Workshop Mikrorechner des DARC e.V. HOB

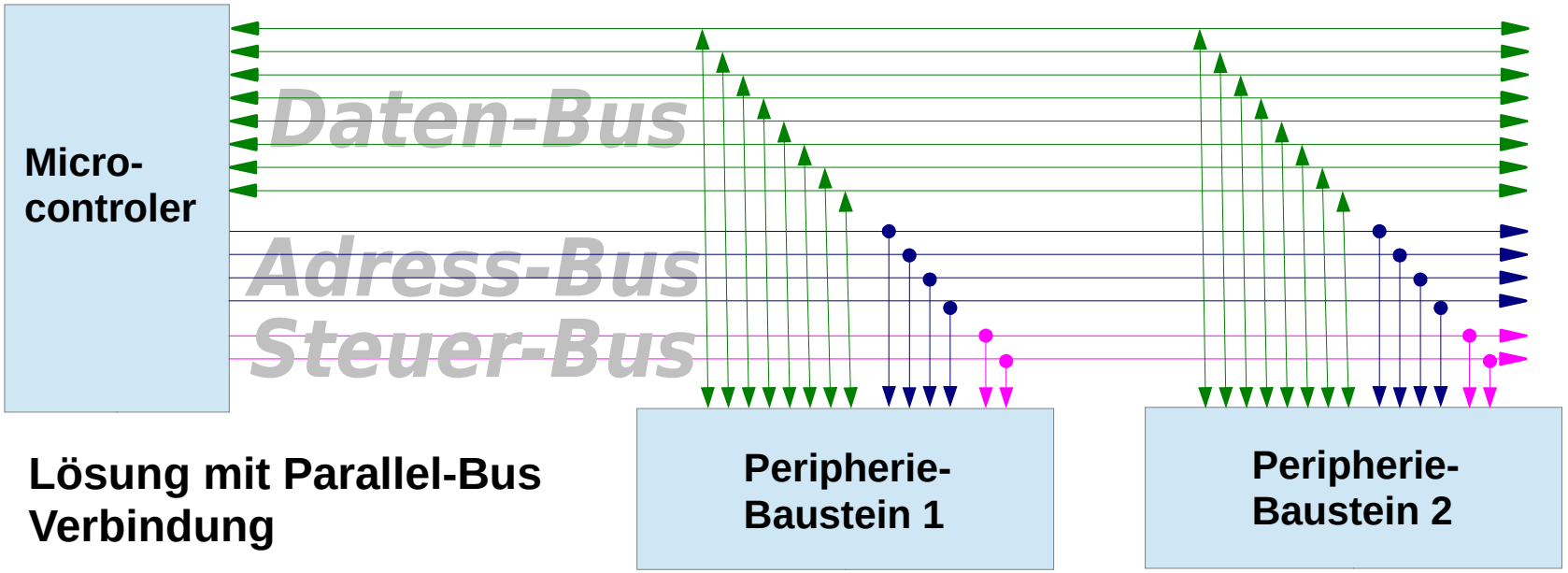
20.04.2013/17.03.2014, DK4AQ

Theorie I2C Bus

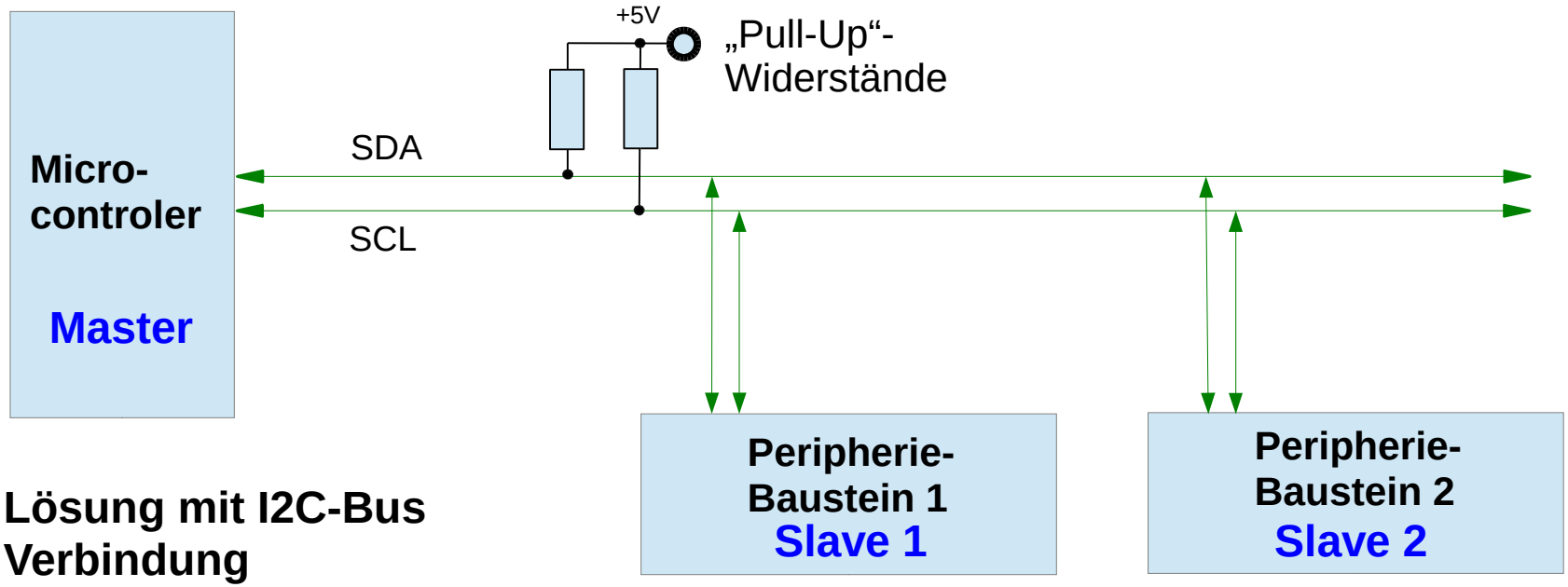
1. Historie und Zweck des I2C-Bus

Historie:
Der niederländische Elektronikkonzern **PHILIPS** hat den I2C-Bus in den frühen 80er Jahren entwickelt, um eine einfache Kommunikation zwischen den Komponenten, die sich auf der gleichen Leiterplatte oder im gleichen Gerät befinden, herzustellen. Philips Semiconductors ging im Jahr 2006 in NXP Semiconductors N.V. Auf.

Nach der Verfügbarkeit von komplexen Microcontrollern (z.B. 8051) und komplexer Peripherie-Bausteine (z.B. Videotext-Generatoren) wurde es notwendig auf Leiterplatten der Unterhaltungselektronik Bussysteme für Daten und Adressen unterzubringen. Das kostete sehr viel Platz auf der Leiterplatte.



Lösung mit Parallel-Bus Verbindung



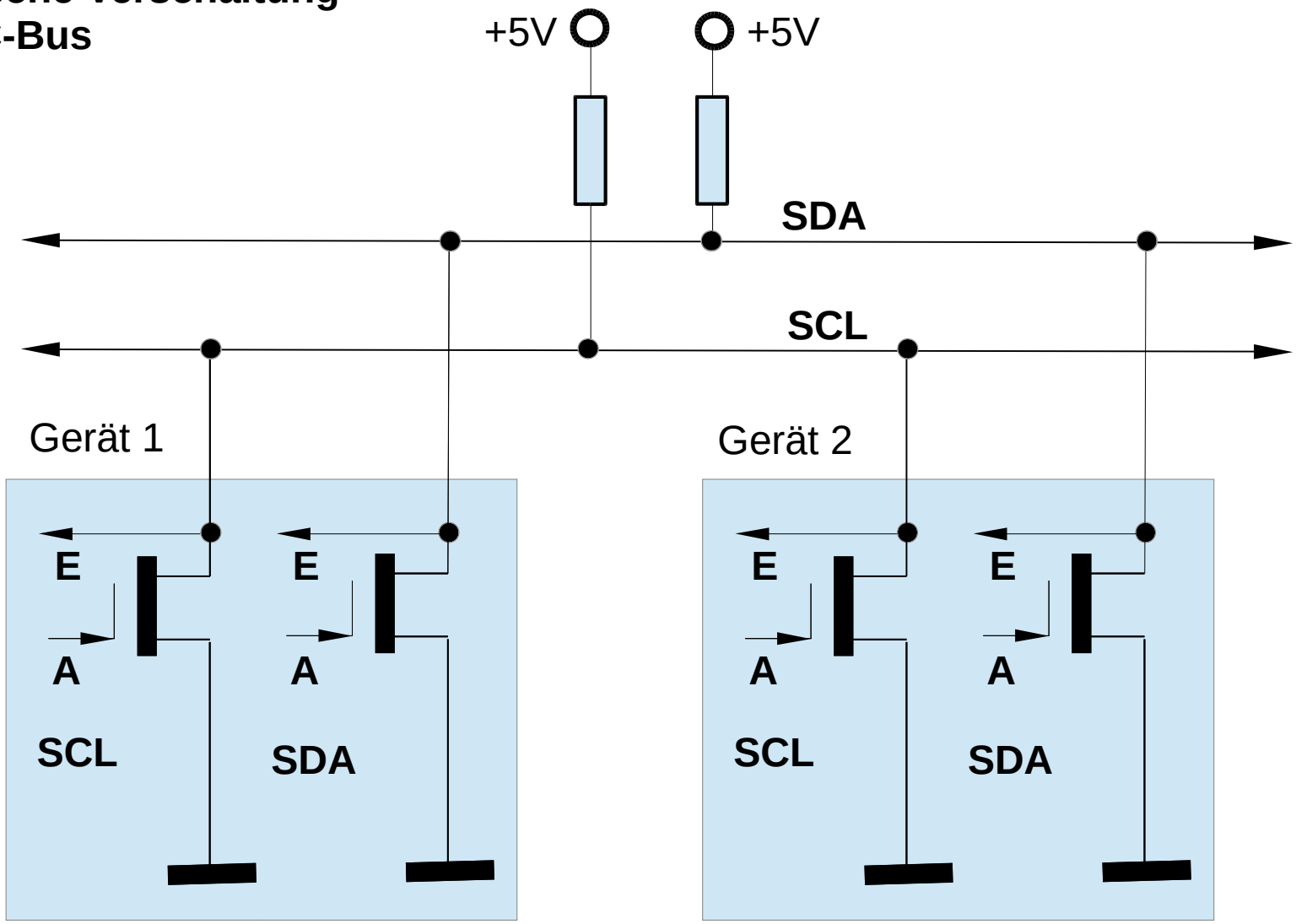
Lösung mit I2C-Bus Verbindung

Es ist deutlich, dass der 2-Draht-Bus deutlich einfacher anzuwenden ist und nur geringe Aufwände bei der Verbindung notwendig sind.

Allerdings erreicht der 2-Draht-Bus nicht den Datendurchsatz wie ein Parallel-Bus. Aber für die Steuerung von Bausteinen, die Lautstärke, Helligkeit einstellen oder die Text-Übertragung durchführen, reicht die Datengeschwindigkeit locker aus.

Für heutige Anwendungen liegt der Vorteil darin, dass es hochintegrierte leistungsfähige ICs und Baugruppen gibt, die mit dieser Schnittstelle versehen sind und die Initialisierung von Registern und Steuerung der Bausteine über I2C erfordern (z.B. GPS, Magn.Kompass, Digitale Oszillatoren...). So ist die I2C Schnittstelle auch heute bei Arduino und Raspberry Pi vorhanden.

Elektrische Verschaltung des I2C-Bus



Alle Geräte greifen über „Open Collector“-Schnittstellen zu.Die Transistorschalter können die Leitungspegel nur gegen Masse ziehen. Im Ruhezustand liegen beide Leitungen auf 5V-Pegel.

Der Name des Busses kommt folgendermassen zustande :

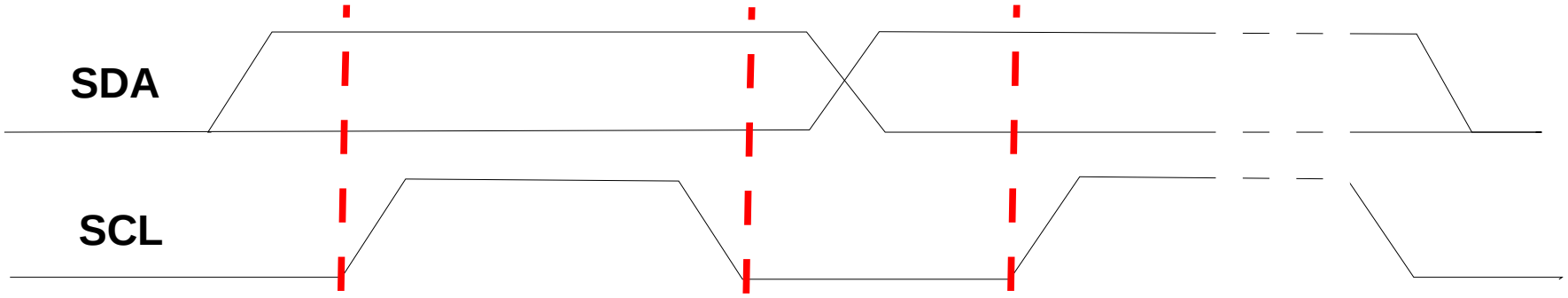
Inter-IC-Bus => **IIC-Bus** => **I2C-Bus**, auch als **„ I Square C Bus“** oder **„I-Quadrat-C-Bus“** gelesen.

Die aktuelle Spezifikation ist in dem Dokument ***UM10204 „I2C-bus specification and user manual“, Rev. 5 — 9, October 2012*** der Fa. NXP zu finden.
http://www.nxp.com/documents/user_manual/UM10204.pdf

2. Übertragungsprinzip mit SDA und SCL

Die beiden Leitungen haben folgende Bedeutung:
SDA „Serielle Daten“ Bitweise Übertragung der Daten
SCL „Serielle Clock“ Signal zur Datenübernahme

SCL wird (normalerweise) vom Master erzeugt, SDA kann sowohl vom Master als auch vom Slave kommen



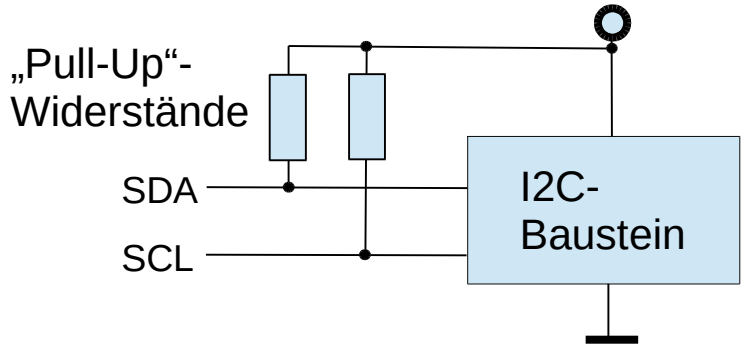
Hier wird der Datenwert übernommen

Solange muss der Datenwert stabil stehen

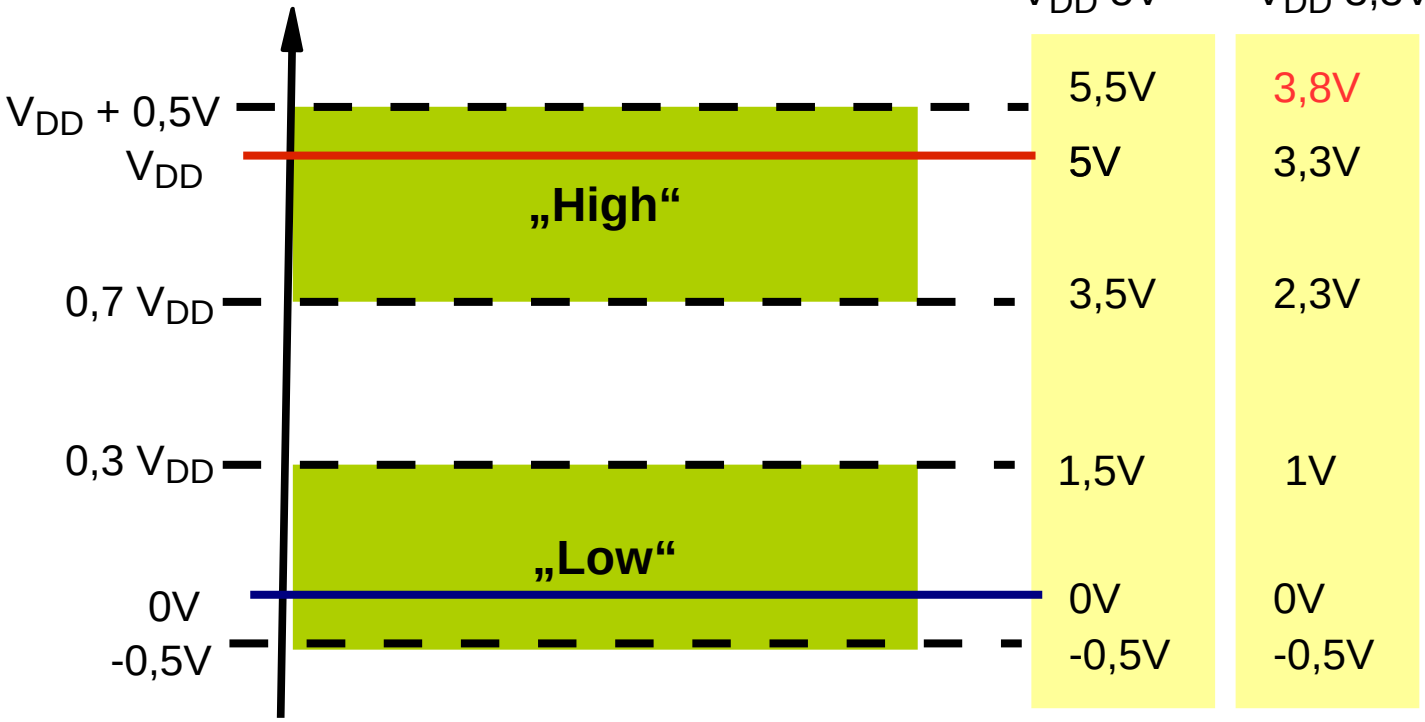
Hier darf der Datenwert sich ändern

3. OC-Leitungseigenschaften

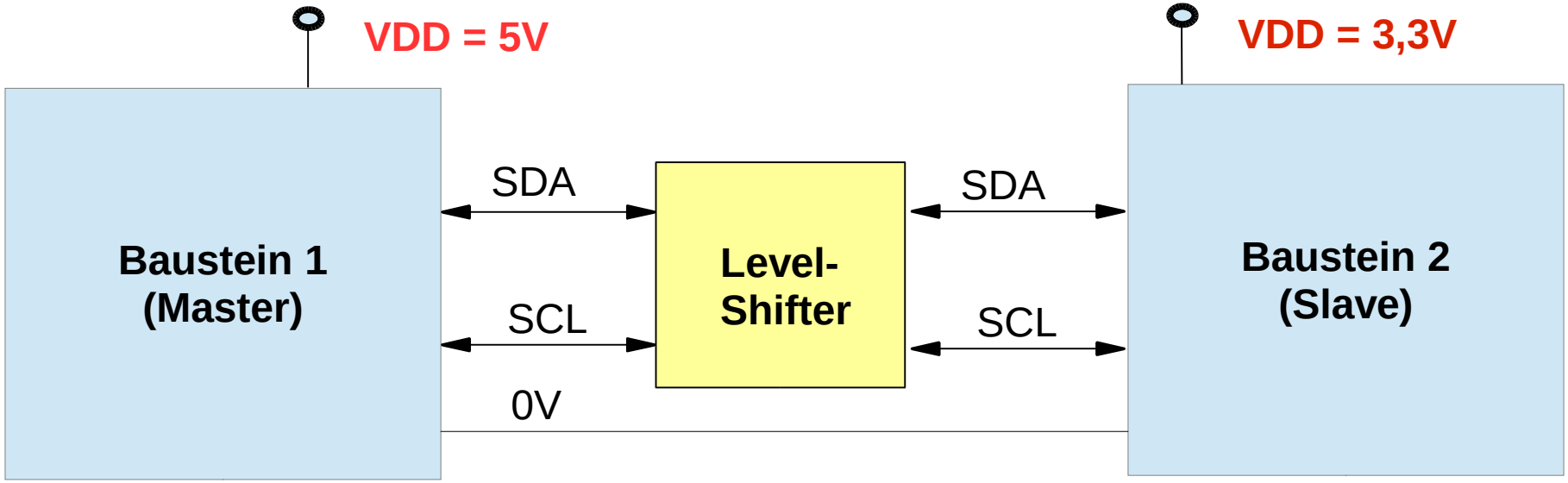
3.1 Leitungs-Pegel



Wenn Bausteine mit unterschiedlichen Betriebsspannungen an einem I2C-Bus arbeiten sollen, dann müssen Level.shifter eingesetzt werden !



Kopplung von Bausteinen mit unterschiedlicher Betriebsspannuing



HIGH	5V 3,5V		3,3V 2,3V	HIGH
LOW	1,5V 0V		1V 0V	LOW

Bei Kopplung von I2C-Bausteinen mit unterschiedlicher Betriebsspannung muss ein Level-Shifter zur Anpassung der Pegel eingesetzt werden.
Wenn eine direkte Verbindung der Leitungen stattfindet, dann kann der Baustein mit der niedrigen Betriebsspannung zerstört werden !
Ausnahme: „5V-tolerante“ Bausteine (siehe Datenblatt)

3.2 Clock-Rate

Die Taktfrequenz des SCL-Signals bestimmt die Übertragungsgeschwindigkeit.

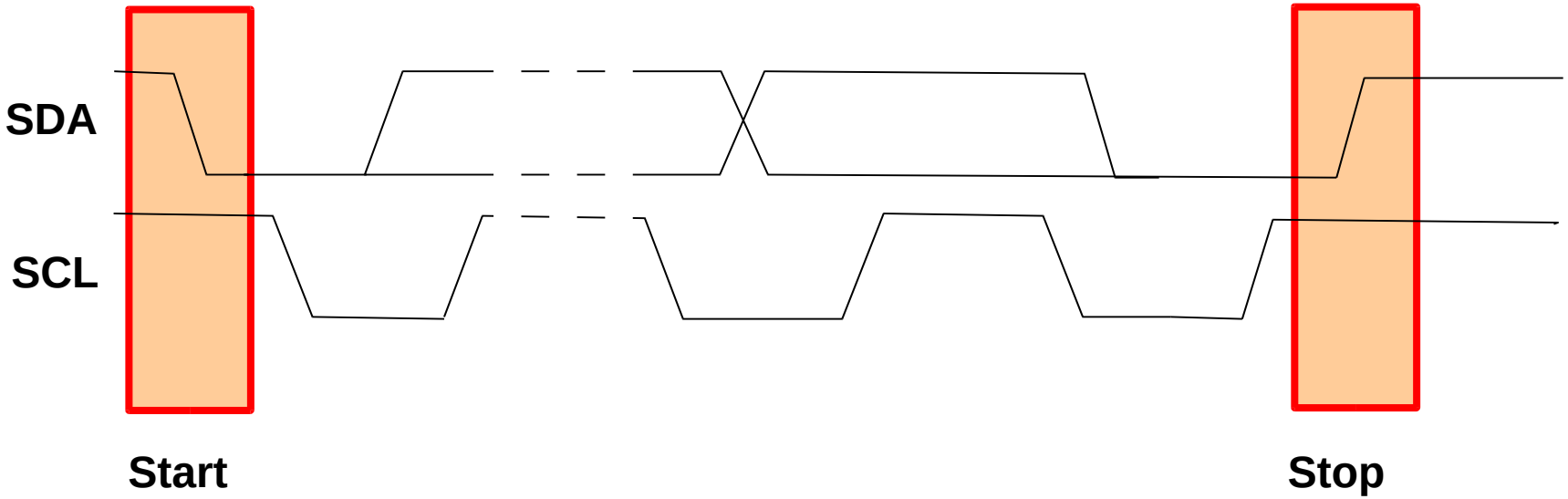
Takt-frequenz	Mode	Daten-rate	Dauer 1 Byte (ca.)*	Anmerkung
100kHz	Standard (S),	100kb/s	10kB/s	1980
400kHz	Fast Mode (F)	400kb/s	40kB/s	Ab 1992
1MHz	Fast Mode Plus	1Mb/s	100kB/s	Ab 2007
3,4MHz	High Speed (HS)	3,4Mb/s	340kB/s	Ab 1998

* reiner Datentransfer ohne Startbyte und Abstände

4. Bus Ereignisse

4.1 Start Condition

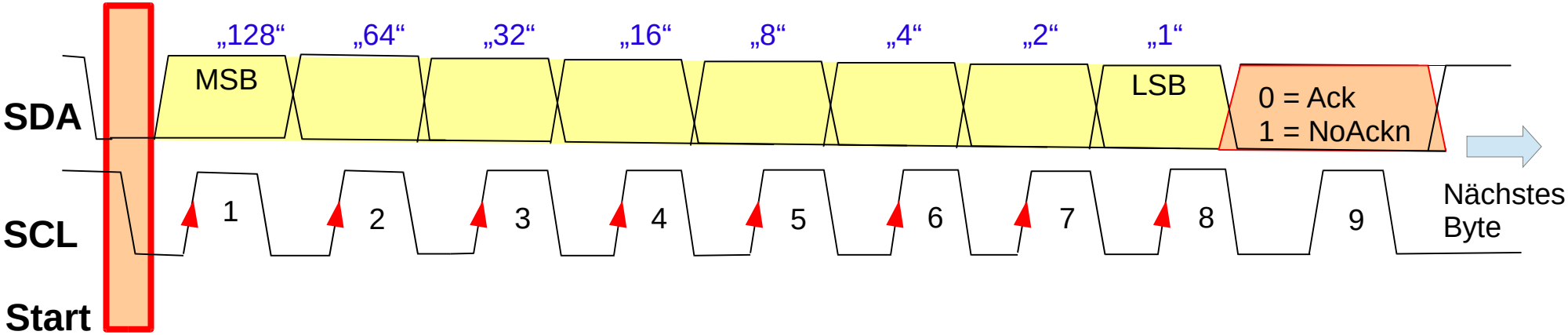
4.2 Stop Condition



Der Master setzt das SDA-Signal auf LOW und generiert dann eine negative Flanke auf SCL. Damit wird der **Start** einer Sendung angekündigt

Nach dem letzten übertragenen Byte wird vom Master das SCL-Signal auf HIGH gesetzt und eine positive Flanke auf SDA generiert. Damit beendet der Master seine Sendung mit **Stop.**

4.3 Acknowledge
4.4 NoAcknowledge

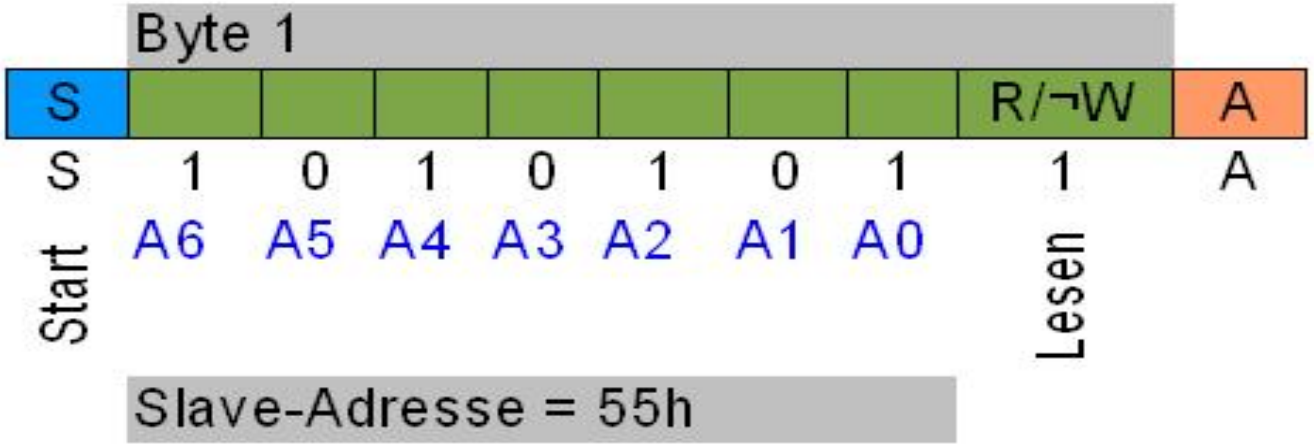


Nach Empfang eines Bytes , sendet der Empfänger **(!)** auf der Datenleitung einen Low-Impuls (9). Dieser Impuls ist die Information **Acknowledge** (Bestätigung) und bedeutet, der Empfänger wurde angesprochen und hat das Byte empfangen. Es bedeutet allerdings keine Bestätigung des richtigen Dateninhalts.

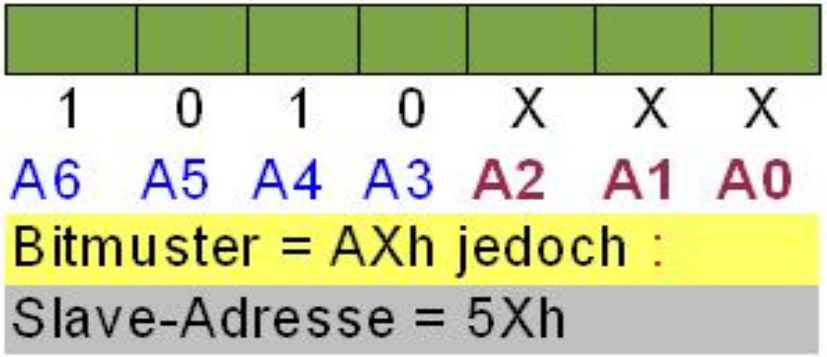
Wenn der Impuls (9) fehlt, bestätigt der Empfänger nicht und damit wird **NoAcknowledge** vom Sender empfangen. Dies ist ein typischer Fehler wenn die Adresse des Bausteins nicht stimmt **NoAcknowledge** wird auch gesendet, wenn der Empfänger nach eine Reihe von Bytes kein weiteres Byte mehr aufnehmen will.

Sender oder Empfänger können je nach Lese- oder Schreibvorgang sowohl Master als auch Slave sein !

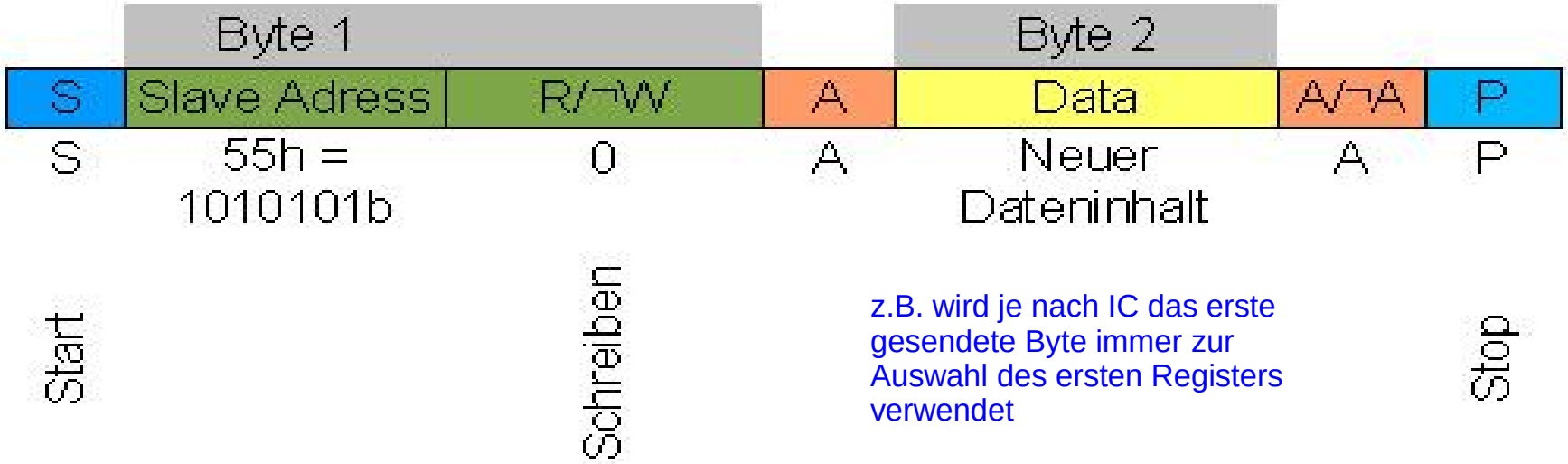
5. Zugriff auf Slave
5.1 Adressierung Slaves



Die Slave-Adresse des einzelnen Bausteins wird im ersten Byte mit 7 bit abgelegt.
Damit lassen sich 128 Adressen bilden.
Die Adresse für einen bestimmten Bausteintyp (teilweise für eine Fertigungscharge) wird vom Hersteller festgelegt. Damit man in einem System mehrere Bausteine nutzen kann, ist meist ein Teil der Adressbits von aussen per Hardware mit Drahtbrücken adressierbar (z.B. A0,A1,A2) :

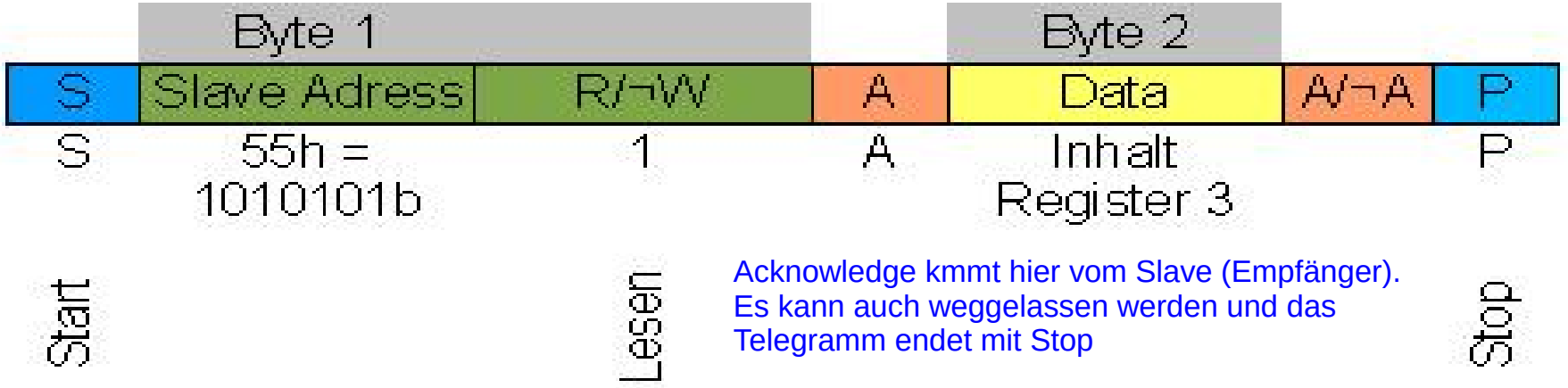


5.3 Schreiben Byte in Slave-Register



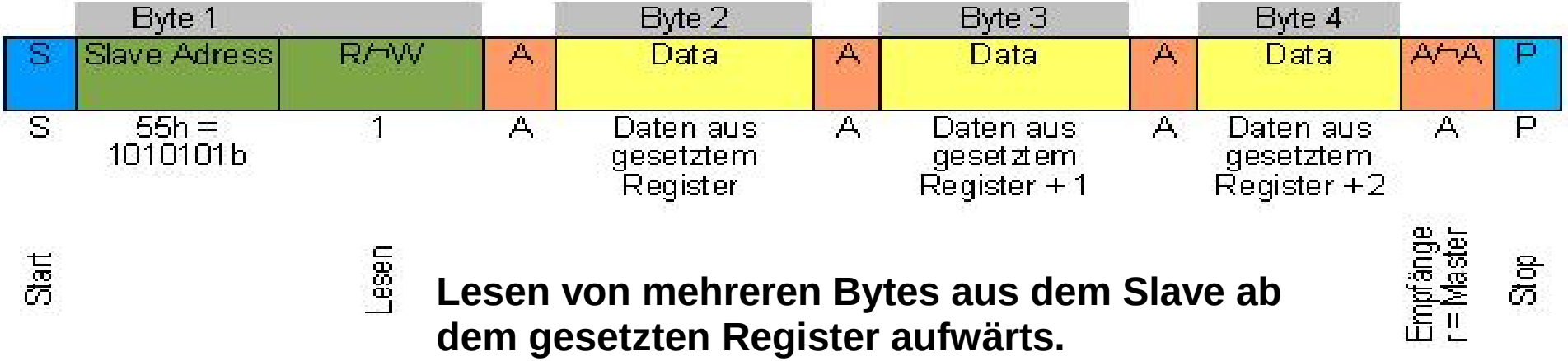
Schreiben eines Bytes zum Slave

5.4 Lesen Byte aus Slave Register



Lesen eines Bytes vom Slave

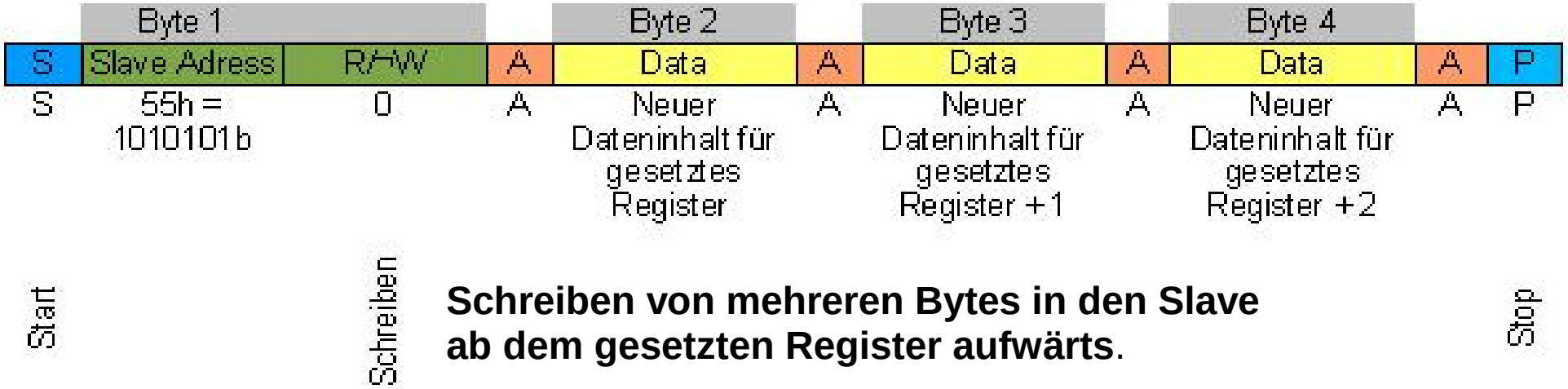
5.5 Lesen mehrerer Bytes



Lesen von mehreren Bytes aus dem Slave ab dem gesetzten Register aufwärts.

Acknowledge kmmt hier vom Slave (Empfänger). Es kann auch weggelassen werden und das Telegramm endet mit Stop

5.6 Schreiben mehrerer Bytes



Schreiben von mehreren Bytes in den Slave ab dem gesetzten Register aufwärts.

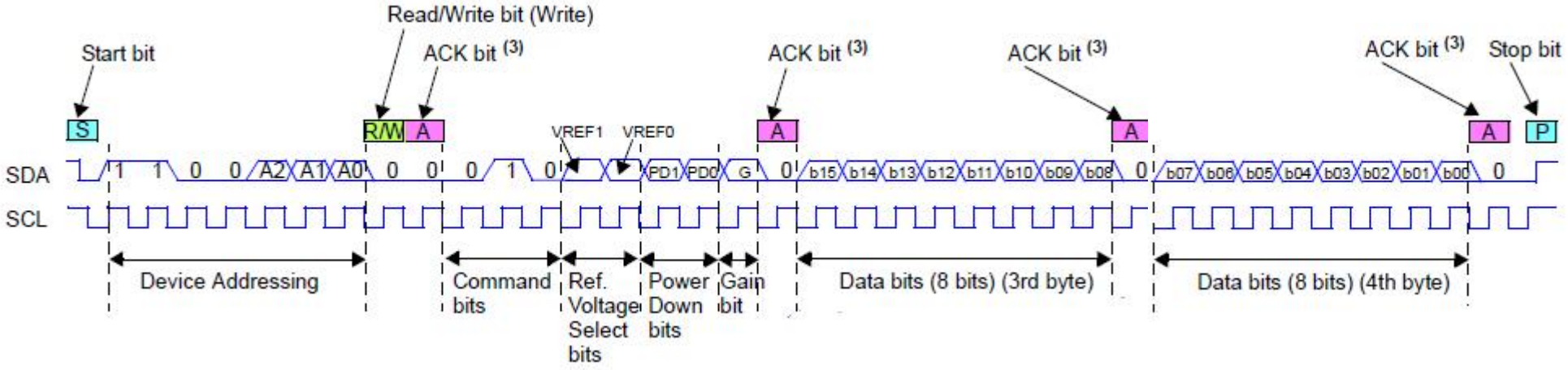
Acknowledge kommt vom Empfänger (Slave), Beendung durch Einstellung der SCL-Clocks durch den Master

Die Bedeutung der übertragenen Daten-Bytes sowie die Auswahl des ersten zu übertragenden Bytes ist herstellerspezifisch !

Es gibt hier keinen Standard.

D.h. Wenn man ein unbekanntes IC an den Bus anschließen will, muss man aus dem Datenblatt herausfinden, in welcher Reihenfolge der Hersteller welche Informationen übertragen will.

Das heißt auch, dass man als Anwender einer Software-Bibliothek den Aufbau eines Telegramms durch Aufrufreihenfolge steuern muß.



Beispiel: Hier wird das erste Datenbyte immer als Kommando-Byte genutzt und immer 2 Datenbytes nachträglich gelesen

5.7 Reservierte Adressen

Folgende Adresse sollen nicht für eigene Zwecke benutzt werden:

SLAVE ADDRESS	$\overline{R/W}$ BIT	DESCRIPTION
0000 000	0	General call address
0000 000	1	START byte(1)
0000 001	X	CBUS address(2)
0000 010	X	Reserved for different bus format(3)
0000 011	X	Reserved for future purposes
0000 1XX	X	Hs-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

Notes

1. No device is allowed to acknowledge at the reception of the START byte.
2. 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.
3. 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.

6. Besonderheiten

6.1 Langsame Slaves (Clock-Stretching)

Es kann vorkommen, dass der Slave, der angesprochen werden soll, gerade nicht in der Lage ist zu antworten. Heutzutage tritt das im wesentlichen nur noch bei langsamen Technologien auf. Beispiele sind EEPROMs mit langsamen Zugriffszeiten oder Software-basierenden Slaves. Auch Synchronisierungsprobleme lassen sich damit in den Griff bekommen.
In solchen Fällen ist es dem Slave erlaubt die SCL-Leitung auf LOW-Level zu halten bis er reagieren kann. Er dehnt sozusagen den Clock-Impuls aus. Problem: Es wird der gesamte Bus ausgebremst.

6.2 Multimasterbetrieb

Prinzipiell ist der I2C-Bus in der Lage mit mehreren Mastern zu arbeiten. Allerdings ist dabei eine Erkennung notwendig, wenn mehrere Master gleichzeitig zugreifen wollen oder wenn eine Übertragung gestört wird. Diese Mechanismen sind in den meisten Bausteinen nicht enthalten.
Für Hobbyzwecke dürfte so eine Anwendung jedoch überzogen sein.

6.3 Anwendung von Interrupts

Manche ICs haben einen Interruptausgang (INT). Dieser Ausgang hat meist auch Open-Collector-Eigenschaften und kann über eine weitere Leitung auf einen Interrupteingang des Prozessors arbeiten. Der Prozessor kann sich nach Eingang des Interrupt bevorzugt um eine Abrage des Bausteins kümmern. Evtl. fragt der Propzessor den gesamten I2C-Bus nur einmal ab, wenn er einen Interrupt erkannt hat.

7. Beispiel I2C-Hardware in Microcontrollern

7.1 ATmega328 (Arduino)

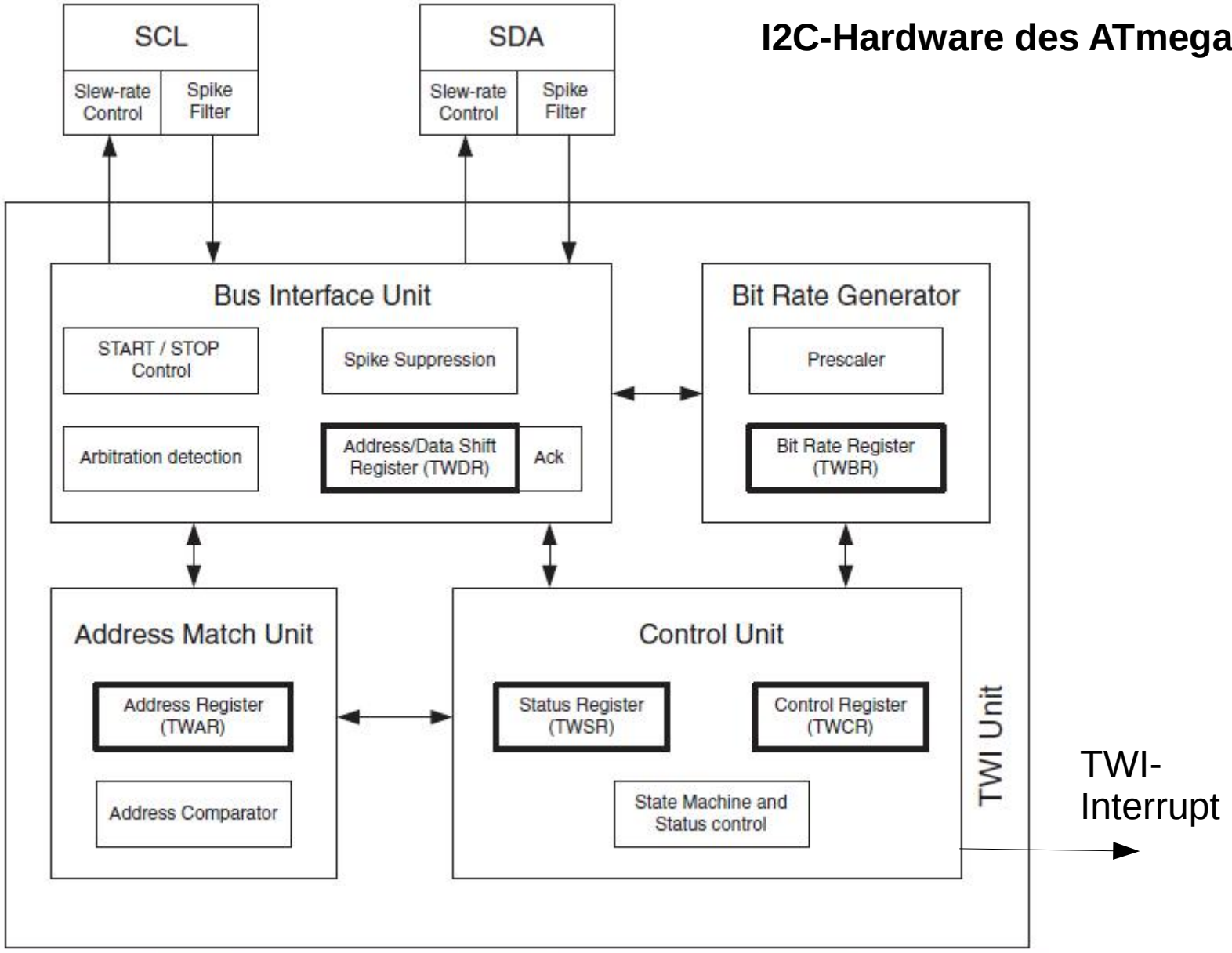
Die I2C-Schnittstelle wird bei ATMEL Two-Wire-Interface genannt.

- Unterstützt Master- und Slave-Betrieb
- Unterstützt Multimaster-betrieb
- Gleichzeitiger Betrieb I2C- und SPI-Bus und UART
- Kann als Sender oder Empfänger arbeiten
- Arbeitet mit 7 bit-Adressen (max.128 Einheiten adressierbar)
- Max. 400kb/s Datentransfer
- Ausgangssignal mit Tiefpass-Filterung
- Eingangssignal mit Unterdrückung von Störimpulsen
- Slave-Adressbereich voll nutzbar (128 Adr.) mit General Call-Erkennung (Adr. 00h)
- Kompatibel mit Philips I2C-Protokoll

Der I2C-Bus im Atmega328 wird durch Hardware unterstützt. Im Gegensatz zu reinen Softwareimplementationen, die sich um jedes Bit kümmern müssen, wird hier durch die Software nach Start eines Telegramms sehr viel in Hardware durchgeführt. Das verbessert die Arbeitsgeschwindigkeit und belastet die Software nur wenig.

Tiefpassfilter und Unterdrückung von Störspitzen zur Verbesserung der Störsicherheit sind integriert. Die Hardware übernimmt die Erkennung von Start- und Stop-Conditions. Die Einfügung des Acknowledge-Bits und den Zusammenbau des 1. Byte mit Adresse bzw. dem Vergleich eintreffender Adressen werden von der Hardware unterstützt Empfang und Erzeugung des Datensignals wird über ein Schieberegister durchgeführt.

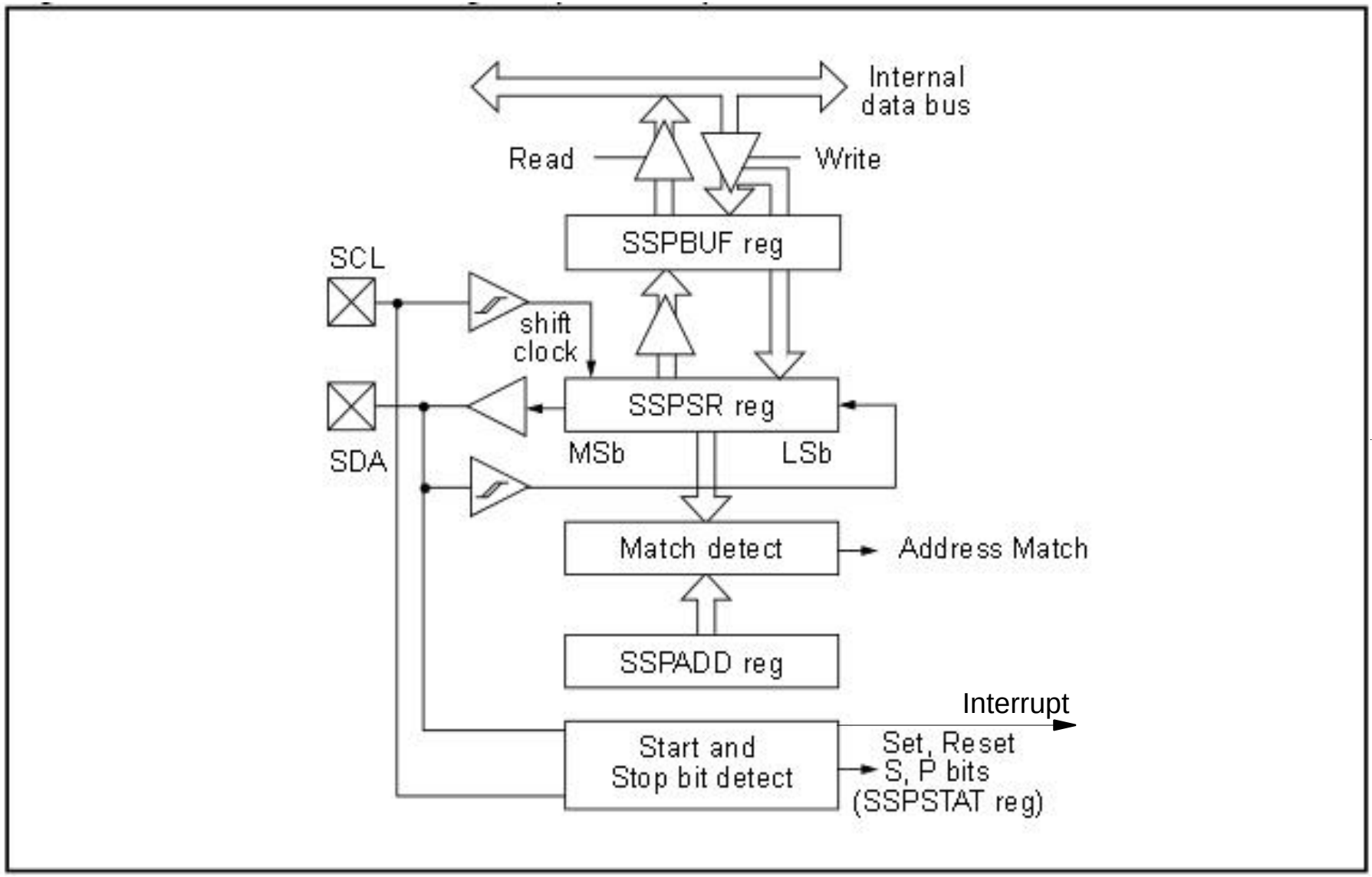
I2C-Hardware des ATmega328



Wenn die Hardware die Zuarbeit der Software benötigt kann nach Enable ein Interrupt genutzt werden. Die Bedeutung des Interrupts kann im TWINT gelesen werden. Die SCL-Leitung wird solange LOW gehalten bis das TWINT-Register gelöscht wurde.

Da der Prozessor von 2-5V betrieben werden kann, ändern sich die Spannungsschwellen für den I“C-Bus. Ein z.B. mit 3V betriebener Prozessor darf nicht mit 5V-Signalen auf den SDA- und SCL-Leitungen betrieben werden !

I2C-Hardware des PIC16F87/88



8. Anhänge

8.1 Bausteine / Schaltungen für Level-Shifter

Quelle: Bi-directional level shifter for I²C-bus and other systems.
AN97055 NXP

z.B. R_P = 1kOhm

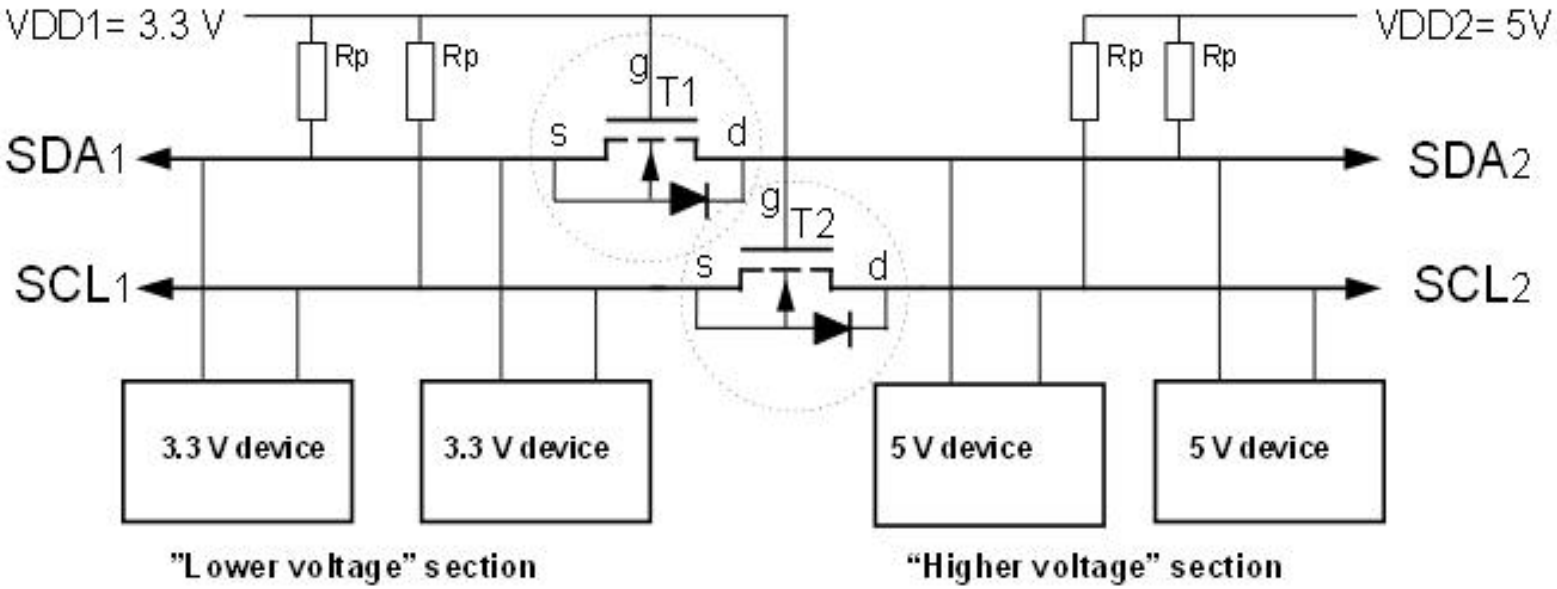
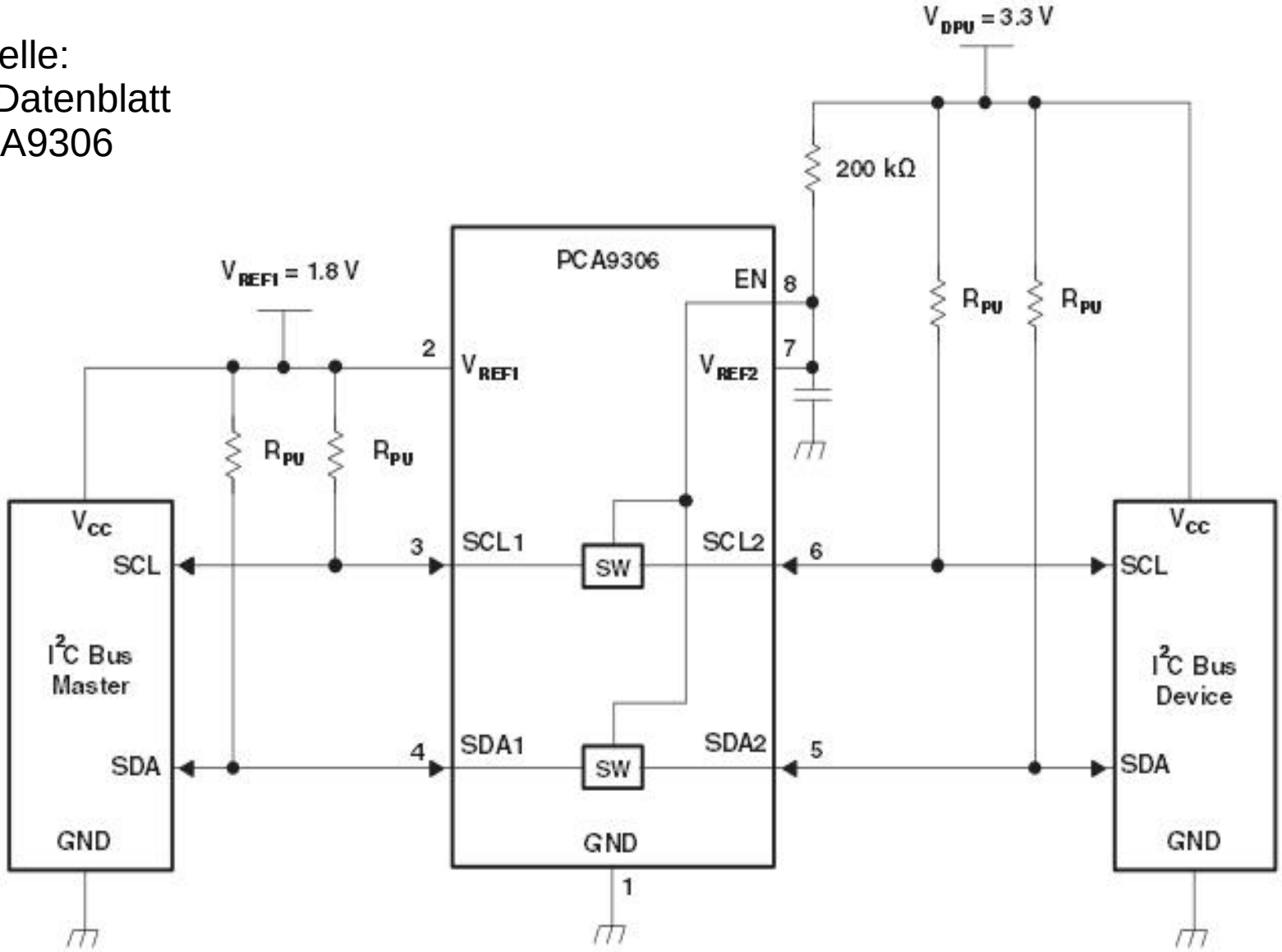


Figure 2. Bi-directional level shifter circuit connects two different voltage sections of an I²C-bus system.

TYPE	$V_{GS(th)}$	$R_{DS(on)}$	C_{iss}	Package
BSN10	min. 0.4V max. 1.8V	25 Ohm (typ)	15 pF	TO-92
BSN20	min. 0.4V max. 1.8V	25 Ohm (typ)	15 pF	SOT23
BSS83	min. 0.1V max. 2.0V	70 Ohm (typ)	1.5 pF (typ)	SOT143
BSS88	min. 0.4V max. 1.2V	15 Ohm	50 pF (typ)	TO-92

Quelle:
TI Datenblatt
PCA9306



Überblick Bausteine Level-Shifter (1)

Type Number	Description
GTL2000	22-bit bi-directional low voltage translator
GTL2002	2-bit bidirectional low voltage translator
GTL2003	8-bit bidirectional low voltage translator
GTL2005	Quad GTL/GTL+ to LVTTL/TTL bidirectional non-latched translator
GTL2010	10-bit bidirectional low voltage translator
NVT2001/02	Bidirectional voltage level translator for open-drain and push-pull applications
NVT2003/04/06	Bidirectional voltage-level translator for open-drain and push-pull applications
NVT2008;	
NVT2010	Bidirectional voltage-level translator for open-drain and push-pull application
P82B96DP	Dual bidirectional bus buffer
P82B96TD	Dual bidirectional bus buffer
PCA9306	Dual bidirectional I2C-bus and SMBus voltage-level translator
PCA9507D	2-wire serial bus extender for HDMI DDC I2C-bus and SMBus
PCA9507DP	2-wire serial bus extender for HDMI DDC I2C-bus and SMBus
PCA9508D	Hot swappable level translating I2C-bus repeater
PCA9508DP	Hot swappable level translating I2C-bus repeater
PCA9509ADP	Low power level translating I2C-bus/SMBus repeater
PCA9509AGM	Low power level translating I2C-bus/SMBus repeater
PCA9509D	Level translating I2C-bus/SMBus repeater

Überblick Bausteine Level-Shifter (2)

Type Number	Description
PCA9509DP	Level translating I2C-bus/SMBus repeater
PCA9509GM	Level translating I2C-bus/SMBus repeater
PCA9512BD	Level shifting hot swappable I²C-bus and SMBus bus buffer
PCA9512BDP	Level shifting hot swappable I²C-bus and SMBus bus buffer
PCA9515AD	I²C-bus repeater
PCA9515ADP	I²C-bus repeater
PCA9516AD	5-channel I2C-bus hub
PCA9516APW	5-channel I2C-bus hub
PCA9517AD	Level translating I²C-bus repeater
PCA9517ADP	Level translating I²C-bus repeater
PCA9518AD	Expandable 5-channel I2C-bus hub
PCA9518APW	Expandable 5-channel I2C-bus hub
PCA9519BS	4-channel level translating I²C-bus/SMBus repeater
PCA9521D	Fast dual bidirectional bus buffer
PCA9521DP	Fast dual bidirectional bus buffer
PCA9522D	Fast dual bidirectional bus buffer with hot insertion logic
PCA9522DP	Fast dual bidirectional bus buffer with hot insertion logic
PCA9527DP	3-channel bidirectional bus extender for HDMI, I2C-bus and SMBus
PCA9600	Dual bidirectional bus buffer
PCA9601	Dual bidirectional bus buffer

Achtung! Repeater und Hubs achten auf den logischen Datenverkehr, das muss nicht zu jeder Anwendung passen !

8.2 Typische Fehlerursachen

Quelle: http://www.i2c-bus.org/i2c-bus/

TWI bedeutet Two-Wire-Interface und für die meisten Anwendungsgebiete ist dieser Bus identisch zum I2C-Bus. .Der Name TWI wurde von ATMEL und anderen Firmen eingeführt, um Konflikte zur Trademark „I^2C“ zu vermeiden. Eine Beschreibung der Fähigkeiten der TWI-Interfaces findet sich in den Datenblättern der Bausteine . Man kann erwarten, dass TWI.Interfaces kompatibel zu I2C sind mit Ausnahme einiger Kleinigkeiten wie z.B. General Broadcaste (Adr. 00h) oder 10bit-Adressierung.

Potentielle Fehlerquellen sind:

- Einige I2C-Master (speziell in reiner Softwareimplementierung) können nicht in Multimaster-Systemen arbeiten und/oder funktionieren nicht wenn Clock-Stretching angefordert wird. Voll ausgeprägte Multimaster-Systeme sind in ihrer Realisierung nicht trivial
- Sehr einfache I2C-Master-Implementierungen können nicht mal Fehlerbedingungen erkennen wie z.B. NoAcknowledge vom Slave.
- Manche I2C-ICs benutzen nicht die festgelegten Spannungsschwellen, wodurch Abhängigkeiten durch die Pull-Up-Widerstände und Leitungskapazitäten auftreten können.
- Manche I2C-ICs haben keine Schmitt-Trigger im Eingang und sind daher empfindlich gegen Störimpulse und Übersprechen aus benachbarten Signalen.
- …….

Quellen

Probleme, Fehler, Erfahrungen
http://www.i2c-bus.org/i2c-bus/

Design calculations for robust I2C communications
http://www.edn.com/design/analog/4371297/Design-calculations-for-robust-I2C-communications

Microchip PICmicro™ Mid-Range MCU Family Reference Manual
http://ww1.microchip.com/downloads/en/devicedoc/33023a.pdf

Microchip PIC16F87/88 Data Sheet
http://ww1.microchip.com/downloads/en/devicedoc/30487c.pdf

Atmel Atmega328 Data Sheet
www.atmel.com/Images/doc8161.pdf