



Electronic
components
and materials

PHILIPS

I²C bus specification

Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

Contents - I²C section

1.0 INTRODUCTION

2.0 THE I²C BUS CONCEPT

3.0 GENERAL CHARACTERISTICS

4.0 BIT TRANSFER

4.1 Data validity

4.2 Start and stop conditions

5.0 TRANSFERRING DATA

5.1 Byte format

5.2 Acknowledge

6.0 ARBITRATION AND CLOCK GENERATION

6.1 Synchronisation

6.2 Arbitration

6.3 Use of clock synchronising mechanism

7.0 FORMATS

8.0 ADDRESSING

8.1 Definition of bits in the first byte

8.1.1 General call address

8.1.2 Start byte

8.1.3 CBUS compatibility

9.0 ELECTRICAL SPECIFICATIONS OF INPUTS AND OUTPUTS OF I²C DEVICES

10.0 TIMING

11.0 'LOW SPEED' MODE

11.1 Start and stop conditions

11.2 Data format and timing

APPENDIX A

Maximum and minimum values of the pull-up resistors R_p and the series resistor R_s .

1.0 INTRODUCTION

For 8-bit applications, such as those requiring single-chip microcomputers, certain design criteria can be established:

- A complete system usually consists of at least one microcomputer and other peripheral devices - such as memories and I/O expanders.
- The cost of connecting the various devices within the system must be kept to a minimum.
- Such a system usually performs a control function and does not require high-speed data transfer.
- Overall efficiency depends on the devices chosen and the interconnecting bus structure.

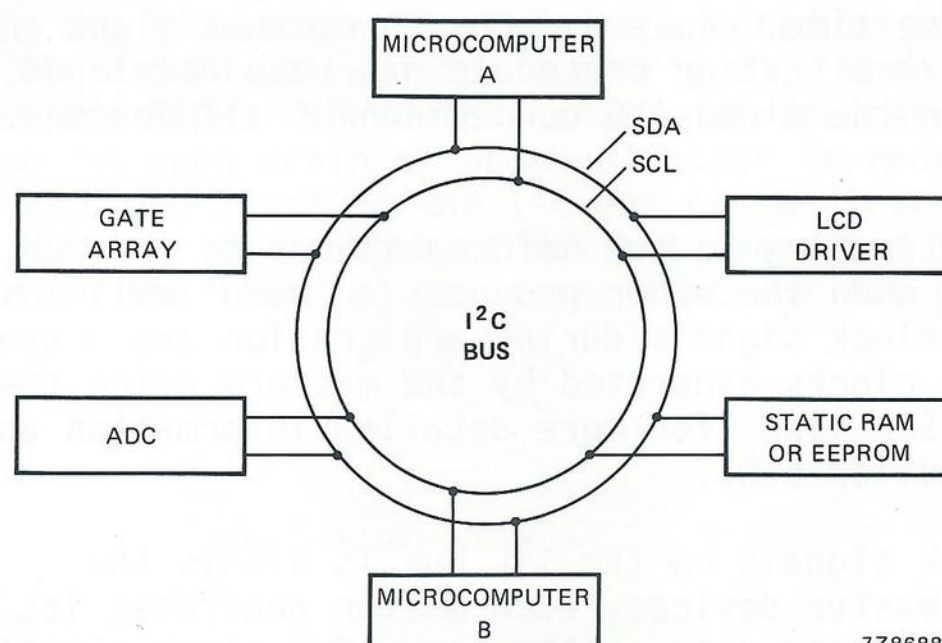
In order to produce a system to satisfy these criteria, a serial bus structure is needed. Although serial buses don't have the throughput capability of parallel buses, they do require less wiring and fewer connecting pins. However, a bus is not merely an interconnecting wire, it embodies all the formats and procedures for communication within the system.

Devices communicating with each other on a serial bus must have some form of protocol which avoids all possibilities of confusion, data loss and blockage of information. Fast devices must be able to communicate with slow devices. The system must not be dependent on the devices connected to it, otherwise modifications or improvements would be impossible. A procedure has also to be resolved to decide which device will be in control of the bus and when. And if different devices with different clock speeds are connected to the bus - the bus clock source has to be defined.

All these criteria are involved in the specification of the I²C bus.

2.0 THE I²C BUS CONCEPT

Any manufacturing process (NMOS, CMOS, I²L) can be supported by the I²C bus. Two wires (SDA - serial data, SCL - serial clock) carry information between the devices connected to the bus. Each device is recognised by a unique address - whether it is a microcomputer, LCD driver, memory or keyboard interface - and can operate as either a transmitter or receiver, depending on the function of the device we're considering. Obviously an LCD driver is only a receiver, while a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers (see table 1). A master 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 slave.



7Z86885

Fig.1 Typical I²C bus configuration.

The I²C bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it. As masters are usually microcomputers, let's consider the case of a data transfer between two microcomputers connected to the I²C bus (Fig.1). This highlights the master-slave and receiver-transmitter relationships to be found on the I²C bus. It should be noted that these relationships are not permanent, but only depend on the direction of data transfer at that time. The transfer of data would follow in this way:

- 1) Suppose microcomputer A wants to send information to microcomputer B
 - microcomputer A (master) addresses microcomputer B (slave)
 - microcomputer A (master transmitter) sends data to microcomputer B (slave receiver)
 - microcomputer A terminates the transfer.
- 2) If microcomputer A wants to receive information from microcomputer B
 - microcomputer A (master) addresses microcomputer B (slave)
 - microcomputer A (master receiver) receives data from microcomputer B (slave transmitter)
 - microcomputer A terminates the transfer.

Even in this case, the master (microcomputer A) generates the timing and terminates the transfer.



The possibility of more than one microcomputer being connected to the I²C bus means that more than one master 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 devices to the I²C bus.

If two or more masters try to put information on to the bus, the first to produce a 'one' when the other produces a 'zero' will lose the arbitration. The clock signals during arbitration are a synchronised combination of the clocks generated by the masters using the wired-AND connection to the SCL line (for more detailed information concerning arbitration see section 6.0).

Generation of clock signals on the I²C bus is always the responsibility of master devices; each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow slave device holding down the clock line or by another master when arbitration takes place.

Table 1 Definition of I²C bus terminology

Transmitter	- The device which sends data to the bus
Receiver	- The device which receives data from the bus
Master	- The device which initiates a transfer, generates clock signals and terminates a transfer
Slave	- The device addressed by a master
Multi-master	- More than one master can attempt to control the bus at the same time without corrupting the message
Arbitration	- Procedure to ensure that if more than one master simultaneously tries to control the bus, only one is allowed to do so and the message is not corrupted
Synchronisation	- Procedure to synchronise the clock signals of two or more devices

3.0 GENERAL CHARACTERISTICS

Both SDA and SCL are bidirectional lines, connected to a positive supply voltage via a pull-up resistor (see Fig.2). 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 in order to perform the wired-AND function. Data on the I²C bus can be transferred at a rate up to 100 kbit/s. The number of devices connected to the bus is solely dependent on the limiting bus capacitance of 400 pF.

4.0 BIT TRANSFER

Due to the variety of different technology devices (CMOS, NMOS, I²L) which can be connected to the I²C bus, the levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the appropriate level of V_{DD} (see section 9.0 for Electrical specifications). One clock pulse is generated for each data bit transferred.

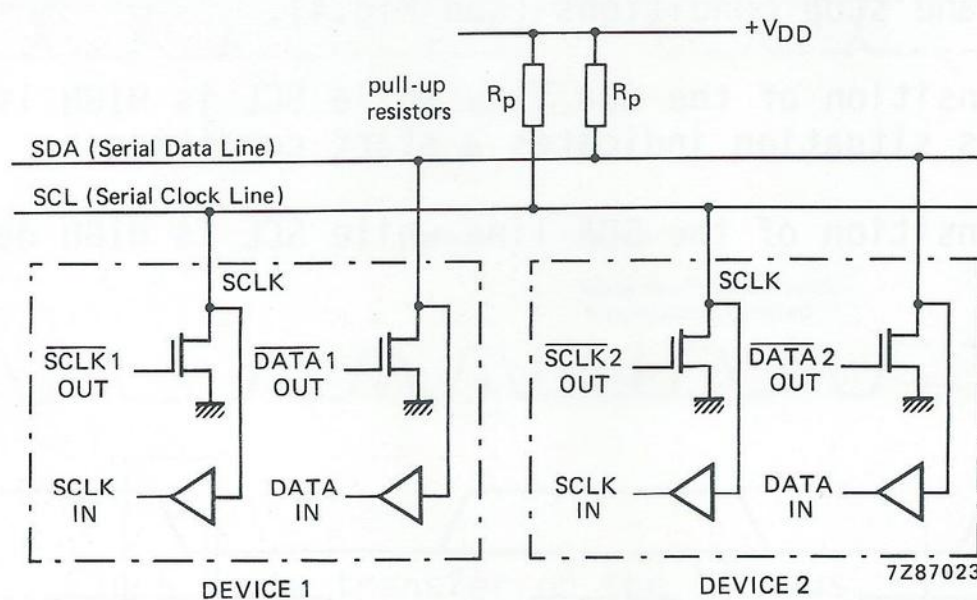


Fig.2 Connection of devices to the I²C bus.

4.1 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 (Fig.3).

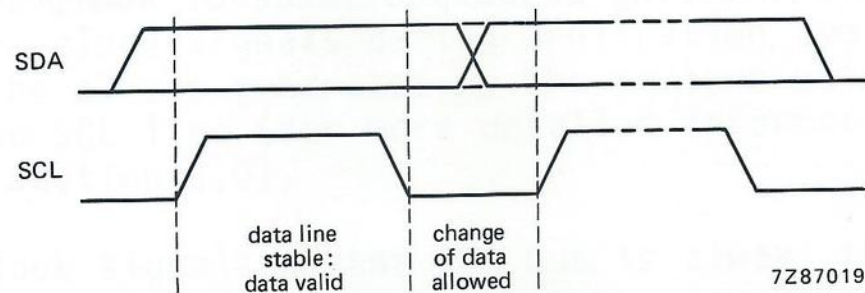


Fig.3 Bit transfer on the I²C bus.

4.2 Start and stop conditions

Within the procedure of the I²C bus, unique situations arise which are defined as start and stop conditions (see Fig.4).

A HIGH to LOW transition of the SDA line while SCL is HIGH is one such unique case - this situation indicates a start condition.

A LOW to HIGH transition of the SDA line while SCL is HIGH defines a stop condition.

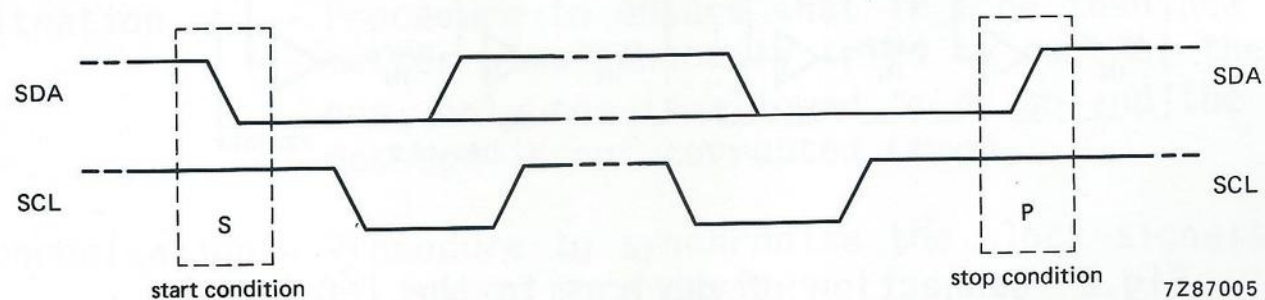


Fig.4 Start and stop conditions.

Start and stop conditions are always generated by the master. 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 will be described later in detail (in section 10.0).

Detection of start and stop conditions by devices connected to the bus is easy if they possess the necessary interfacing hardware. However, microcomputers with no such interface have to sample the SDA line at least twice per clock period in order to sense the transition.

5.0 TRANSFERRING DATA

5.1 Byte format

Every byte put on the SDA line must be 8-bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first (Fig.5). If a receiving device cannot receive another complete byte of data until it has performed some other function, for example, to service an internal interrupt, it can hold the clock line SCL LOW to force the transmitter into a wait state. Data transfer then continues when the receiver is ready for another byte of data and releases the clock line SCL.

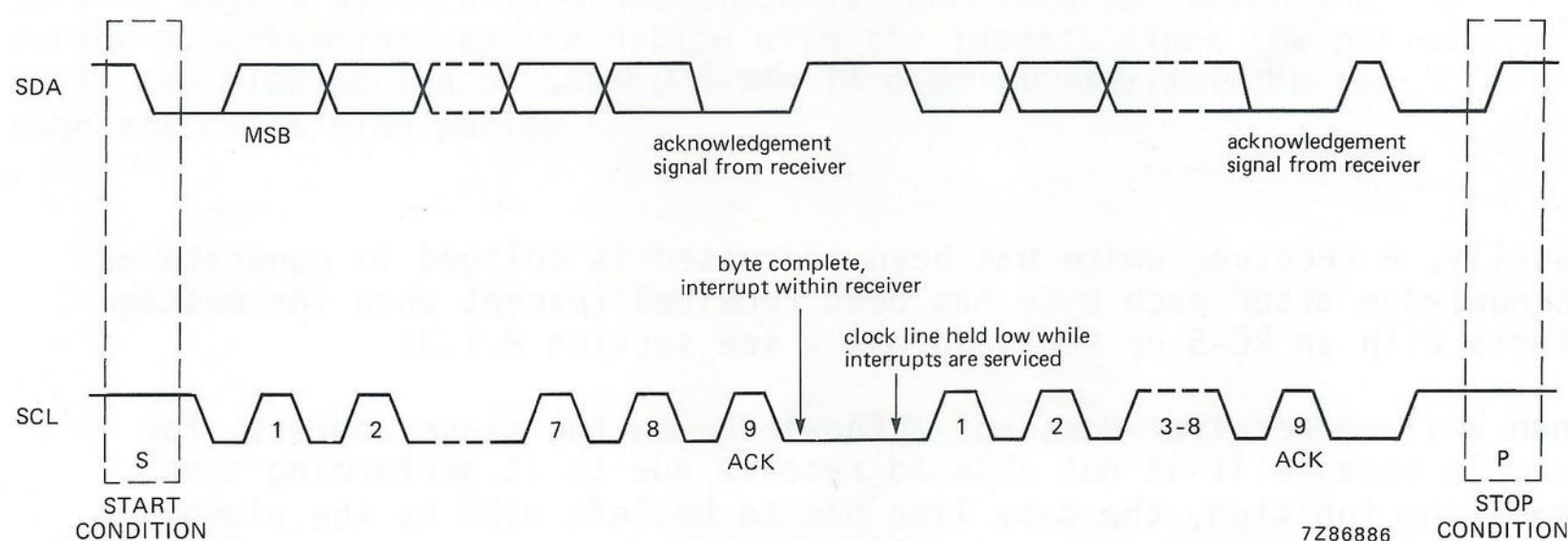


Fig.5 Data transfer on the I²C bus.

In some cases, it is permitted to use a different format from the I²C bus format (for example for CBUS compatible devices). A message which starts with such an address can be terminated by the generation of a stop condition, even during the transmission of a byte. In this case, no acknowledge is generated (see section 8.4).

5.2 Acknowledge

Data transfer with acknowledge is obligatory. The acknowledge-related clock pulse is generated by the master. The transmitting device releases the SDA line (HIGH) during the acknowledge clock pulse.

The receiving device has to pull down the SDA line during the acknowledge clock pulse so that the SDA line is stable LOW during the high period of this clock pulse (Fig.6). Of course, set-up and hold times must also be taken into account and these will be described in section 10.0.

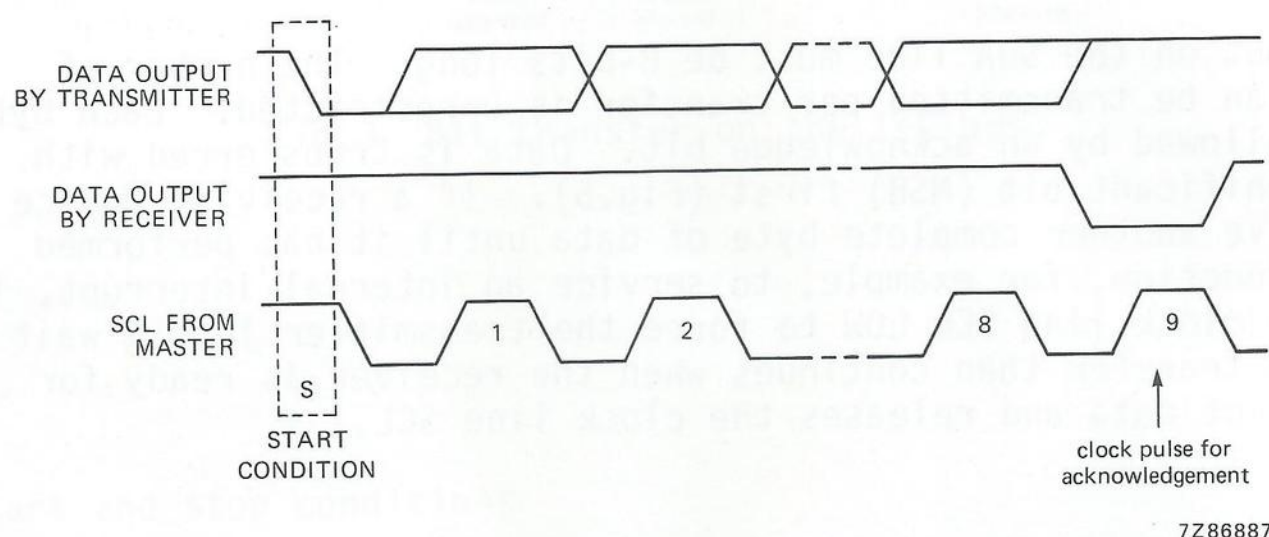


Fig.6 Acknowledge on the I²C bus.

Usually, a receiver which has been addressed is obliged to generate an acknowledge after each byte has been received (except when the message starts with an RC-5 or CBUS address - see section 8.1.3)

When a slave receiver does not acknowledge on the slave address, for example because it is not able to receive due to it performing some real-time function, the data line has to be left HIGH by the slave. The master can then generate a STOP condition to abort the transfer.

If a slave receiver does acknowledge the slave address, but some time later in the transfer cannot receive any more data bytes, the master must again abort the transfer. This is indicated by the slave not generating the acknowledge on the first byte following. The slave leaves the data line HIGH and the master generates the STOP condition.

In the case of a master receiver involved in a transfer - it must signal an end of data to the slave transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave transmitter must release the data line to allow the master to generate the STOP condition.

6.0 ARBITRATION AND CLOCK GENERATION

6.1 Synchronisation

All masters generate their own clock to the SCL line to transfer messages on the I²C bus. Data is only valid during the clock HIGH period on the SCL line. A defined clock is needed therefore, if the bit-by-bit arbitration procedure is to take place.

Clock synchronisation is performed using the wired-AND connection of devices to the SCL LINE. This means that a HIGH to LOW transition on the SCL line will affect the devices concerned, such that they start counting off their LOW period - and once a device clock has gone LOW it will hold the SCL line in that state until the clock HIGH state is reached (Fig.7). However, the LOW to HIGH change in this device clock may not change the state of the SCL line, if another device clock is still within its LOW period. Therefore SCL will be held LOW by the device with the longest LOW period. Devices with shorter LOW periods enter a HIGH wait state during this time.

When all devices concerned have counted off their LOW period, the clock line will be released and go HIGH. There will then be no difference between the device clocks and the state of the SCL line and all of them will start counting their HIGH periods. The first device to complete its HIGH period will again pull the SCL line LOW.

In this way, a synchronised SCL clock is generated for which the LOW period is determined by the device with the longest clock LOW period while the HIGH period on SCL is determined by the device with the shortest clock HIGH period.

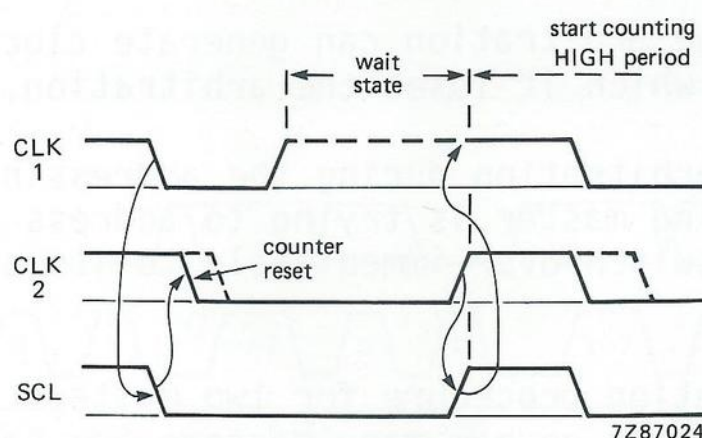


Fig.7 Clock synchronisation during the arbitration procedure.

6.2 Arbitration

Arbitration takes place on the SDA line in such a way that the master which transmits a HIGH level, while another master transmits a LOW level, will switch off its DATA output stage since the level on the bus does not correspond to its own level.

Arbitration can carry on through many bits. The first stage of arbitration is the comparison of the address bits (information on addressing can be found in section 8.0). If the masters are each trying to address the same device - arbitration continues into a comparison of the data. Because address and data information is used on the I²C bus for the arbitration, no information is lost during this process.

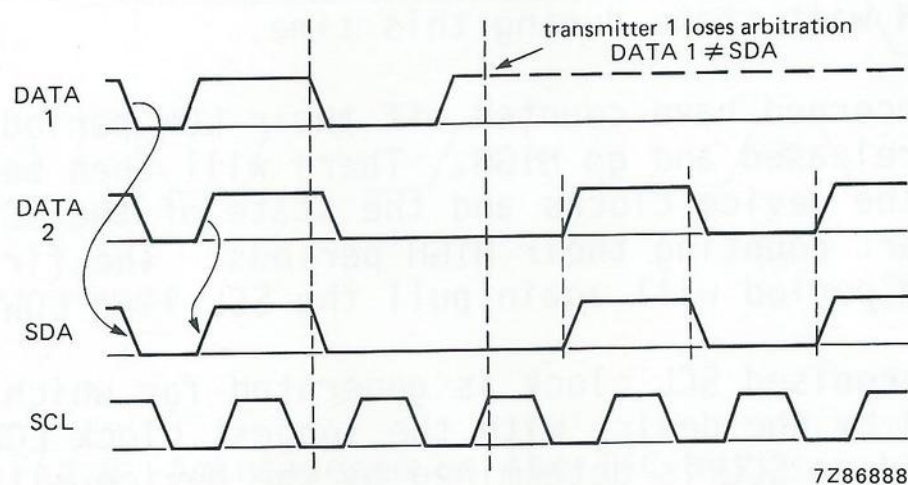


Fig.8 Arbitration procedure of two masters.

A master which loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration.

If a master does lose arbitration during the addressing stage - it is possible that the winning master is trying to address it. The losing master must therefore switch over immediately to its slave receiver mode.

Fig.8 shows the arbitration procedure for two masters. Of course more may be involved - depending on how many masters are connected to the bus. The moment there is a difference between the internal data level of the master generating DATA 1 and the actual level on the SDA line, its data output is switched off, which means that a HIGH output level is then connected to the bus - this will not affect the data transfer initiated by the winning master. As control of the I²C bus is decided solely on the address and data sent by competing masters, there is no central master, nor any order of priority on the bus.

6.3 Use of the clock synchronising mechanism as a handshake

In addition to being used during the arbitration procedure, the clock synchronisation mechanism can be used to enable receiving devices to cope with fast data transfers, either on a byte or bit level.

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. Slave devices can then hold the SCL line LOW, after reception and acknowledge of a byte, to force the master into a wait state until the slave is ready for the next byte transfer in a type of handshake procedure.

On the bit level, a device such as a microcomputer without a hardware I²C interface on-chip can slow down the bus clock by extending each clock LOW period. In this way, the speed of any master is adapted to the internal operating rate of this device.

7.0 FORMATS

Data transfers follow the format shown in Fig.9. After the start condition, a slave address is sent. This address is 7-bits long, the eighth bit is a data direction bit (R/W) - a 'zero' indicates a transmission (WRITE), a 'one' indicates a request for data (READ). A data transfer is always terminated by a stop condition generated by the master. However, if a master still wishes to communicate on the bus, it can generate another start condition, and address another slave without first generating a stop condition. Various combinations of read/write formats are then possible within such a transfer.

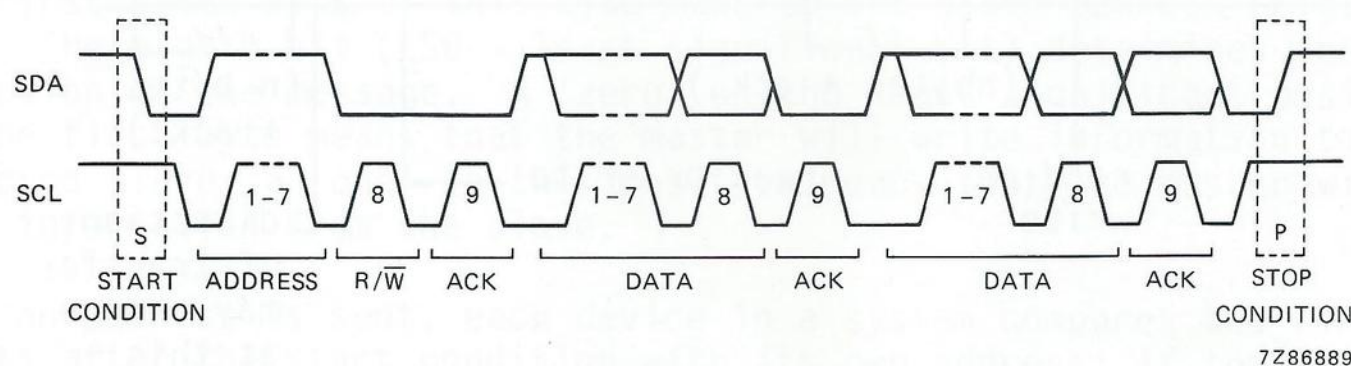
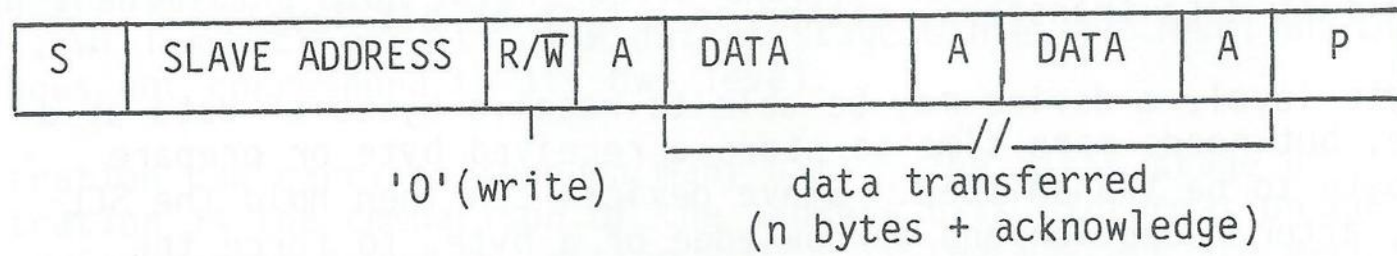


Fig.9 A complete data transfer.

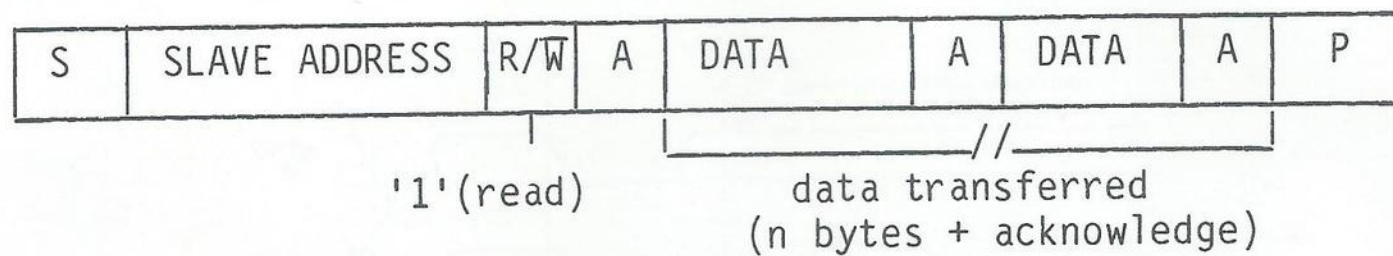
Possible data transfer formats are:

- a) Master transmitter transmits to slave receiver. Direction is not changed.



A = Acknowledge
 S = Start
 P = Stop

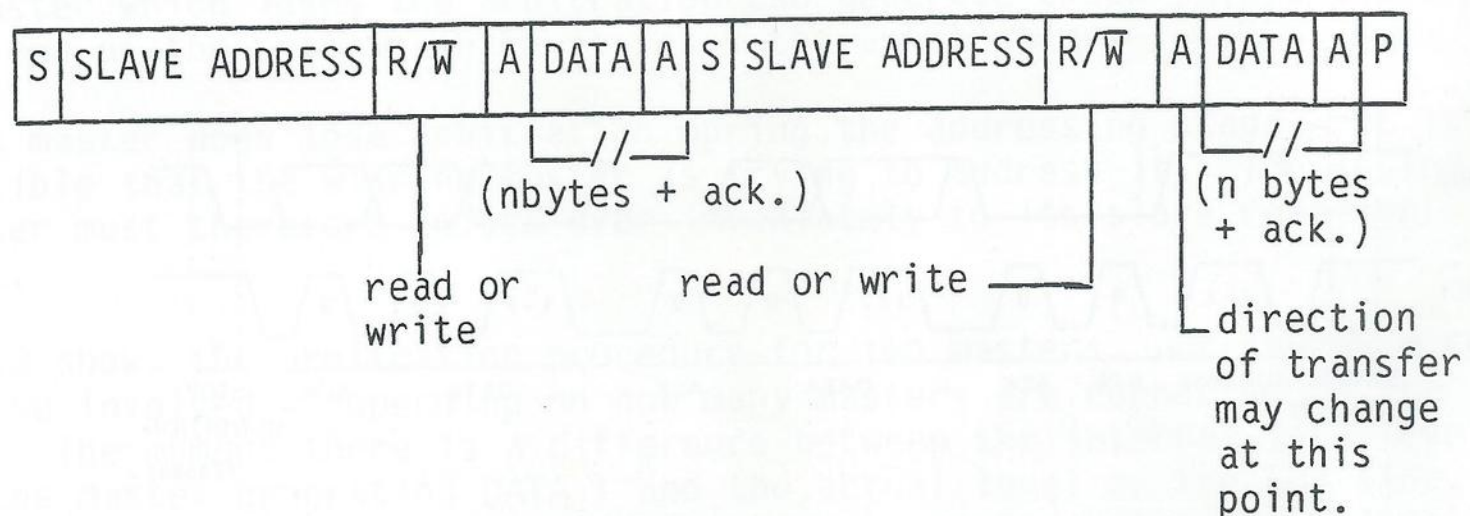
- b) Master reads slave immediately after first byte.



At the moment of the first acknowledge, the master transmitter becomes a master receiver and the slave receiver becomes a slave transmitter. This acknowledge is still generated by the slave.

The stop condition is generated by the master.

- c) Combined formats.



During a change of direction within a transfer, the start condition and the slave address are both repeated, but with the R/W bit reversed.

NOTE:

- 1) Combined formats can be used, for example, to control a serial memory. During the first data byte, the internal memory location has to be written. After the start condition is repeated, data can then be transferred.

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 acknowledge as indicated by the A blocks in the sequence.
- 4) I²C devices have to reset their bus logic on receipt of a start condition such that they all anticipate the sending of a slave address.

8.0 ADDRESSING

The addressing procedure for the I²C bus is such that the first byte after the start condition determines which slave will be selected by the master. Usually, this first byte follows that start procedure. The exception is the 'general call' address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge - although devices can be made to ignore this address. The second byte of the general call address then defines the action to be taken.

8.1 Definition of bits in the first byte

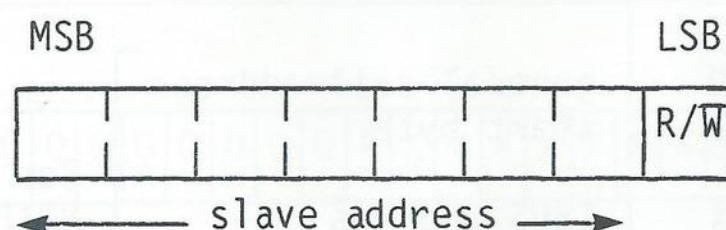


Fig.10 The first byte after the start procedure.

The first seven bits of this byte make up the slave address (Fig. 10). The eighth bit (LSB - least significant bit) determines the direction of the message. A 'zero' on the least significant position of the first byte means that the master will write information to a selected slave; a 'one' in this position means that the master will read information from the slave.

When an address is sent, each device in a system compares the first 7 bits after the start condition with its own address; if there is a match, the device will consider itself addressed by the master as a slave receiver or slave transmitter, depending on the R/W bit.



The slave address can be made up of a fixed and a programmable part. Since it is expected that identical ICs will be used more than once in a system, the programmable part of the slave address enables the maximum possible number of such devices to be connected to the I²C bus. The amount of programmable address bits of a device depends on the amount of pins available. For example, if a device has 4 fixed and 3 programmable address bits, a total of eight identical devices can be connected to the same bus.

The I²C bus committee is available to coordinate allocation of I²C addresses.

The bit combination 1111XXX of the slave address is reserved for future extension purposes.

The address 1111111 is reserved as the extension address. This means that the addressing procedure will be continued in the next byte(s). Devices that do not use the extended addressing do not react at the reception of this byte. The seven other possibilities in group 1111 will also only be used for extension purposes but are not yet allocated.

The combination 0000XXX has been defined as a special group. The following addresses have been allocated:

first byte			
SLAVE ADDRESS		R/W	
0000	000	0	general call address start byte
0000	000	1	
0000	001	X	CBUS address Address reserved for different bus format
0000	010	X	
0000	011	X	to be defined
0000	100	X	
0000	101	X	
0000	110	X	
0000	111	X	

See
NOTES

NOTES:

- No device is allowed to acknowledge at the reception of the start byte.
- The CBUS address has been reserved to enable the inter-mixing of CBUS and I²C devices in one system. I²C bus devices are not allowed to respond at the reception of this address.
- The address reserved for a different bus format is included to enable the mixing of I²C and other protocols. Only I²C devices that are able to work with such formats and protocols are allowed to respond to this address.

8.1.1 General call address

The general call address should be used to address every device connected to the I²C bus. However, if a device does not need any of the data supplied within the general call structure, it can ignore this address by not acknowledging. If a device does require data from a general call address - it will acknowledge this address and behave as a slave receiver. The second and following bytes will be acknowledged by every slave receiver capable of handling this data. A slave which cannot process one of these bytes must ignore it by not acknowledging.

The meaning of the general call address is always specified in the second byte (Fig.11).

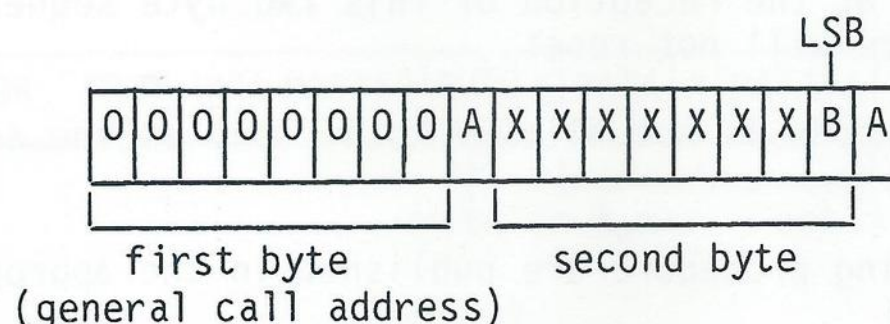


Fig.11 General call address format.

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 B is a zero; the second byte has the following definition:

00000110 (H'06') Reset and write programmable part of slave address by software and hardware.
On receiving this two byte sequence, all devices (designed to respond to the general call address) will reset and take in the programmable part of their address.

Precautions have to 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.

00000010 (H'02') Write slave address by software only.
All devices which obtain the programmable part of their address by software (and which have been designed to respond to the general call address) will enter a mode in which they can be programmed. The device will not reset.

An example of a data transfer of a programming master is shown in Fig.12 (ABCD represents the fixed part of the address)

H'06'

S	H'00'	A	H'02'	A	ABCD000	X	A	ABCD001	X	A	ABCD010	X	A	P
---	-------	---	-------	---	---------	---	---	---------	---	---	---------	---	---	---

Fig.12 Sequence of a programming master.

00000100 (H'04') Write slave address by hardware only.
All devices which define the programmable part of their address by hardware (and which respond to the general call address) will latch this programmable part at the reception of this two byte sequence. The device will not reset.

00000000 (H'00') 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 these codes.

When B is a one; the two byte sequence is a 'hardware general call'. This means that the sequence is transmitted by a hardware master device, such as a keyboard scanner, which cannot be programmed to transmit a desired slave address. Since a hardware master 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 (Fig.13).

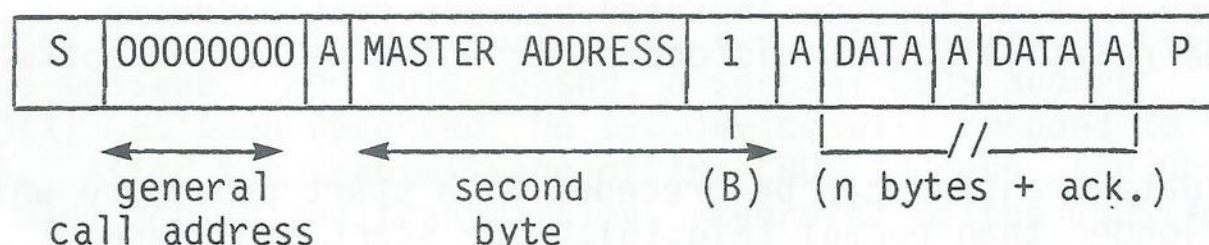
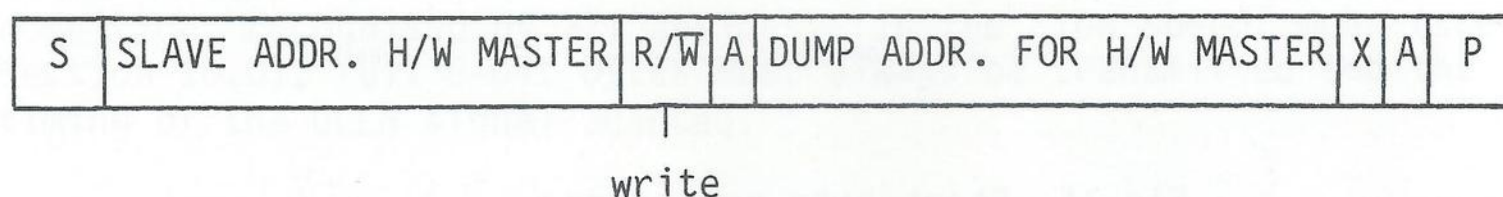


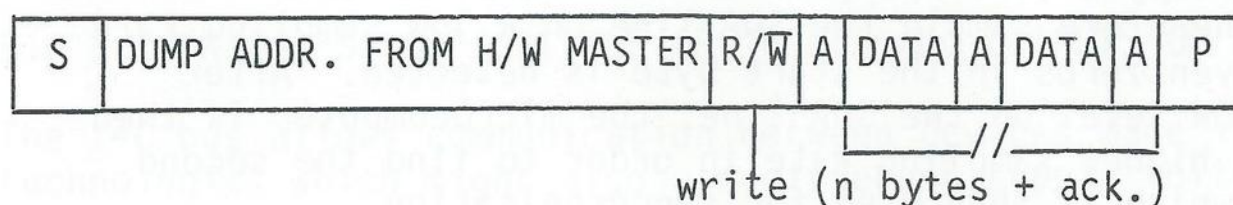
Fig.13 Data transfer from hardware master transmitter.

The seven bits remaining in the second byte contain the device address of the hardware master. This address is recognised by an intelligent device, such as a microcomputer, connected to the bus which will then direct the information coming from the hardware master. If the hardware master can also act as a slave, the slave address is identical to the master address.

In some systems an alternative could be that the hardware master transmitter is brought in the slave receiver mode after the system reset. In this way, a system configuring master can tell the hardware master transmitter (which is now in slave receiver mode) to which address data must be sent (Fig.14). After this programming procedure the hardware master remains in the master transmitter mode.



(a)



(b)

Fig.14 Data transfer of hardware master transmitter capable of dumping data directly to slave devices, (a) configuring master sends dump address to hardware master, (b) hardware master dumps data to selected slave device.

8.1.2 Start byte

Microcomputers can be connected to the I²C bus in two ways. If an on-chip hardware I²C bus interface is present, the microcomputer can be programmed to be only interrupted by requests from the bus. When the device possesses no such interface, it must constantly monitor the bus via software. Obviously, the more times the microcomputer monitors, or polls, the bus - the less time it can spend carrying out its intended function.

Therefore, there is a difference in speed between fast hardware devices and the relatively slow microcomputer which relies on software polling.

In this case, data transfer can be preceded by a start procedure which is much longer than normal (Fig.15). The start procedure consists of:

- a) A start condition S
- b) A start byte 00000001
- c) An acknowledge clock pulse
- d) A repeated start condition Sr

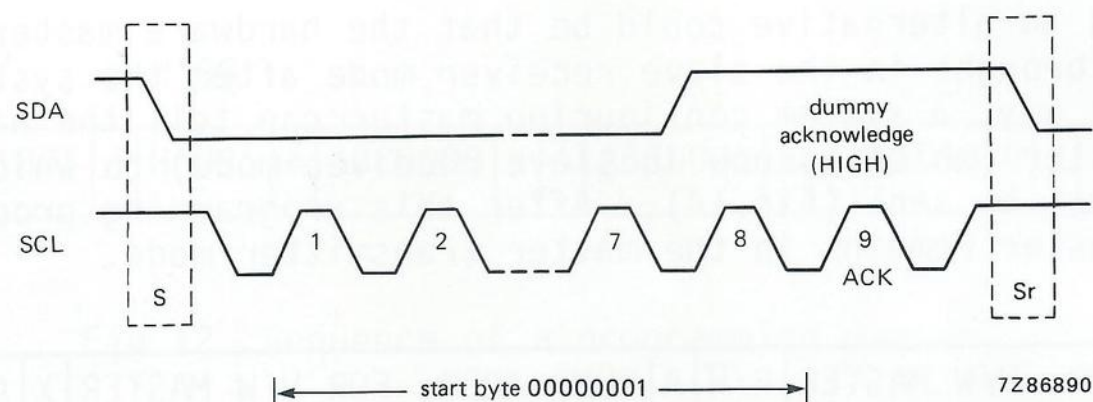


Fig.15 Start byte procedure.

After the start condition S has been transmitted by a master requiring bus access, the start byte (00000001) is transmitted. Another microcomputer can therefore sample the SDA line on 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 microcomputer is then able to switch to a higher sampling rate in order to find the second start condition Sr which is then used for synchronisation.

A hardware receiver will reset at the reception of the second start condition Sr and will therefore ignore the start byte.

After the start byte, an acknowledge-related clock pulse is generated. This is present only to conform with the byte handling format used on the bus. No device is allowed to acknowledge the start byte.

8.1.3 CBUS compatibility

Existing CBUS receivers can be connected to the I²C bus. However, in this case, a third line called DLEN has to be connected and the acknowledge bit omitted. Normally, I²C transmissions are multiples of 8-bit bytes; CBUS devices have different formats however.

In a mixed bus structure, I²C devices are not allowed to respond on the CBUS message. For this reason, a special CBUS address (0000001X) has been reserved. No I²C device will respond to this address. After the transmission of the CBUS address, the DLEN line can be made active and transmission, according to the CBUS format, can be performed (Fig.16).

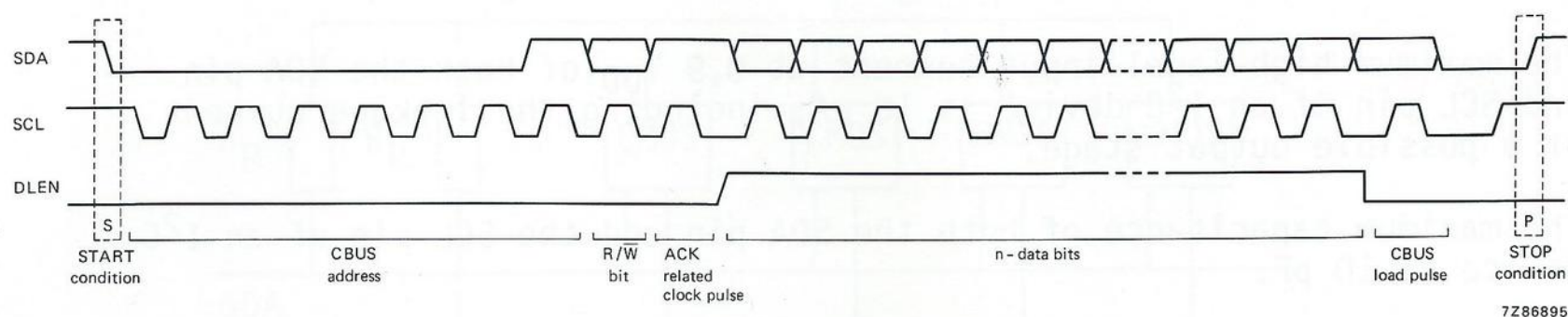


Fig.16 Data format of transmissions with CBUS receiver/transmitter.

After the stop condition, all devices are again ready to accept data.

Master transmitters are allowed to generate CBUS formats after having sent the CBUS address. Such a transmission is terminated by a stop condition, recognised by all devices. In the 'low speed' mode (see Section 10.0), full 8-bit bytes must always be transmitted and the timing of the DLEN signal adapted.

NOTE: If the CBUS configuration is known and no expansion with CBUS devices is foreseen, the user is allowed to adapt the hold time to the specific requirements of device(s) used.

9.0 ELECTRICAL SPECIFICATIONS OF INPUTS AND OUTPUTS OF I²C DEVICES

The I²C bus allows communication between devices made in different technologies which might also use different supply voltages.

For devices with fixed input levels, operating on a supply voltage of 5 V $\pm$ 10%, the following levels have been defined:

$V_{ILmax} = 1,5$ V (maximum input LOW voltage)
 $V_{IHmin} = 3,0$ V (minimum input HIGH voltage)

Devices operating on a fixed supply voltage different from 5 V (e.g. I²L), must also have these input levels of 1,5 V and 3,0 V for V_{IL} and V_{IH} respectively.

For devices operating over a wide range of supply voltages (e.g. CMOS), the following levels have been defined:

$V_{ILmax} = 0,3$ V_{DD} (maximum input LOW voltage)
 $V_{IHmin} = 0,7$ V_{DD} (minimum input HIGH voltage)

For both groups of devices, the maximum output LOW value has been defined:

$V_{OLmax} = 0,4 \text{ V}$ (max. output voltage LOW) at 3 mA sink current.

The maximum low level input current at V_{OLmax} of both the SDA pin and the SCL pin of an I²C device is $-10 \mu\text{A}$, including the leakage current of a possible output stage.

The maximum high-level input current at $0,9 V_{DD}$ of both the SDA pin and SCL pin of an I²C device is $10 \mu\text{A}$, including the leakage current of a possible output stage.

The maximum capacitance of both the SDA pin and the SCL pin of an I²C device is 10 pF.

Devices with fixed input levels can each have their own power supply of $5 \text{ V} \pm 10\%$. Pull-up resistors can be connected to any supply (Fig.17).

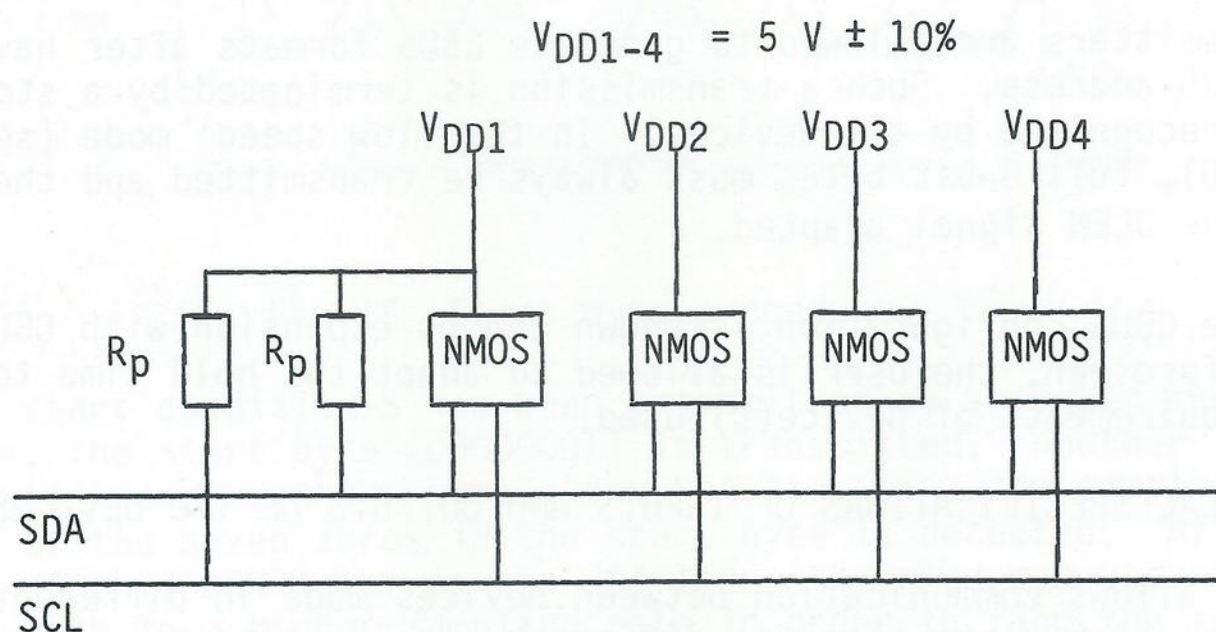


Fig.17 Fixed input level devices connected to the I²C bus.

However, the devices with input levels related to V_{DD} must have one common supply line to which the pull-up resistor is also connected (Fig.18).

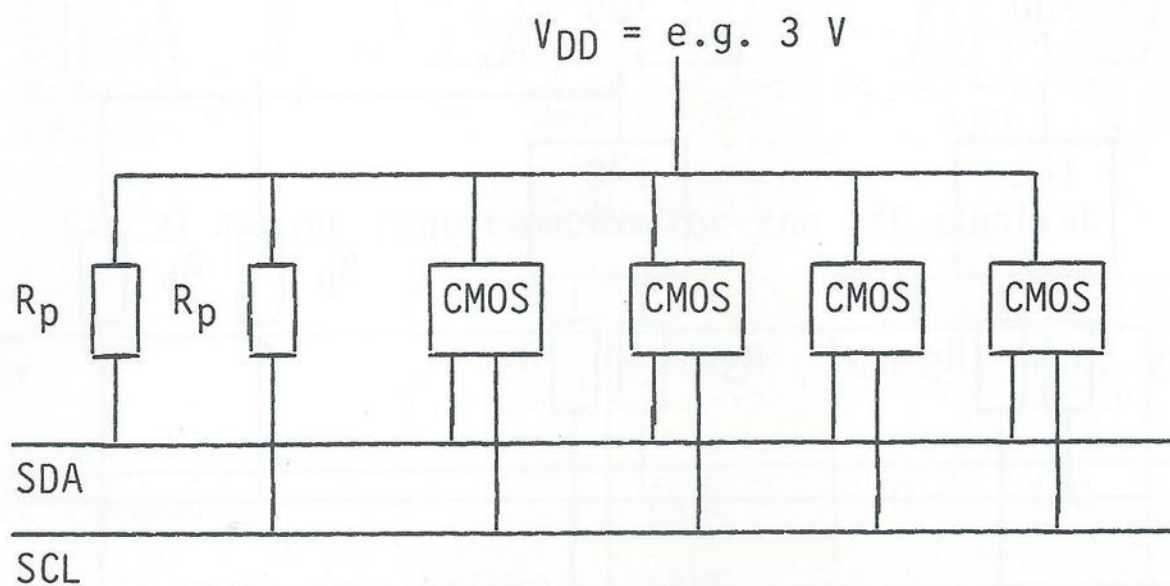


Fig.18 Devices with a wide range of supply voltages connected to the I²C bus.

When devices with fixed input levels are mixed with devices with V_{DD} related levels, the latter devices have to be connected to one common supply line of $5 \text{ V} \pm 10\%$ along with the pull-up resistors (Fig.19)

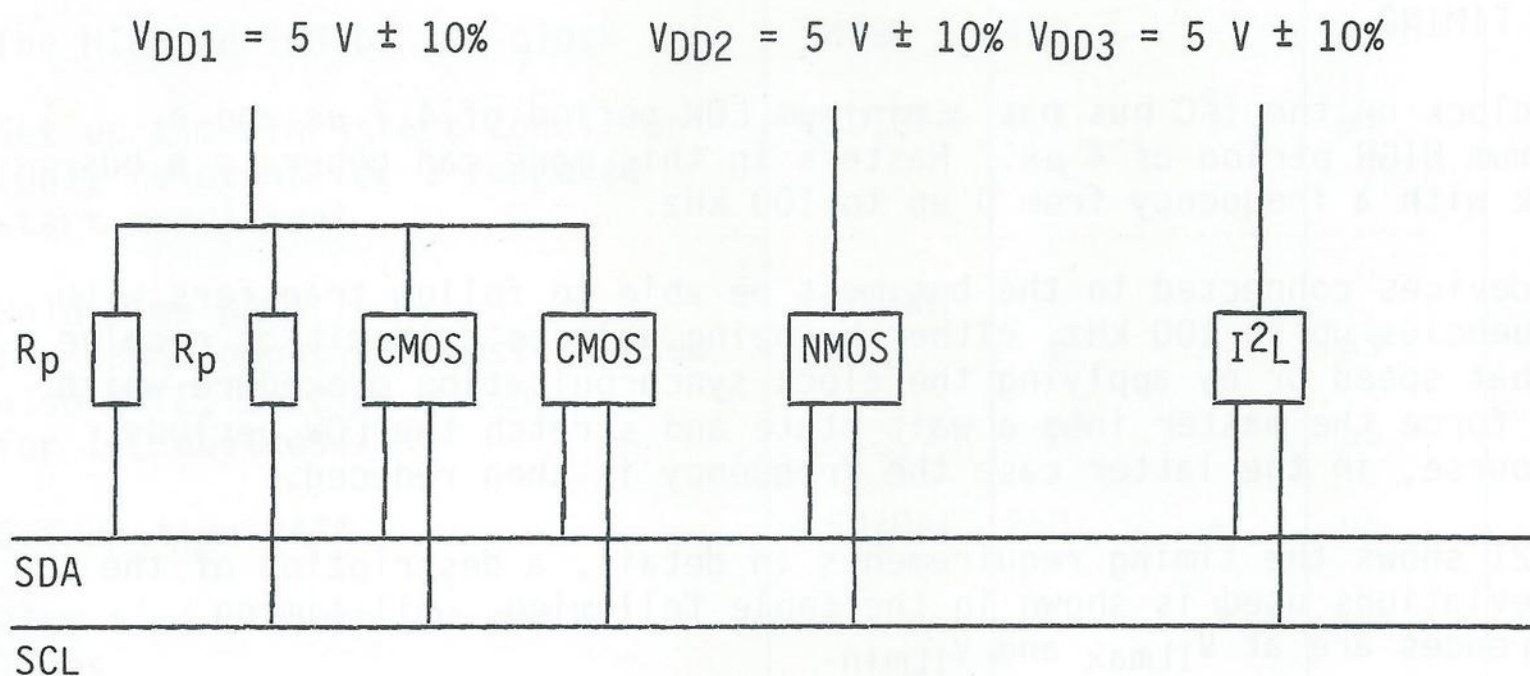


Fig.19 Devices with V_{DD} -related levels mixed with fixed input level devices on the I²C bus.

Input levels are defined in such a way that:

1. The noise margin on the LOW level is $0,1 V_{DD}$.
2. The noise margin on the HIGH level is $0,2 V_{DD}$.
3. Series resistors (R_S) up to 300Ω can be used for flash-over protection against high voltage spikes on the SDA and SCL line due to flash-over of a TV picture tube, for example (Fig.20).

The maximum bus capacitance per wire is 400 pF. This includes the capacitance of the wire itself and the capacitance of the pins connected to it.

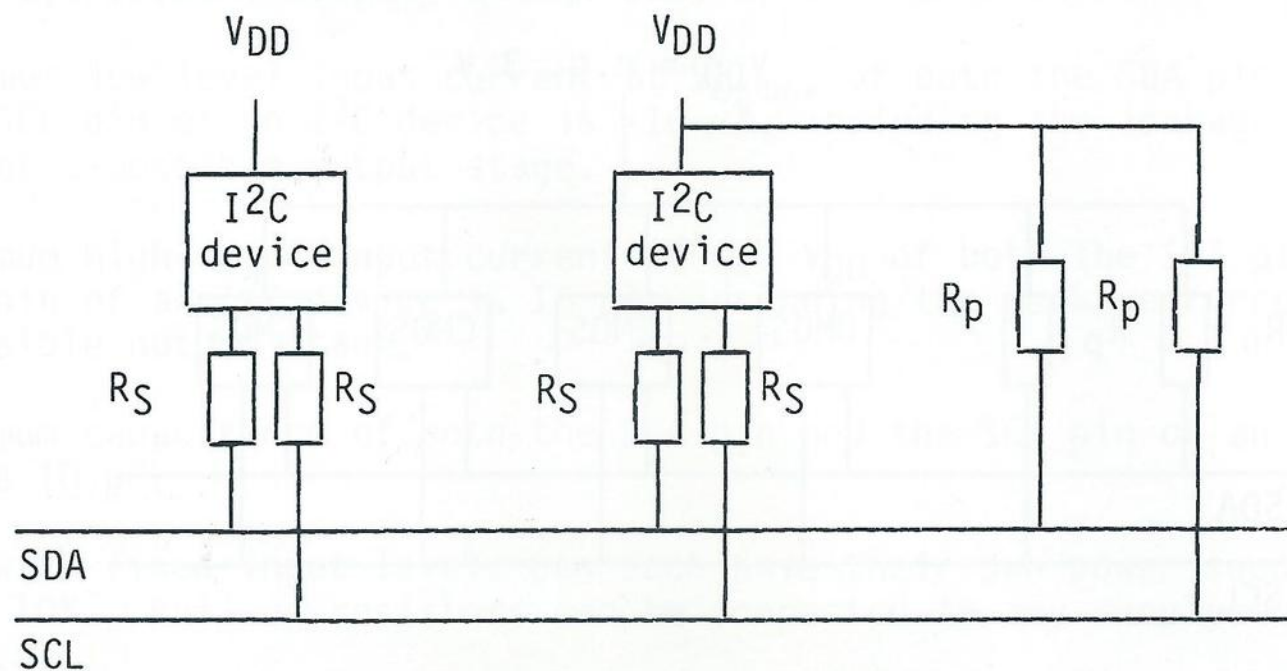


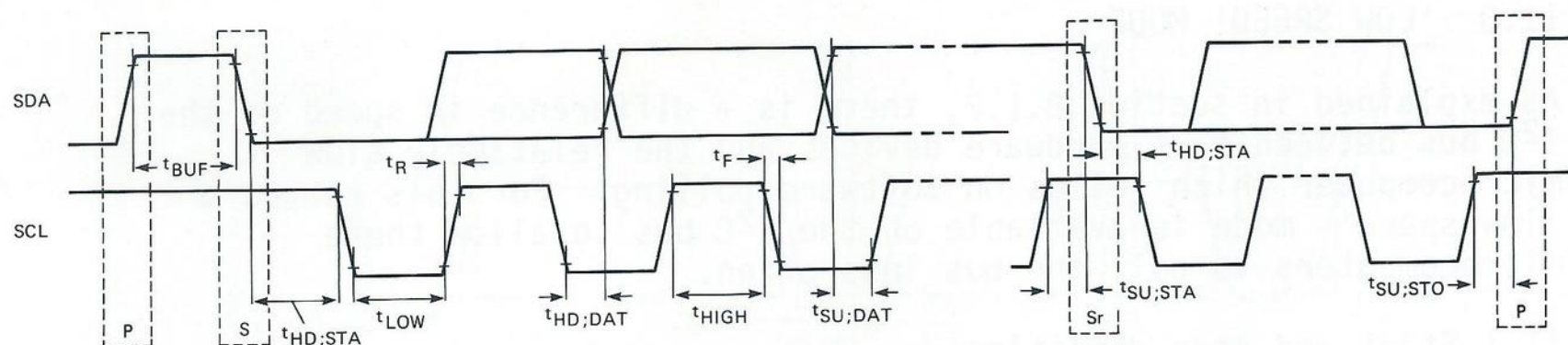
Fig.20 Serial resistors (R_S) for protection against high voltage spikes.

10.0 TIMING

The clock on the I2C bus has a minimum LOW period of 4,7 μs and a minimum HIGH period of 4 μs . Masters in this mode can generate a bus clock with a frequency from 0 up to 100 kHz.

All devices connected to the bus must be able to follow transfers with frequencies up to 100 kHz, either by being able to transmit or receive at that speed or by applying the clock synchronisation procedure which will force the master into a wait state and stretch the LOW periods. Of course, in the latter case the frequency is then reduced.

Fig.21 shows the timing requirements in detail, a description of the abbreviations used is shown in the table following. All timing references are at V_{ILmax} and V_{ILmin} .



7286896

Fig.21 Timing requirements for the I²C bus.

parameter	symbol	min.	max.	units
SCL clock frequency	fSCL	0	100	kHz
Time the bus must be free before a new transmission can start	tBUF	4,7	-	μs
Hold time start condition. After this period the first clock pulse is generated	tHD;STA	4,0	-	μs
The LOW period of the clock	tLOW	4,7	-	μs
The HIGH period of the clock	tHIGH	4,0	-	μs
Set up time for start condition (Only relevant for a repeated start condition)	tSU;STA	4,7	-	μs
Hold time DATA for CBUS compatible masters (see also NOTE, Section 8.1.3) for I ² C devices	tHD;DAT	5	-	μs
		0*	-	μs
Set-up time DATA	tSU;DAT	250	-	ns
Rise time of both SDA and SCL lines	tR	-	1	μs
Fall time of both SDA and SCL lines	tF	-	300	ns
Set-up time for stop condition	tSU;STO	4,7	-	μs
All values referred to VIH and VIL levels (see section 9.0).				
* Note that a transmitter must internally provide at least a hold time to bridge the undefined region (max. 300 ns) of the falling edge of SCL.				

Archief
Ir. A.P.M. Moelands

11.0 'LOW SPEED' MODE

As explained in section 8.1.2, there is a difference in speed on the I²C bus between fast hardware devices and the relatively slow microcomputer which relies on software polling. For this reason a 'low speed' mode is available on the I²C bus to allow these microcomputers to poll the bus less often.

11.1 Start and stop conditions

In the 'low speed' mode, data transfer is preceded by the start procedure of section 8.1.2

11.2 Data format and timing

The bus clock in this mode has a LOW period of $130 \mu s \pm 25 \mu s$ and a HIGH period of $390 \mu s \pm 25 \mu s$, resulting in a clock frequency of 2 kHz approx. The duty cycle of the clock has this LOW to HIGH ratio to allow for more efficient use of microcomputers without an on-chip hardware I²C bus interface. In this mode also, data transfer with acknowledge is obligatory. The maximum number of bytes transferred is not limited (Fig.22).

In this mode, a transfer cannot be terminated during the transmission of a byte.

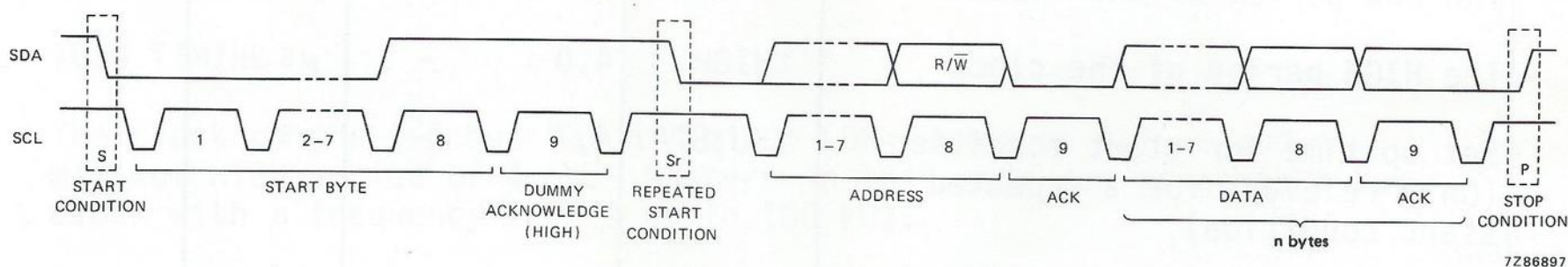
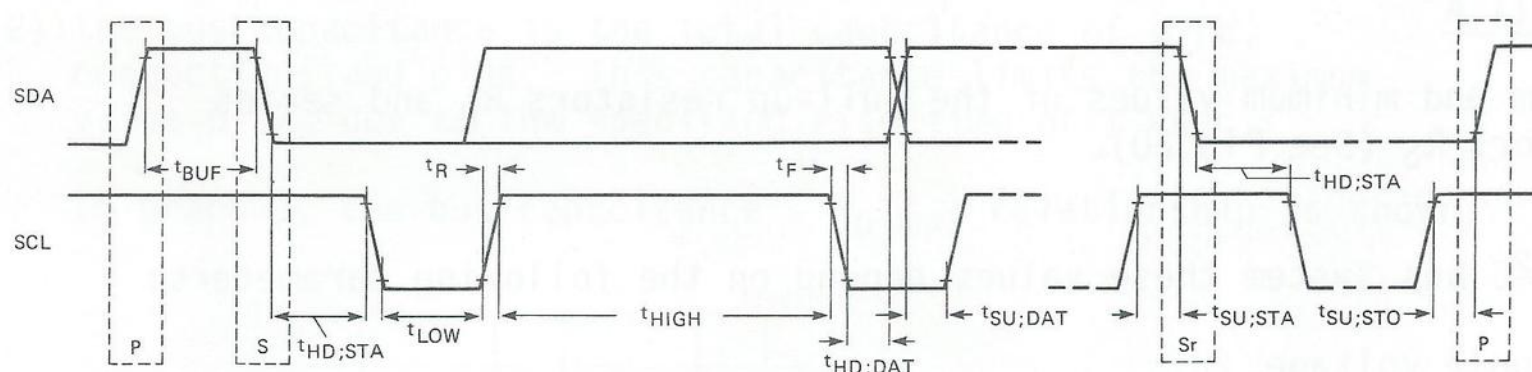


Fig.22 Data transfer 'low speed' mode.

CLOCK	: $t_{LOW} = 130 \mu s \pm 25 \mu s$
	: $t_{HIGH} = 390 \mu s \pm 25 \mu s$
DUTY CYCLE	: 1:3 LOW to HIGH (Duty cycle of clock generator)
START BYTE	: 0000 0001
MAX. NO. OF BYTES	: UNRESTRICTED
PREMATURE TERMINATION OF TRANSFER	: NOT ALLOWED
ACKNOWLEDGE CLOCK BIT	: ALWAYS PROVIDED
ACKNOWLEDGEMENT OF SLAVES	: OBLIGATORY

The bus is considered busy after the first start condition. It is considered free again one minimum clock LOW period, $105 \mu s$, after the detection of the stop condition. Fig.23 shows the timing requirements in detail, the table following explains the abbreviations.



7286898

Fig.23 Timing 'low speed' mode.

parameter	symbol	min.	max.	units
Time the bus must be free before a new transmission can start	tBUF	105	-	μ s
Hold time start condition. After this period the first clock pulse is generated	tHD;STA	365	-	μ s
Hold time (repeated start condition only)	tHD;STA	210	-	μ s
The LOW period of the clock	tLOW	105	155	μ s
The HIGH period of the clock	tHIGH	365	415	μ s
Set up time for start condition (Only relevant for a repeated start condition)	tSU;STA	105	155	μ s
Hold time DATA for CBUS compatible masters (see also NOTE, Section 8.1.3) for I ² C devices	tHD;DAT	5	-	μ s
		0*	-	μ s
Set-up time DATA	tSU;DAT	250	-	ns
Rise time of both SDA and SCL lines	tR	-	1	μ s
Fall time of both SDA and SCL lines	tF	-	300	ns
Set-up time for stop condition	tSU;STO	105	155	μ s
All values referred to VIH and VIL levels (see section 9.0).				
* Note that a transmitter must internally provide at least a hold time to bridge the undefined region (max. 300 ns) of the falling edge of SCL.				

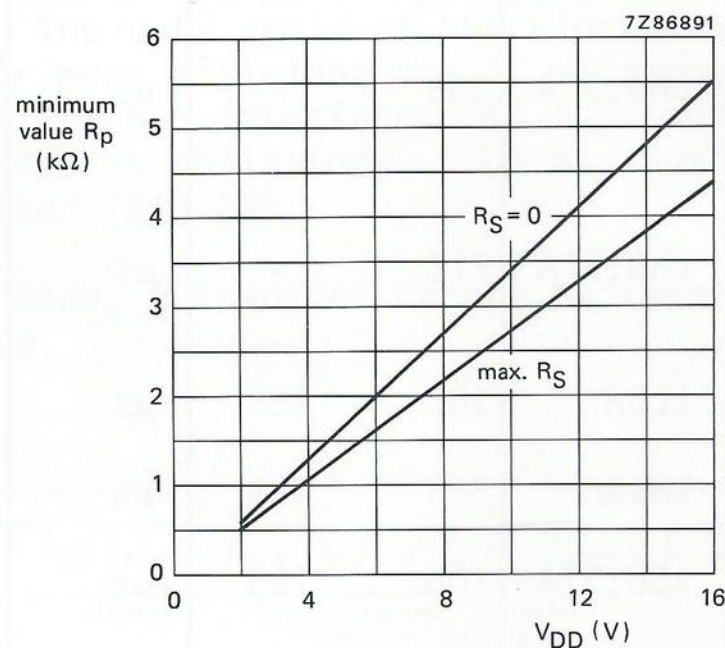
APPENDIX A

Maximum and minimum values of the pull-up resistors R_p and series resistors R_s (See Fig.20).

In a I²C bus system these values depend on the following parameters:

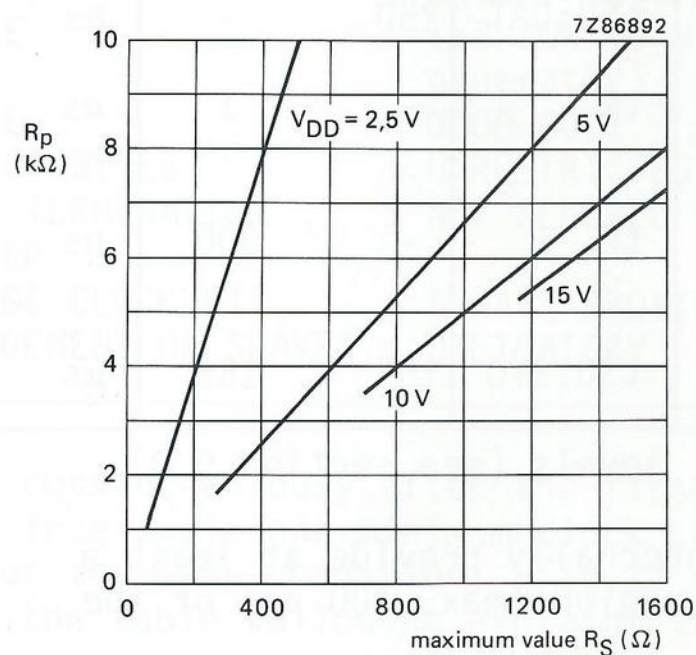
- Supply voltage
- Bus capacitance
- Number of devices (input current + leakage current)

- 1) The supply voltage limits the minimum value of the R_p resistor, due to the specified 3 mA as minimum sink current of the output stages, at 0,4 V as maximum low voltage. In graph 1, V_{DD} against R_p min is shown.



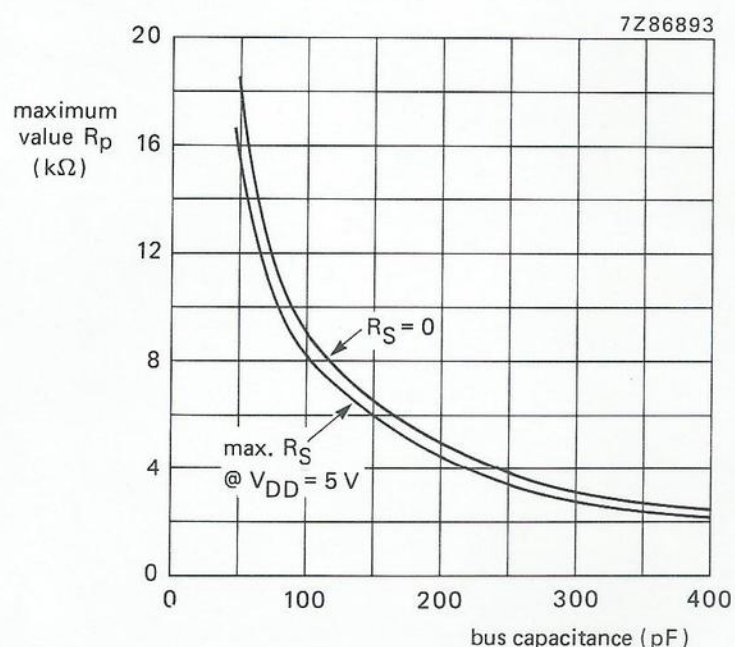
The desired noise margin of 0,1 V_{DD} for the low level limits the maximum value of R_s .

In graph 2, R_s max against R_p is shown.



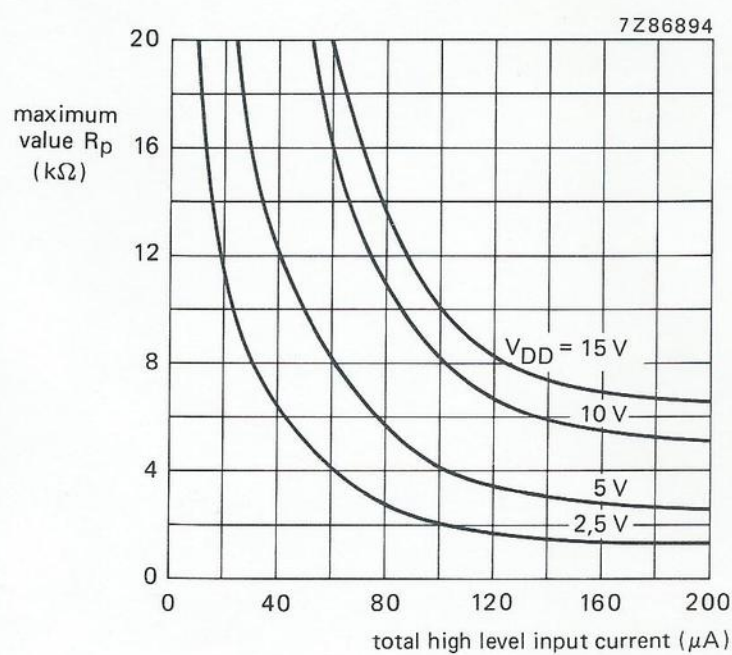
- 2) The bus capacitance is the total capacitance of wire, connections and pins. This capacitance limits the maximum value of R_p due to the specified rise time of $1 \mu\text{sec}$.

In graph 3, the bus capacitance - R_p max relationship is shown.



- 3) The maximum high-level input current of each input/output connection has a specified value of max. $10 \mu\text{A}$. Due to the desired noise margin of $0,2 V_{DD}$ for the high level, this input current limits the maximum value of R_p . This limit is dependent on V_{DD} .

In graph 4 the total high-level input current - R_p max relationship is shown.



The bus capacitance is the total capacitance of wire connections and pins. This capacitance limits the maximum value of k , and the specified rise time of k is dependent on the bus capacitance. In graph 2, the bus capacitance is shown as a function of the bus length and the number of pins.



Graph 2 shows the relationship between bus length and bus capacitance. The bus capacitance is a function of the bus length and the number of pins. The bus capacitance is shown as a function of the bus length and the number of pins. The bus capacitance is a function of the bus length and the number of pins.

The maximum signal level (about 100 mV) is shown in graph 3. The signal level is a function of the bus length and the number of pins. The signal level is shown as a function of the bus length and the number of pins. The signal level is a function of the bus length and the number of pins.



Graph 3 shows the relationship between bus length and signal level. The signal level is a function of the bus length and the number of pins. The signal level is shown as a function of the bus length and the number of pins. The signal level is a function of the bus length and the number of pins.

© 1983 Philips Export B.V.

This information is furnished for guidance, and with no guarantees as to its accuracy or completeness; its publication conveys no licence under any patent or other right, nor does the publisher assume liability for any consequence of its use; specifications and availability of goods mentioned in it are subject to change without notice; it is not to be reproduced in any way, in whole or in part, without the written consent of the publisher.

Printed in The Netherlands

Date of release: January 1983

9398 615 00011