowed to provide the data bits on SDA with minimal set-up or hold time relative to the falling edges of its SCL. If the Master is changing a data bit then it can do that at essentially the same time as its SCL falls. This is especially the case if the bus is bit-bashed and anyway for Fast-mode Plus (Fm+) this delay may be as short as 120 ns.

   If the Master data bit is changing from a HIGH to a LOW then the situation is very similar to a START. Both Master SCL and SDA are driven LOW at nearly the same time and 'skew' might cause their intended relative timing to reverse after passing through many buffers. To ensure the correct timing of data falling edges the master hardware will delay all SDA falling edges by around 10 ns per series buffer, relative to the already delayed Master's SCL falling edges.

   When the Master's transmitted data bit is changing from a LOW to a HIGH then it can release the SDA line at essentially the same time as it drives its SCL LOW. The slew on the SCL falling edge means the SCL will be delayed by 70 ns/buffer more than the

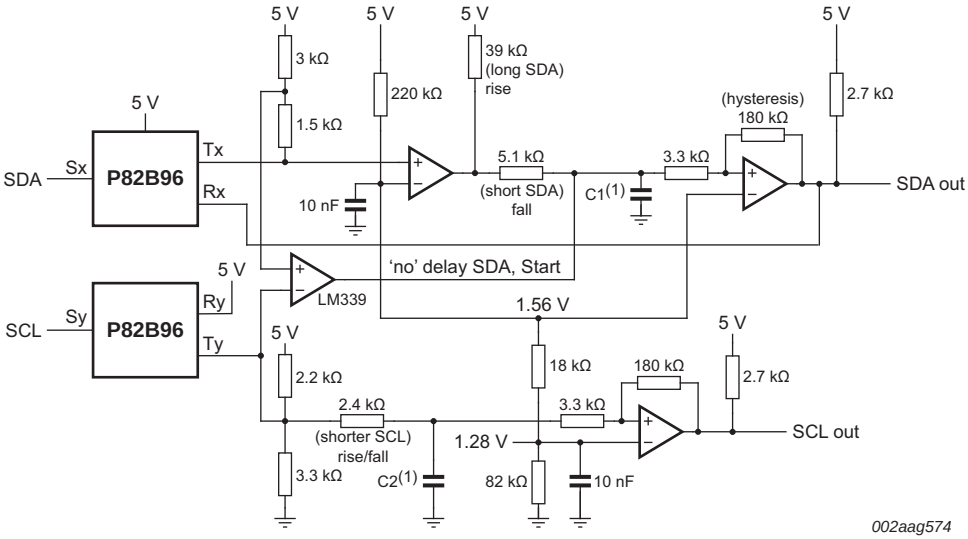
SDA rising edges. After traveling through 100 buffers the SDA rising edge would reach a distant Slave 7 µs before the Master's SCL falling edge. The consequent rising of SDA, while SCL is still HIGH, represents a STOP condition for that Slave.

In this example the hardware at the Master must therefore delay the Master's SDA rising edges by more than 70 ns / series buffer (in addition to the 10 ns delay of the Master's SCL falling edges) to ensure its SCL falling edges reach every Slave before the rise of its SDA signal. For 100 series buffers that delay becomes 8 µs and it becomes clear this can be a significant factor limiting the possible bus speed.

3. In a large system, a Slave located close to the Master will generate data bits on its bus section with closely the same timing as the Master would generate. Its data edges will be delayed only by the Slave's t_{VD} response delay relative to the received Master's SCL falling edges. For Fm+ parts that is typically only 200 ns. It is therefore necessary to delay the Slave's SDA signals in closely the same way as the Master's SDA above. The Slave SDA falling edges need to be delayed by > 10 ns, per series buffer in the system, and its rising edges need to be delayed 100 ns / series buffer.

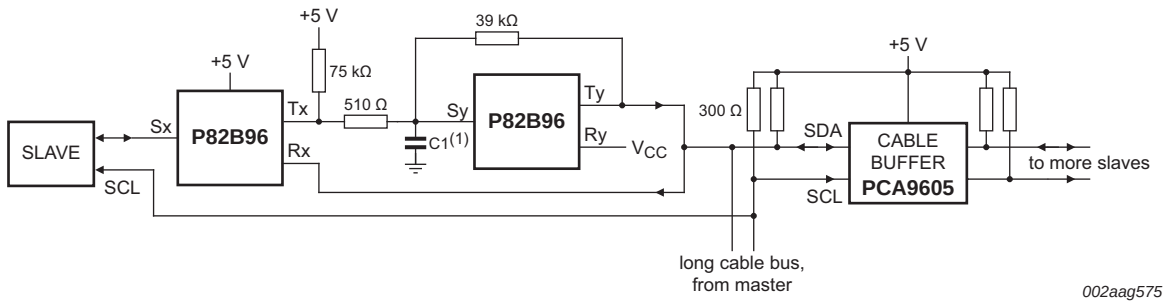
   Delaying the Slave's 'output' data bit timing, while allowing un-delayed input signals, requires the inclusion of a buffer with the capability to 'split' the bidirectional SDA signals into their unidirectional components. Possible buffers are P82B96 and PCA9600 and either may be used at the Master, but because only one half of a buffer is required at a Slave, only P82B96 should be used at Slaves because only its Sx/Sy characteristic will allow its remaining half to be used to generate the delays.

Figure 4 shows the hardware for generating the delays at the Master while Figure 5 shows the hardware for the Slaves. While no delay would actually be needed at the slaves that have the largest number of series buffers, i.e., are furthest from the Master, and mid-system Slaves could have smaller delays, the most practical system solution, for service and maintenance, will be to fit the same delay hardware at every Slave.



(1) C1 = C2 = 5.6 pF / buffer (**minimum** = 180 pF)

Fig 4. Schematic for SCL/SDA delay generation to be fitted at the Master

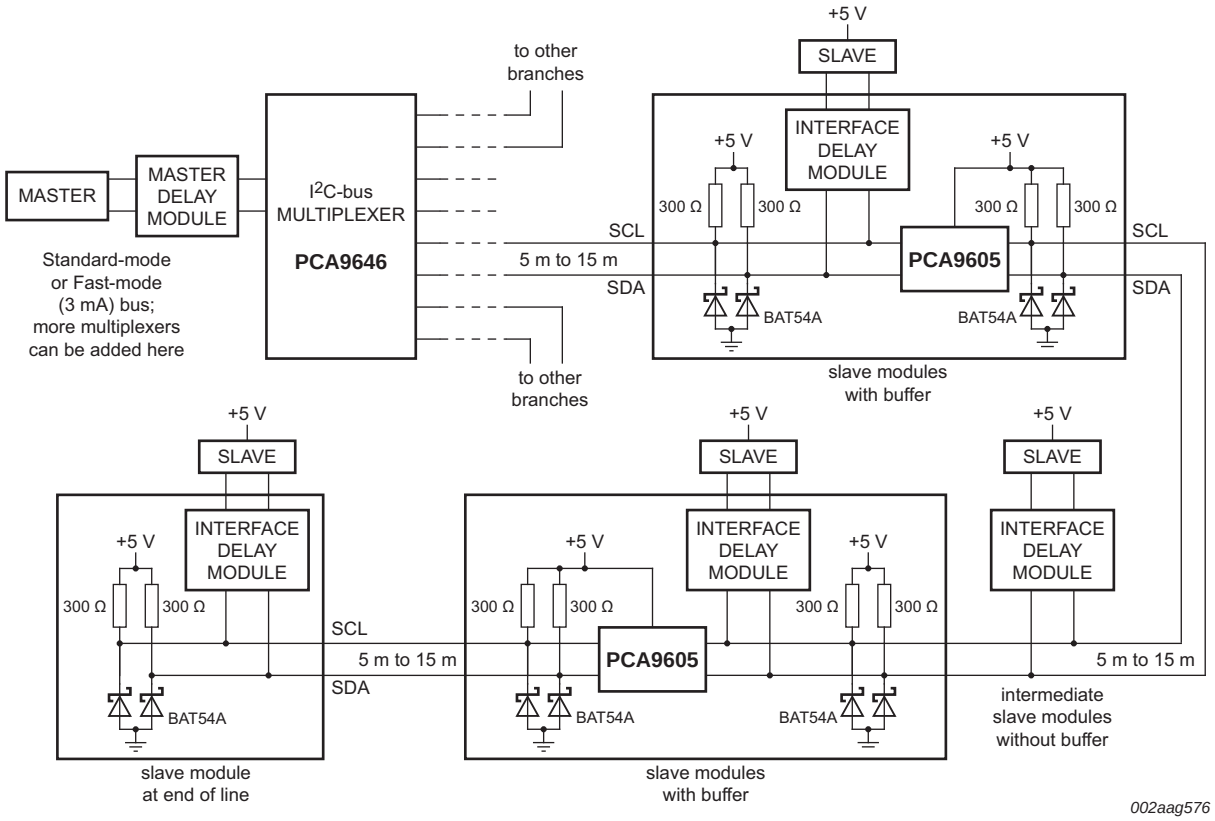


(1) C1 = 10 pF / cable buffer.

Fig 5. Schematic for SDA delay generation fitted at each Slave

3. Building practical systems

Figure 6 shows an example of a large system using a combination of multiplexers and buffers to build a very complex system with many branches in which each branch can be very long, maybe even 1 km, and have many connected slave devices. Slave devices with over 100 non-conflicting addresses make possible multidrop systems with more drops than are possible in systems such as RS-485. Where long cables are directly driven by the multiplexer it needs to include the zero offset buffers. PCA9646 is an example of a 4-output I²C addressable bus multiplexer including buffers. Where cables are not driven, for example between the master delay module and the PCA9646 in Figure 6, it is possible to include conventional Fast-mode addressable multiplexers, such as PCA9543C, that can use addresses that do not conflict with the PCA9646.



Where Slaves are fitted to a cable segment with spacing smaller than the allowed cable length between buffers (up to 15 meters), they can connect directly to the cable via the delay module.

Fig 6. Large arrays will include I²C-bus multiplexers

4. Compensating buffer and wiring delays - example system with 100 buffers on a branch

Because the delays of a single buffer and, say, 5 meters of cable are very small this example will be based on a system with 100 buffers in one branch and 5 meters of cable between each buffer — so this total bus segment length becomes 500 meters. The nominal delays caused by each PCA9605 buffer are less than 10 ns on rising edges and up to 90 ns on falling edges. In both cases those times are measured between the switching thresholds of the buffer (about 1 V) when it is driving cables up to 15 meters long with symmetrical 300 Ω pull-ups on both SDA and SCL. Because SCL is unidirectional, conventional treatment of the cable loading would be to place 150 Ω at the 'receiving' buffer because there is no point 'terminating' the cable in the reverse direction. Here the SCL pull-ups are made the same as the bidirectional SDA arrangement because it is important, as far as is possible, to preserve the symmetry of SDA and SCL buses to minimize timing 'skew' between their signals.

For Cat5 and similar communication cables the propagation delay, when driven and terminated as in Figure 6, will be closely 5 ns / meter. For 100 buffers with a total of 500 m cable the total delays that need to be taken into account are:

- **2.5 μs** cable propagation delay. It is closely controlled by the cable so has only a very small variation.
- The falling edge delay through the buffers. This will total around 7 μs typical, so for design purposes to include spreads, **9 μs** will be used.
- The rising edge delay through the buffers. This will total around 700 ns typical, but for design purposes **1 μs** will be used.
- The 'skew' or difference between SCL and SDA falling edge delays. This is very small, around 1 ns, but SDA slightly lags SCL and it can change with bus impedances, supply voltage, etc., so an allowance of 10 ns will be used. Total becomes **1 μs** for a system with 100 buffers.

All of these delays are proportional to the number of buffers and cables in any long bus segment.

To accommodate the above requirements, plus the minimum propagation delay of the low cost LM339 comparator used to generate them, the **Master** delay module is designed to produce the following delays:

- Minimum 1.5 μs delay of both rising and falling SCL edges. It may spread to 2 μs max.
- Minimum 10 μs delay of SDA rising edges. It may spread to 12 μs max.
- Minimum 3 μs delay of SDA falling edges when SCL is LOW. It may spread to 4 μs max.
- Minimum delay of SDA falling edges when SCL is HIGH for a START condition. (It is about 0.5 μs with quite low spread.)

The first three of these delays need to be proportional to the number of buffers, so the timing capacitor values in Figure 4 are selected according to the number of buffers in the segment being designed.

Because the comparators delay SDA by 0.5 µs during the START, the SCL must always be delayed slightly more than this to preserve the set-up time of the START. The corresponding minimum delay of SCL, independent of the number of buffers in the bus, requires C1/C2 to have absolute minimum values as well as values proportional to the number of buffers. During data transfers SDA changes are allowed only after SCL falling edges so SDA falling edges must have a minimum delay greater than that SCL delay. The minimum value for C1/C2 provides a minimum SCL delay greater than 0.5 µs and an SDA fall delay greater than1 µs.

To accommodate the buffer and cable delays, the Slave delay module is designed to produce the following delays:

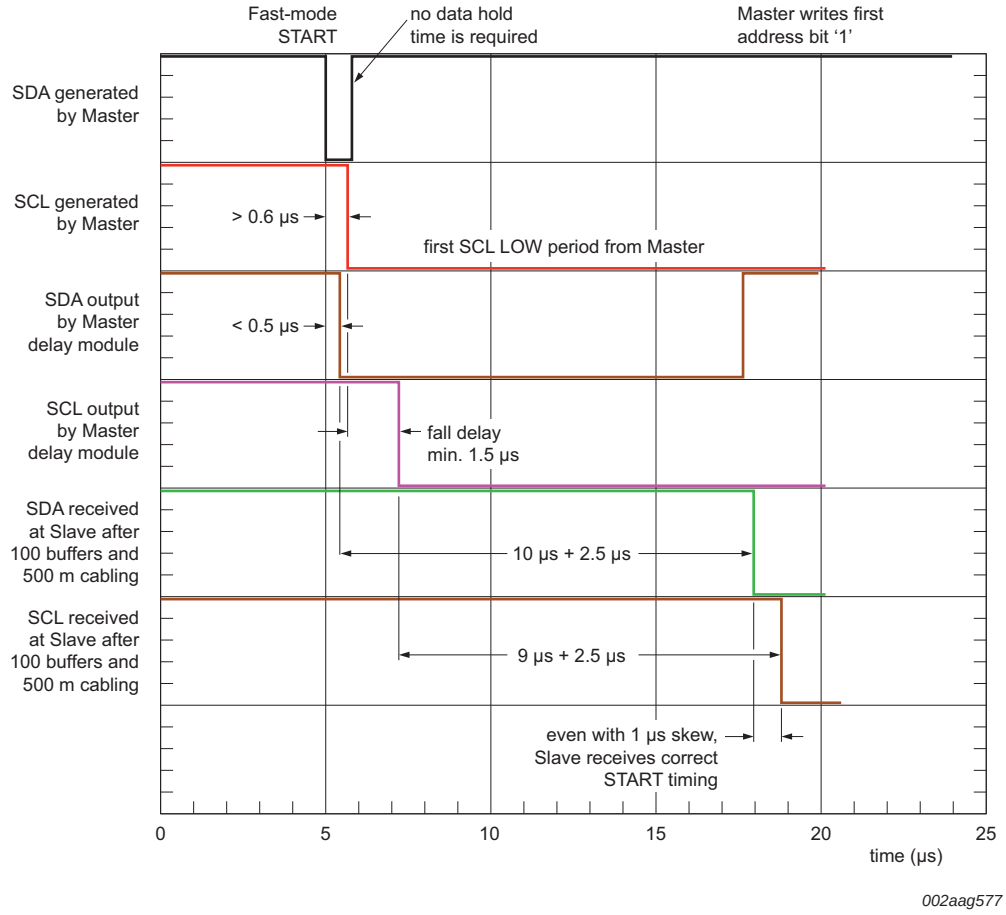
- Minimum 10 µs delay of SDA rising edges. It may spread to 15 µs max.
- Minimum 1 µs delay of SDA falling edges when SCL is LOW. It may spread to 2 µs max.

It does not delay the SDA signal coming from the cable to the Slave, except for the small propagation delay of the P82B96, Rx to Sx, that is around 200 ns.

Slave SDA delays are also proportional to the number of buffers, so the timing capacitor value in Figure 5 is selected according to the number of buffers in the segment being designed. The requirement is to ensure that Data delivered by the Slave closest to the Master will be received with its timing, relative to SCL, preserved when it reaches the most distant Slave on the bus segment being designed. For 100 buffers, the SCL falling edges are typically delayed by 7 µs, so the Slave module must delay SDA rising edges more than that. To allow for spreads and module tolerances the minimum delay is set at 10 µs, the maximum may be 15 µs. To allow for ‘skew’ on falling edges the SDA falling edges will be delayed by a minimum 1 µs with a maximum to 2 µs. It is not necessary to provide any absolute minimum value for these Slave delays.

Figure 7 shows the timing of an I2C-bus START generated by a Master that barely observes the Fast-mode timing requirements. The Master drives SDA LOW and, after the minimum set-up of 0.6 µs, drives SCL LOW to start the first clock cycle. Because the I2C-bus specification does not mandate any Data hold time (it only notes that some allowance be made to bridge the region of uncertainty during the maximum fall time of 300 ns), and because a bit-bashed implementation may actually make no allowance, the diagram shows essentially no hold time. (This is not intended to encourage driving this system with a Master that uses bus timings that do not conform to the official I2C-bus timing specifications. Most Masters use the SCL HIGH period as their set-up time for START because this time actually is the HIGH period of the first clock cycle.)

The first Master Address bit, a ‘1’, is provided on SDA at about the same time as SCL falls. The timing module at the Master delays the SCL edges by 1.5 µs to 2 µs. The minimum 1.5 µs allows the correct timing of the START to be preserved after allowing for the minimum delay of SDA (0.5 µs) and a ‘skew’ of 1 µs caused by 100 buffers and variations in the bus wiring. While this represents a possible combination of compliant bus timings, the more usual START condition will use the SCL bus HIGH period as the set-up time. The slow speeds used in very large systems would then not require any additional hardware delays, but most often such finer details about the Master’s timing will not be published, so this safer approach is needed.



The buffer/wiring delay for SDA has 1 µs added to the nominal value of 9 µs, as shown for SCL, to allow for possible timing 'skew'.

Fig 7. Timing diagram for a START condition

The timing of data delivery by the Master is more likely to have short data hold times and the illustrated handling of 'skew' then becomes applicable. Delivery of data by the Master will not be a factor limiting system speed. It will be the reading of data that determines the fastest clock timing. Further, because the delay modules must delay bus rising edges more than the buffers delay falling edges, the 'worst case' timing will apply when the Master is receiving a data '1' generated by the Slave in the system at the furthest end of the longest bus section.

Figure 8 shows the signal delay components involved when the Master reads a data '1' from a slave connected via 100 buffers and 500 meters of cable. The Master SCL is delayed by the Master delay module and then by the buffers and 500 meters of cable. After the Slave data response time, t_{VD}, the slave outputs its data '1' bit. That is delayed by the Slave delay module and then by the buffers and 500 meters of cable as it returns to the Master. It is required to be available at the Master before the end of the Master clock LOW period. In this example the LOW period would be required to be longer than 33.6 µs.

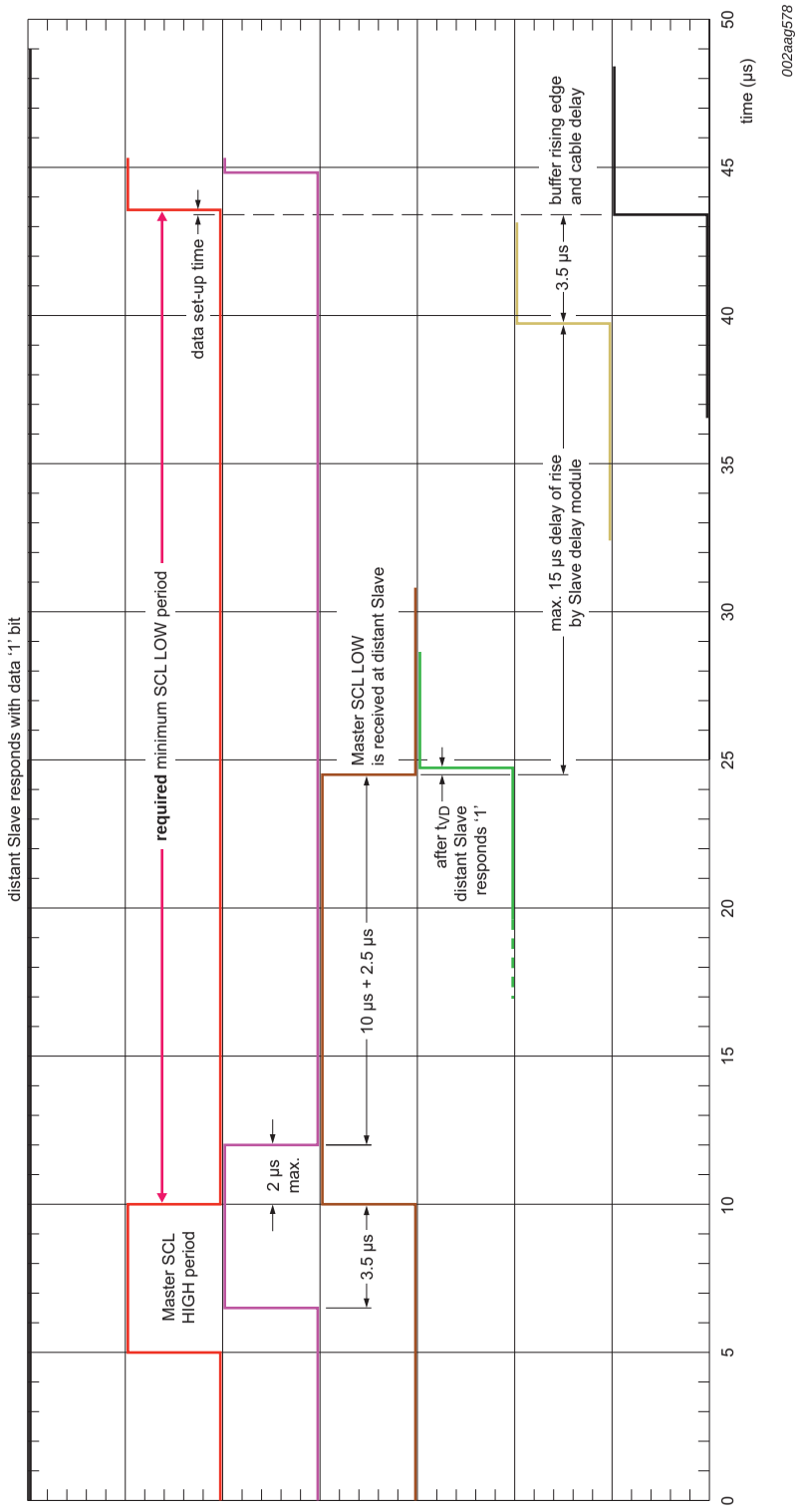


Fig 8. Timing diagram for the Master reading a data '1' bit from the most distant Slave in the system

Where ‘N’ represents the maximum number of buffers and ‘d’ represents the total cable length in meters in the longest branch of the bus, the LOW period is required to be longer than the sum of:

1. The maximum Master module’s SCL delay time = N × 20 ns. (With also an absolute minimum of 1 us.)
2. The buffer’s delay of the SCL falling edge = N × 100 ns.
3. The cable propagation delay of the Master’s SCL falling edge = d × 5 ns.
4. The t_{VD} response time of the slave. Worst case for an Fm+ slave is 450 ns.
5. The maximum Slave module’s delay of the SDA rise = N × 150 ns.
6. The buffer’s delay of the rising SDA edge = N × 10 ns.
7. The cable propagation delay of the Slave’s rising edge = d × 5 ns.
8. The required $t_{SU;DAT}$ of the Master. Example: 50 ns for Fm+ or 100 ns for Fm.

t_{LOW} = N × 280 ns + d × 10 ns + Slave t_{VD} response time + Master data set-up time. (For small numbers of buffers check the absolute minimum SCL delay time is met and used in this sum.)

The resulting LOW period represents the ‘effective’ LOW period as required by the Master. The P82B96 or PCA9600 at the Master will cause the SCL timing on the Master SCL pin to be ‘stretched’ by Master delay module’s delay. If the Master does not recognize that ‘stretch’ then its LOW period must be simply programmed with the result of the above calculation. If the Master device does recognize clock stretching, then the Master’s nominal (that is, programmed) LOW period may be shorter, by the amount of stretch.

The ‘stretch’ will be the nominal delay caused by the Master delay module, minimum 1 µs, typical 1.5 µs.

The Master SCL HIGH period must be long enough to be correctly processed by the Master delay module that must generate a minimum delay of 1 µs. The simple configuration of the delay timers does not make them ‘instantly re-settable’. If the programmed Master HIGH was, for example, just 1 µs then the capacitor in the delay timer will not recover the charge needed to achieve the nominal minimum 1 µs delay of the subsequent edge. The recommended absolute minimum SCL HIGH period is 3 µs and it should also be more than twice the Master module’s maximum SCL delay time (twice N × 20 ns).

t_{HIGH} = N × 40 ns (Absolute minimum 3 µs)

Examples of the resulting possible clock speed (kHz), based on parts with at least Fast-mode capability, and are shown in Table 1. It assumes the Master is programmed with unequal clock HIGH and LOW periods that are calculated as above.

Table 1. Possible bus speed (in kHz) for some longest I²C branch length in meters

Number of buffers used	Total cable length (m) in longest branch					
	40	80	160	320	640	1280
8	128	122	-	-	-	-
16	-	89	82	-	-	-
32	-	-	65	59	-	-
64	-	-	-	39	34	-
128	-	-	-	-	20	18

Note when counting the number of buffers, 'N', it must be the total number of buffers in the branch. Even when a branch is configured as in Figure 9, with the Master driving a mid-point of the branch and making the longest distance from Master to any Slave shorter, the count is still N = X + Y + 2.

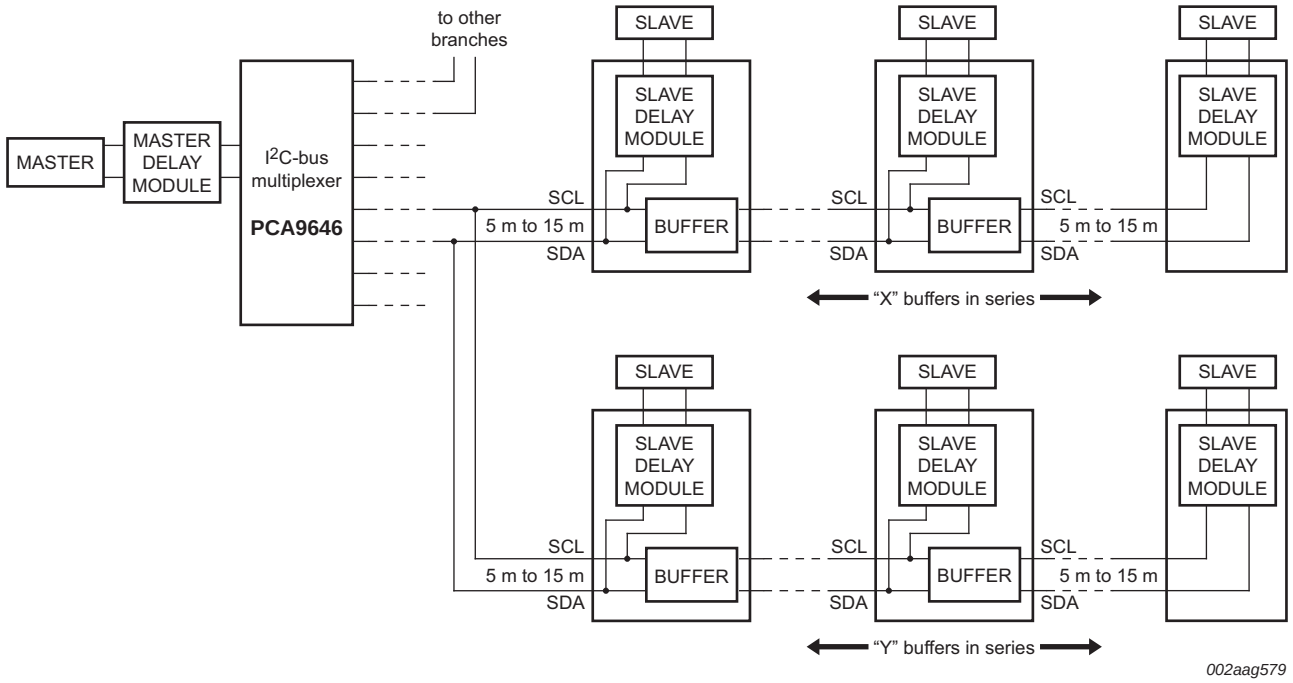


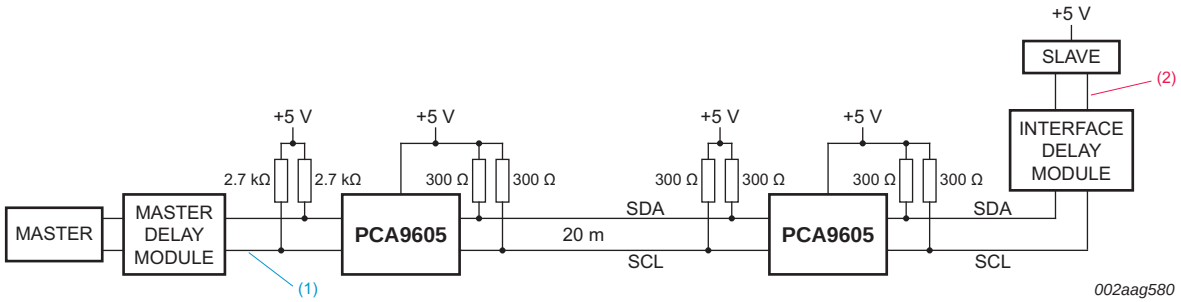
Fig 9. A branch driven at a mid-point and the way to total the buffers

The reason for counting this way is that the slave response from one end of this branch travels not only to the Master but continues to propagate to the Slaves on the other half of this branch. That response, after delays, must remain a valid I²C signal, relative to the Master clock, at every point in the system. If the count was made just (X + 1) or (Y + 1), then the calculated Master SCL period would be shorter and SCL would go HIGH at the end of the calculated LOW period while a Slave SDA response was still propagating from one Slave to the Slave most distant from it.

If the I²C-bus multiplexer can enable more than one of its I/Os at the same time, then this same rule applies – count the total number of buffers in the enabled channels. Generally it would be wise not to enable multiple multiplexer branches. If there are no address conflicts within the two branches that would be enabled together, then consider making them just a single branch instead.

While it was not practical to build such large systems in the laboratory, the following oscilloscope traces illustrate the basic timing principles involved. While only one short cable section is used, the delay modules have their timings adapted for a system that would have 100 buffers.

The waveforms in Figure 11 to Figure 14 were taken at the places shown in the schematic Figure 10.

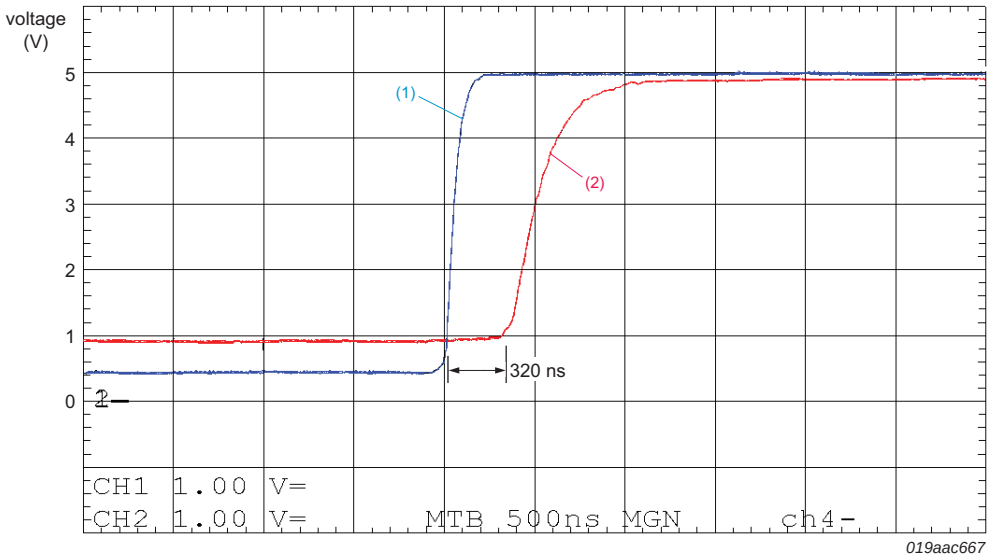


(1) Master delay module

(2) Slave

Fig 10. Schematic used for example waveforms

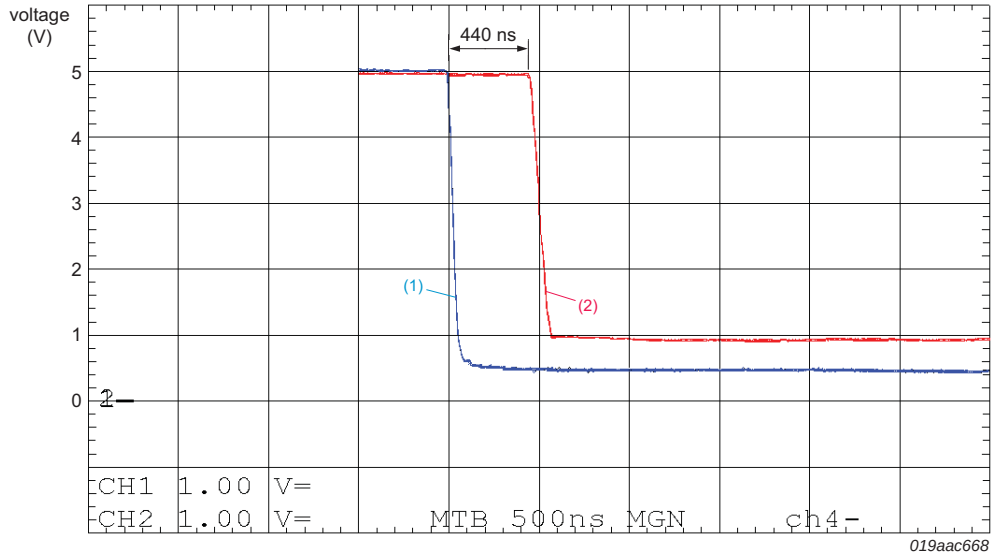
Figure 11 and Figure 12 show how falling bus edges are delayed by the slew control in PCA9605 by about 60 ns more than the rising bus edges. After two buffers, the falling edge has been delayed about 120 ns more than the rising edge in Figure 11. The timings used for system design increase those typical values to allow for spreads.



(1) Master delay module

(2) Slave

Fig 11. Master delay module releases SDA and HIGH propagates to the slave

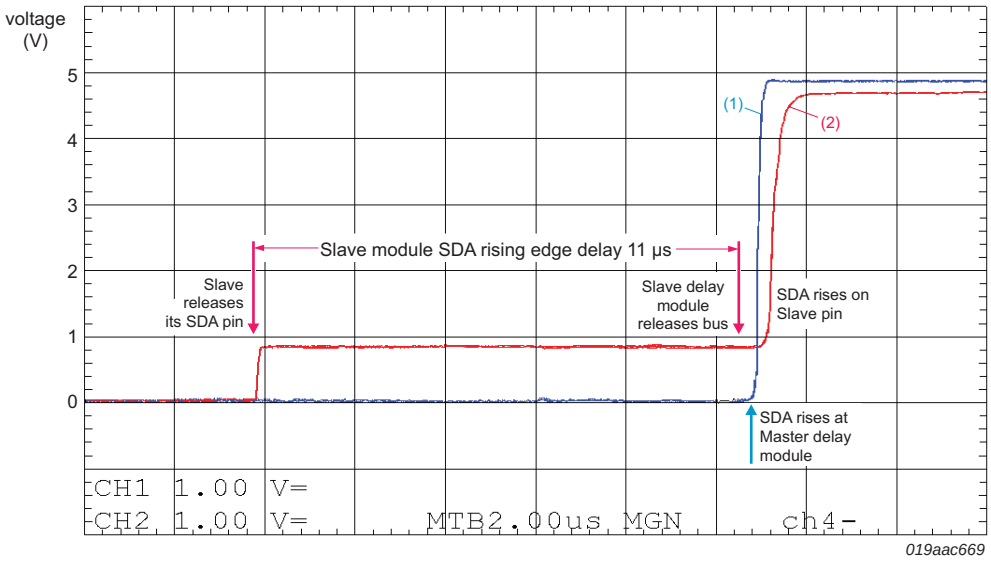


(1) Master delay module

(2) Slave

Fig 12. Master delay module drives SDA and LOW propagates to the slave

Figure 13 shows the Slave releasing its SDA pin, at the end of an Acknowledge or data bit, and its HIGH propagating to the Master. When the Slave releases the SDA it is clamped LOW by the Sx pin of the P82B96 in the Slave delay module until the end of the delay time, about 11 µs. After the delay module releases SDA there is a delay of less than 20 ns in the two buffers and 100 ns in the 20 m cable. The SDA pin of the Slave cannot rise until after the cable drive voltage goes HIGH and the Rx pin of the Slave delay module goes HIGH. Because the P82B96 rising edge delay, Rx to Sx, is about 200 ns the Slave's HIGH in this example actually reaches the Master delay module before the Slave's pin goes HIGH.

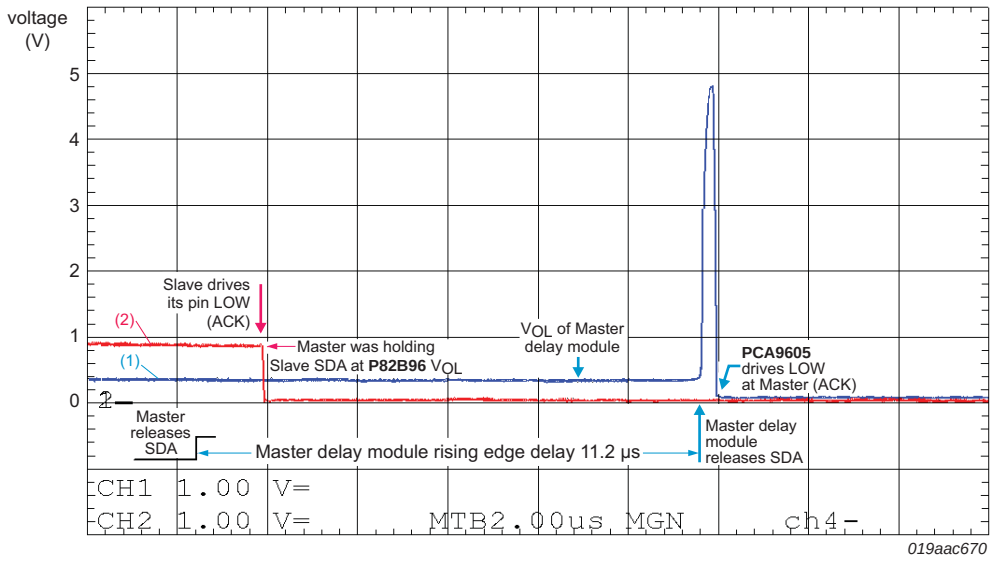


(1) Master delay module

(2) Slave

Fig 13. Slave releases SDA and HIGH propagates to Master delay module

Figure 14 shows the Slave driving an Acknowledge or data LOW. The Master had been holding SDA, at the Slave, at the special LOW level driven by the Sx of P82B96. Even though the Master releases its drive to the Master delay module at the time shown, its rise is delayed about 11 µs by the Master delay module. When the Master delay module releases SDA at the input to the PCA9605 buffer it must rise to the 'unlock' level of this buffer before the buffer can reverse its direction and allow the waiting LOW from the Slave to propagate to the Master.



(1) Master delay module

(2) Slave

Fig 14. Slave drives SDA LOW and LOW propagates to the Master

Figure 15 shows the rise/fall delays of the clock signal by the Master delay module. The Master delay module's SCL input is driven by the open-collector Ty output of PCA9600 and does not rise to the full 5 V. The level is chosen to allow discrimination between the SCL being 'HIGH' when SDA changes on a START or being 'LOW' during normal data level changes on SDA. The trace includes the rising edge delay (about 310 ns) and the falling edge delay (about 370 ns) through one buffer and 20 m cable. The nominal Master module's delay is between 1.5 µs and 2 µs.

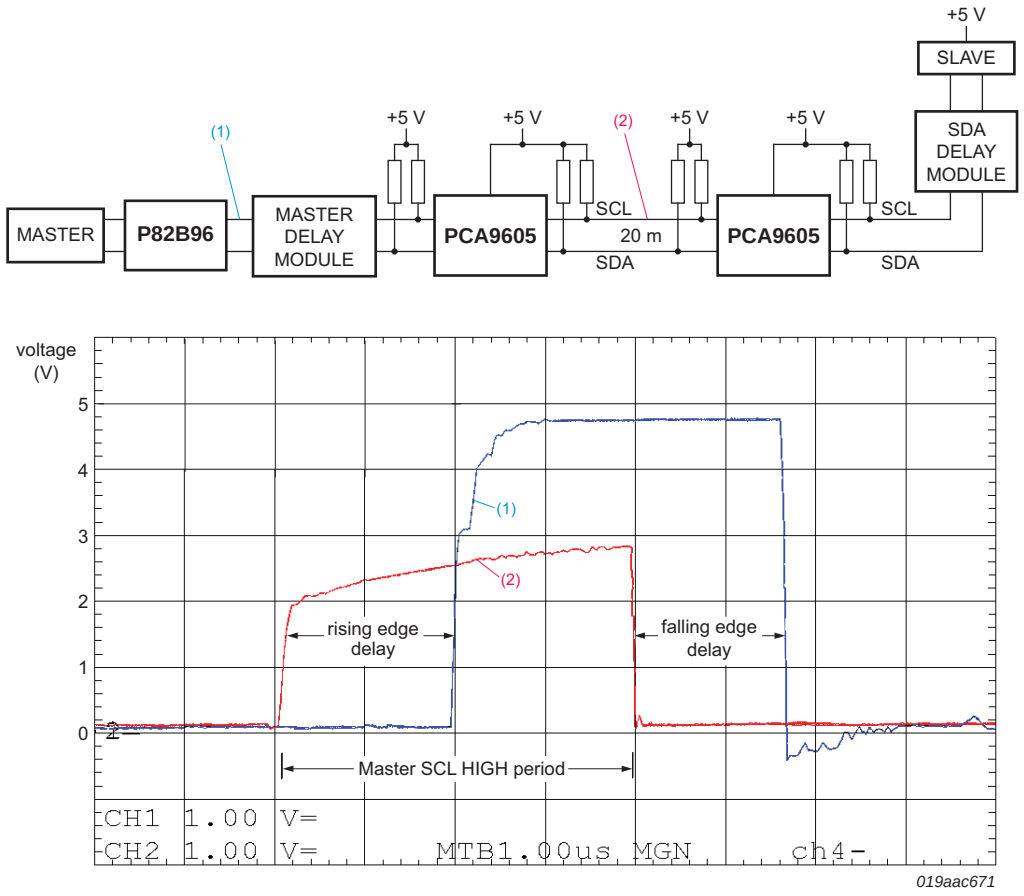


Fig 15. SCL delays through Master delay module (and a buffer)

Figure 16 shows the falling edge delay and Figure 17 shows the rising edge delay generated by the Slave SDA delay module.

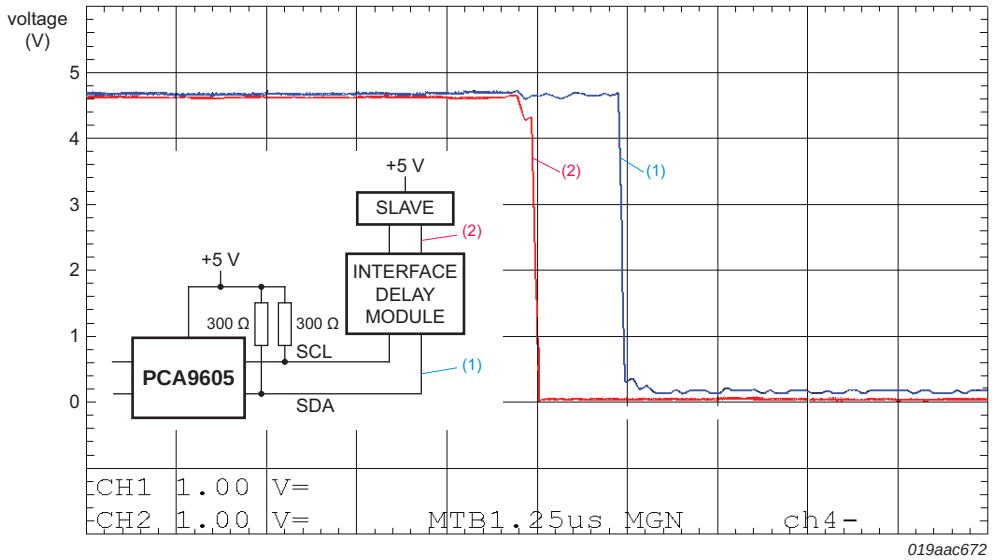


Fig 16. Falling edge delay of SDA through Slave delay module, 1 µs to 2 µs

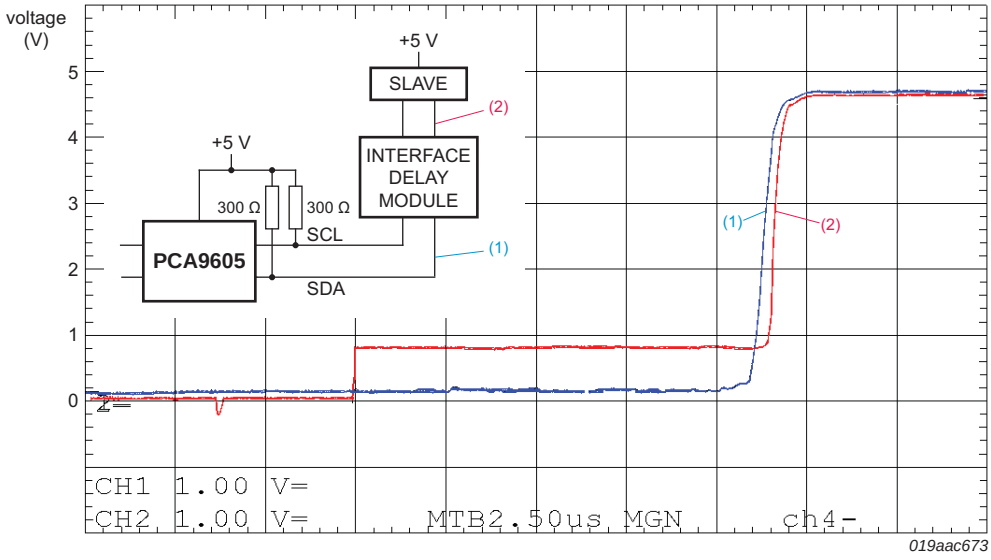


Fig 17. Rising edge delay of SDA through Slave delay module, approximately 11 µs

Figure 18 shows the Master delay module's delay of the SDA falling edge when SCL is LOW. When SCL is HIGH, for a START condition, the delay timer is not activated and the SDA delay is minimal. It is just the propagation delay of the comparator, around 500 ns.

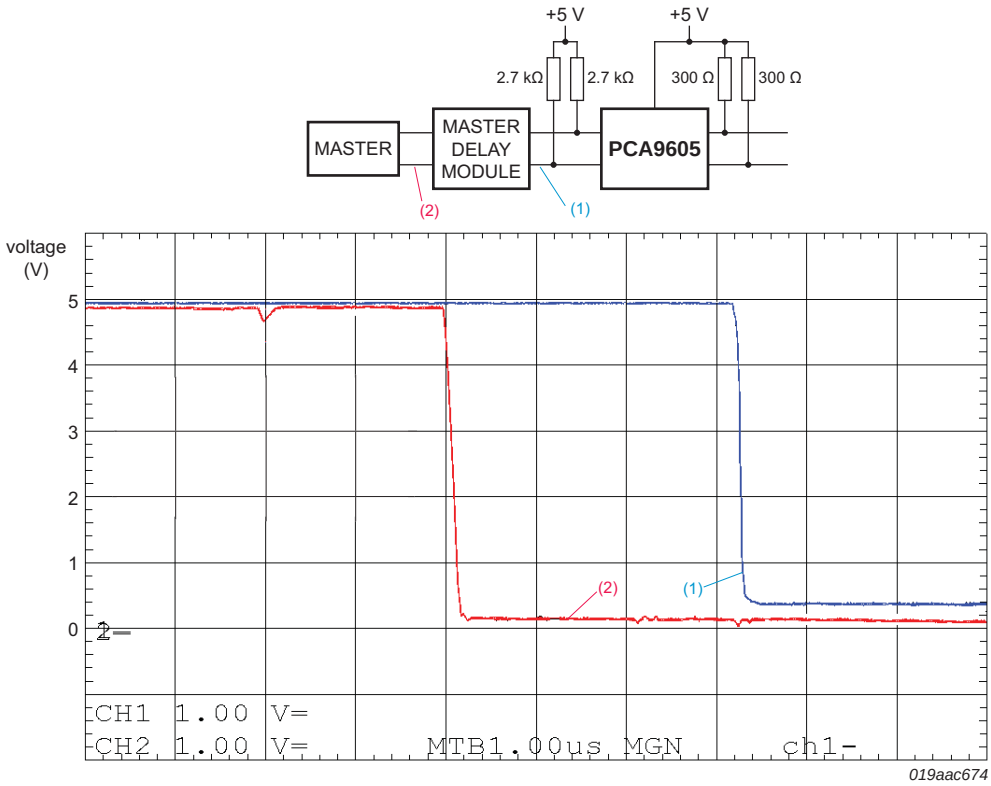


Fig 18. Falling edge delay of SDA through Master module, 3 µs to 4 µs

5. Conclusion

Provided steps are taken to ensure that the propagation delays of the SDA and SCL signals are correctly handled, especially any differences between their rising and falling edges, the application of 'zero offset' buffering of I²C signals enables the construction of very large I²C arrays using conventional twisted-pair communication cables and conventional 5 V signaling.

The availability of fully buffered, addressable, Fm+ multiplexers and Slave devices with over 100 non-conflicting 7-bit addresses makes this approach very attractive and likely to prove competitive with alternative bus arrangements.

When the large array will cover large outdoor areas its relatively low impedance bus structure and moderate data rate allows the use of relatively high capacitance but very rugged clamping devices, such as 1 A rectifier diodes with >40 A/10 ms pulse capability, for protection against the induced energy from powerful sources such as nearby lightning strikes. Of course such interference will result in data bit errors but when used with software designed to recover from those errors a very reliable system is possible.

6. Abbreviations

Table 2. Abbreviations

Acronym	Description
CAN	Controller Area Network
DDC	Data Display Channel
HDMI	High Definition Media Interface
I/O	Input/Output
I²C-bus	Inter-integrated Circuit bus
PC	Personal Computer

7. References

[1] AN11075, "Driving I²C-bus signals over twisted pair cables with PCA9605" — NXP Semiconductors; www.nxp.com/documents/application_note/AN11075.pdf

[2] AN10658, "Sending I²C-bus signals via long communications cables" — NXP Semiconductors, www.nxp.com/documents/application_note/AN10658.pdf

8. Legal information

8.1 Definitions

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

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

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

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

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

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

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

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

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

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

8.3 Trademarks

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

I²C-bus — logo is a trademark of NXP B.V.

9. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 14 October 2011
Document identifier: AN11084