TECHNISCHE HOCHSCHULE DEGGENDORF

IAS Spezielle Protokolle des IoT

Busprotokolle

Motivation

In vielen Anwendungen kommunizieren Komponenten miteinander.

Typische Anforderungen:

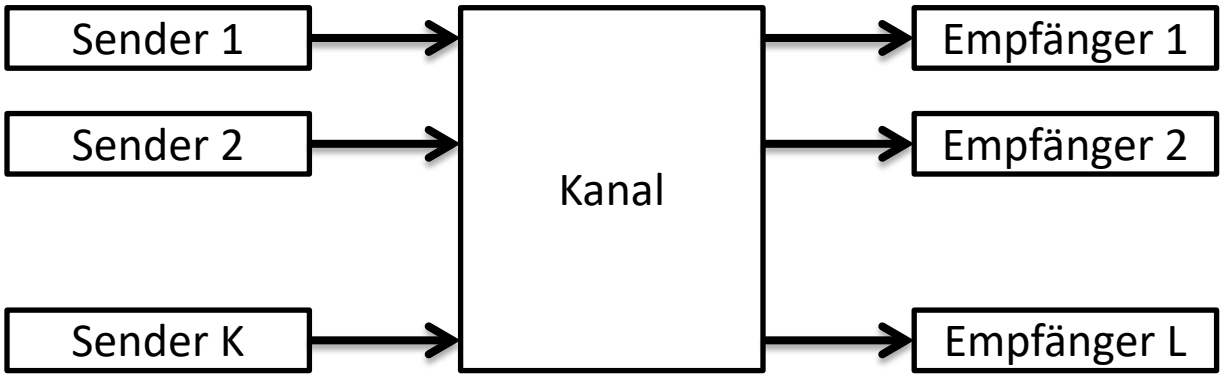
- Aufbau:
  - Wenig Leitungen (falls drahtgebunden)
  - Geringe Bandbreite (falls drahtlos)
  - Geringer Energieverbrauch
  - Große Teilnehmerzahl
- Datenübertragung:
  - Fehlerfrei
  - Geringe Latenz

Kommunikationssytem

Im Allgemeinen wird von Multi-User Kommunikation (Multiple Input / Multiple Output, MIMO) ausgegangen.

Wir unterscheiden:

- **Singlecast: 1 Komponent an 1 Komponte**
- **Multicast: 1 Komponente an eine Auswahl von mehreren Komponenten**
- **Broadcast: 1 Komponente an alle Komponenten**



Physikalische Konfiguration

Der physikalische Aufbau (im Vgl. zum logischen Aufbau) teilt sich auf in:

- Breite des Übertragungskanals:
  - Seriell: Die Daten-Bits werden nacheinander über eine einzige Leitung gesendet, evtl. mit zusätzlicher Takt-Leitung (Clock) zur Synchronisation
  - Parallel: Mehrere Daten-Bits werden auf ebensoviele Leitungen aufgeteilt, typischerweise 8-Bit, evtl. mit zusätzlicher Takt-Leitung
- Topologie und Verschaltung des Übertragungskanals:
  - Matrix / Stern: Alle Komponenten werden an einen zentralen Knotenpunkt angeschlossen, der die Verschaltung der Komponenten übernimmt
  - Ring: Jeder Teilnehmer ist mit dem nächsten Teilnehmer verbunden, der letzte Teilnehmer ist mit dem ersten Teilnehmer verbunden
  - Bus: Alle Teilnehmer verbinden sich mit (einer oder mehreren) durchlaufenden Daten- und Steuerleitungen

Physikalische Konfiguration

Fortsetzung zum physikalischen Aufbau

- Verwendung des Frequenzspektrums:
  - Single-Channel / Single-Frequency: Verwendung eines einzigen Kanals
  - Multi-Channel: Verwendung mehrerer Frequenzbänder parallel

Mehrfachzugriffsstrategien

Wenn mehr als 2 Komponenten sich den Übertragungskanal teilen, muss eine Mehrfachzugriffsstrategie festgelegt werden, die gebräuchlisten sind:

- Frequency-division multiple access (FDMA): Mehrere Frequenzbänder werden genutzt, z.B. sendet jede Komponente auf einem anderen Frequenzband
- Time-division multiple access (TDMA): Jede Komponente sendet in einem zugewiesenen Zeitintervall, z.B. kann Komponente n im Zeitintervall n, dann wieder bei n+K senden
- Code-Division multiple access (CDMA): Jede Komponente sendet mit einer spezifischen Frequenz-Codierungssequenz, die so gewählt werden, dass gleichzeitige Sendungen mehrerer Komponenten unabhängig dekodierbar bleiben (Details in einem späteren Kurs), wird auch als Spread-Spectrum Multiple Access (SSMA) bezeichnet

Bus-Systeme

Im Folgenden liegt der Fokus auf drahtgebundenen Bus-Systemen.

Bei den Komponenten unterscheidet man häufig folgende Rollen:

- Master: Diese Komponente steuert den Buszugriff und stellt Anfragen an andere Komponenten. Als aktive Komponente sendet sie meist zunächst Daten und erwartet dann eine oder mehrere Datenpakete von anderen Komponenten. Der Master ist für die Generierung eventueller Steuersignale verantwortlich (Taktsignal, Select-Signale für die Komponenten, …).
- Slave: Passive Komponente, welche auf ein Signal zur Übertragung wartet und nur auf Aufforderung Daten an den Bus legt. Die vom Master vorgegebenen Steuersignale werden dabei beachtet.

Anmerkung 1: Bei IoT Bus-Protokollen kann die Rollenverteilung zeitvariant sein, d.h. eine Komponente kann für eine bestimmte Zeitdauer seine Rolle von Slave zu Master wechseln.

Anmerkung 2: Bei IoT Bus-Protokollen wird möglichst auf Steuersignale verzichtet.

Bussysteme

Zugriffsverfahren bei kabelgebundenen Busprotokollen:

- TDMA bietet sich an, aber bei fester Zuweisung der Zeitschlitze ergibt sich eine niedrige Anzahl Teilnehmer oder eine niedrige Ausnutzung der Bandbreite
- Alternative:
  - Jede Komponente sendet in Paketen, wenn sie Daten zu senden hat
  - Auf Kollisionen mit anderen Komponenten achten
  - Bei Kollision wird gewartet und erneut gesendet
  - → Da das Versenden der Pakete zu zufälligen Zeitpunkten je nach Verfügbarkeit der Daten geschieht, spricht man von Random Access Methods

Random Access Methods

ALOHA: Einfachste Form der Random Access Methoden

Grundidee:

- Jeder Teilnehmer sendet, sobald Daten vorliegen
- Wenn während des Sendens des Pakets keine anderen Daten auf den Bus gelegt werden, wird von einer erfolgreichen Datenübertragung ausgegangen
- Wenn eine Kollision auftritt, wird eine zufällige Zeit t gewartet. Die Wahrscheinlichkeitsverteilung für diese Zeit t ist:

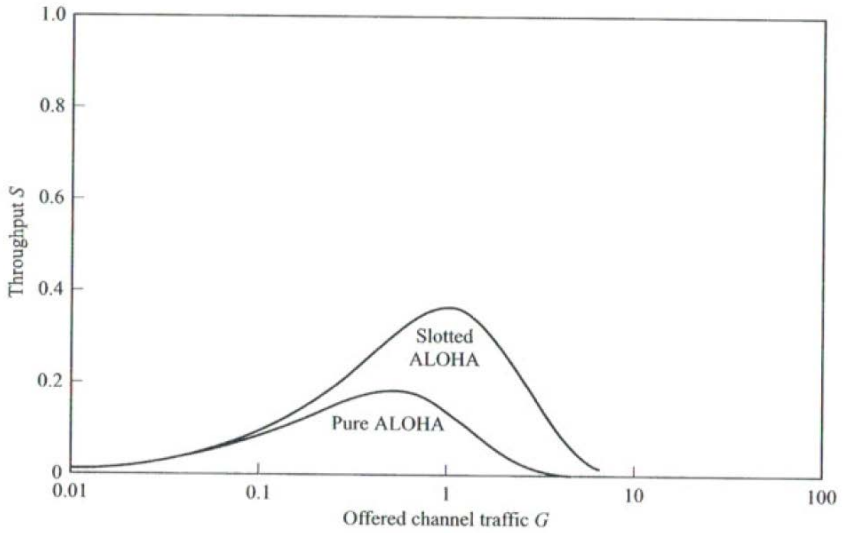
$$p(t) = \alpha \, \mathrm{e}^{-\alpha t}$$

- Der Wert von α bestimmt, wie lang die Verzögerung im Mittel dauert (kleine Werte entsprechen langer Verzögerung) und sollte je nach erwarteter Kanalbelegung gewählt werden

Random Access Methoden

Slotted ALOHA

- Nachteil von ALOHA: Jeder beginnt zu senden, wenn sein Timer abgelaufen ist. Es gibt keinen fest vorgegebenen Zeitschlitz (Slot), in dem ein Paket vor Kollisionen geschützt ist
- Slotted ALOHA definiert Zeitpunkte, zu denen eine Übertragung starten darf
  - Wenn eine Kollision auftritt, wird eine (zufällige) Anzahl Zeitslots gewartet bis erneut eine Übertragung gestartet wird
  - Folge: Verdopplung des maximalen Durchsatzes
  - Aber: Immer noch keine Garantie bei hohem Verkehrsaufkommen
  - Weitere Verbessung: Neue Pakete dürfen erst übertragen werden, wenn alle Kollisionspakete gesendet wurden (Protokoll von Capetanakis 1979)



Random Access Methoden

ALOHA hat zwei Design-Nachteile:

- Eine Komponente fängt mit dem Senden an, egal, ob der Kanal frei ist oder nicht
- Eine Komponente sendet immer das gesamte Paket, egal, ob eine Kollision auftritt oder nicht

In vielen Szenarien können die Komponenten nachsehen, ob der Kanal frei ist oder nicht, dann kommen Carrier Sense Multiple Access with collision detection (CSMA/CD) Protokolle zum Einsatz

- Eine Komponente beginnt nur dann zu senden, wenn der Kanal frei ist
- Wenn zwei oder mehr Komponenten gleichzeitig zu senden beginnen, nachdem sie vorher festgestellt haben, dass der Kanal frei ist, dann sendet die erste Komponente, die die Kollision bemerkt ein JAM Signal, welches allen anderen Komponenten mitteilt, dass eine Kollision stattgefunden hat. Das Senden wird dann mitten im Paket abgebrochen

Random Access Methoden

Wann werden im CSMA/CD die Pakete nach der Kollision erneut gesendet?

- Nonpersistent CSMA: Es wird eine zufällige Zeit gewartet (ähnlich ALOHA)
- p-Persistent CSMA: Es wird zunächst eine Wahrscheinlichkeit p und eine Verzögerungszeit τ festgelegt.
  - (a) Wenn der Kanal frei ist, wird ein Paket mit der Wahrscheinlichkeit p übertragen. Mit der Umkehrwahrscheinlichkeit 1-p wird das Paket um die Zeitdauer τ verzögert
  - (b) Wenn die Zeitdauer τ verstrichen ist (t= τ) und der Kanal frei ist, wird Schritt (a) wiederholt.
  - (c) Wenn die Zeitdauer τ verstrichen ist (t= τ) und der Kanal nicht frei ist, wird solange (aktiv) gewartet, bis der Kanal frei ist und dann bei (a) weiter verfahren

Random Access Methoden

Erweiterung von CSMA/CD bei kurzen Signallaufzeiten

- Bei IoT-Bussystemen müssen Kurzschlüsse vermieden werden, daher wird häufig mit einem prioritären Signalpegel (meist 0V) und einem rezessiven Signalpegel (meist Vcc) gearbeitet.
- Praktisch wird dies mit Open-Kollektor Ausgängen zur Ansteuerung des Busses und Pull-Up Widerständen erreicht
- Daraus folgt
  - (a) Beliebig viele Komponenten können den Bus auf Low setzen, ohne, dass Kurzschlüsse entstehen
  - (b) Ein High-Signal tritt nur dann auf, wenn niemand den Bus auf Low setzt
  - (c) Man kann gleichzeitig den Bus auslesen, um zu erkennen, ob der Bus unerwarteterweise Low ist, obwohl man selbst ein High Signal erwartet
- Daraus ergibt sich eine Collision Detection Möglichkeit ohne eintretende Kollision wie folgt: Wenn mehrere Komponenten gleichzeitig mit dem Schreiben beginnen, können beide gleichzeitig so lange schreiben, wie die Datenbits identisch sind. Beim ersten differerienden Bit erkennt der Sender, der den rezessiven Signalpegel verwendet, dass der Bus von einem anderen Sender verwendet wird, registriert die Kollision und stellt seine Übertragung ein. Der andere Sender kann das Paket ohne Fehler weiter senden.

Praktisches Beispiel I2C

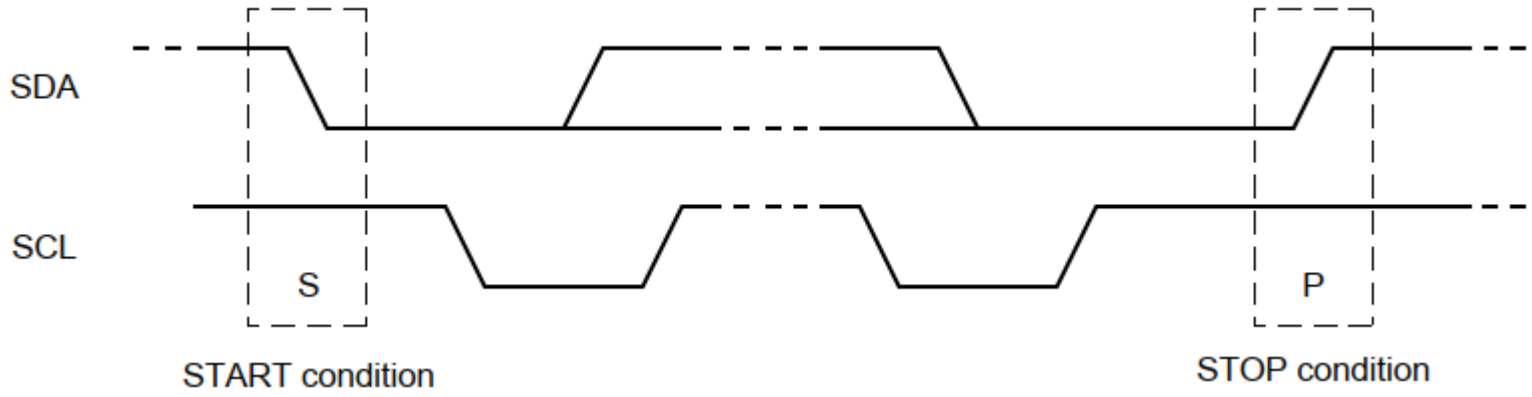
Praktisches Beispiel: Inter Integrated Circuit-Bus (I²C oder I2C)

- Ca. 1980 von Philips entwickelt, erste Spezifikation 1992
- Aktuelle Version: UM10204, Rev. 6, 4. April 2014, NXP Semiconductors
- Verwendet unter anderem in folgenden Architekturen: System Management Bus (SMBus), Power Management Bus (PMBus), Intelligent Platform Management Interface (IPMI), Display Data Channel (DDC), Advanced Telecom Computing Architecture
- 2 aktive Leitungen: Serial Data Line (SDA) und Serial Clock Line (SCL) (plus ggf. GND, VCC)
- Jede Komponente hat eine eindeutige Adresse
- Mehrere Übertragungsgeschwindigkeiten möglich: Standard-mode (100kbit/s), Fast-Mode (400kbit/s), Fast-mode Plus (1Mbit/s), High-Speed Mode (3.4MBit/s), Ultra-Fast Model (5MBit/s)
- Master-Slave Architektur, aber Multi-Master fähig

Praktisches Beispiel I2C

START und STOP definieren, wann ein Master den Bus als Busy belegt

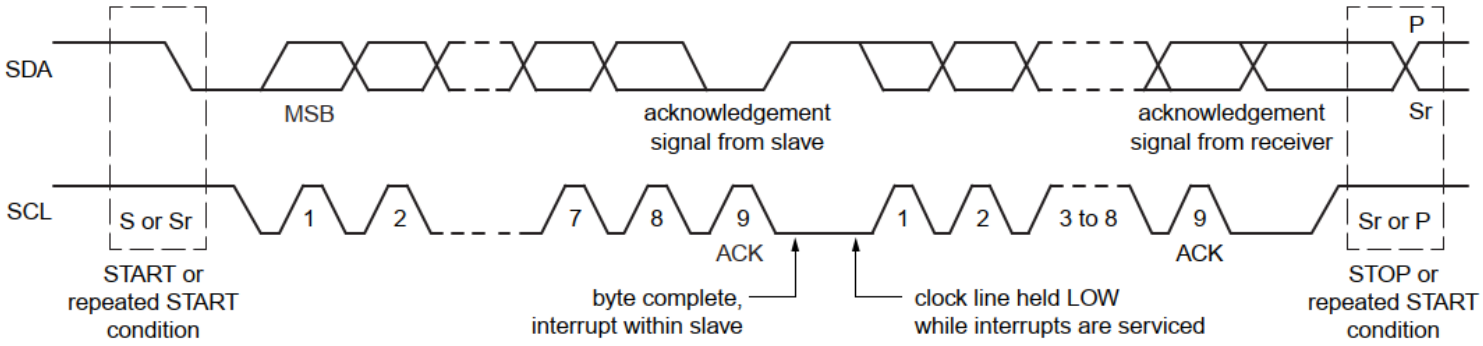
- START: SCL high während SDA eine High-Low-Flanke erhält
- STOP: SCL High, während SDA eine LOW-High-Flanke zeigt
- Nebenbemerkung: Während einer Busy-Phase müssen die Daten bei SCL=high gültig sein und unverändert bleiben, Signalwechsel erfolgen bei SCL=low



Praktisches Beispiel I2C

Datentransfer

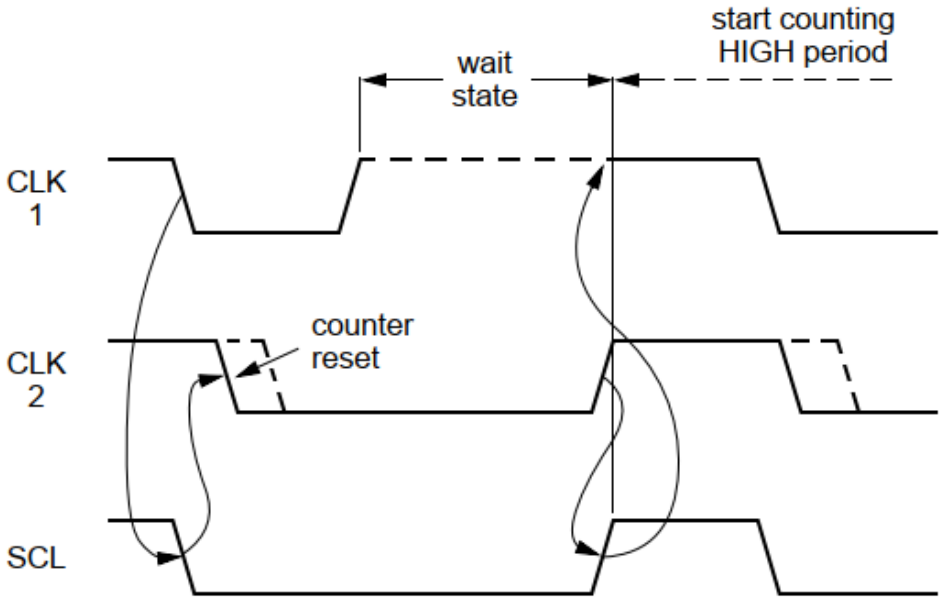
- Daten werden mit dem höchstwertigsten Bit (Most Significant Bit, MSB) zuerst übertragen
- Der Master generiert die Clock Leitung, aber wenn der Slave noch nicht bereit ist, kann er die Clock Leitung auf Low ziehen bis er bereit zum Senden ist
- Es wird von der jeweiligen Gegenseite ein Acknowledgement für jedes Byte gesendet, indem SDA auf Low gezogen wird



Praktisches Beispiel I2C

Kollisionsvermeidung bei 2 Mastern

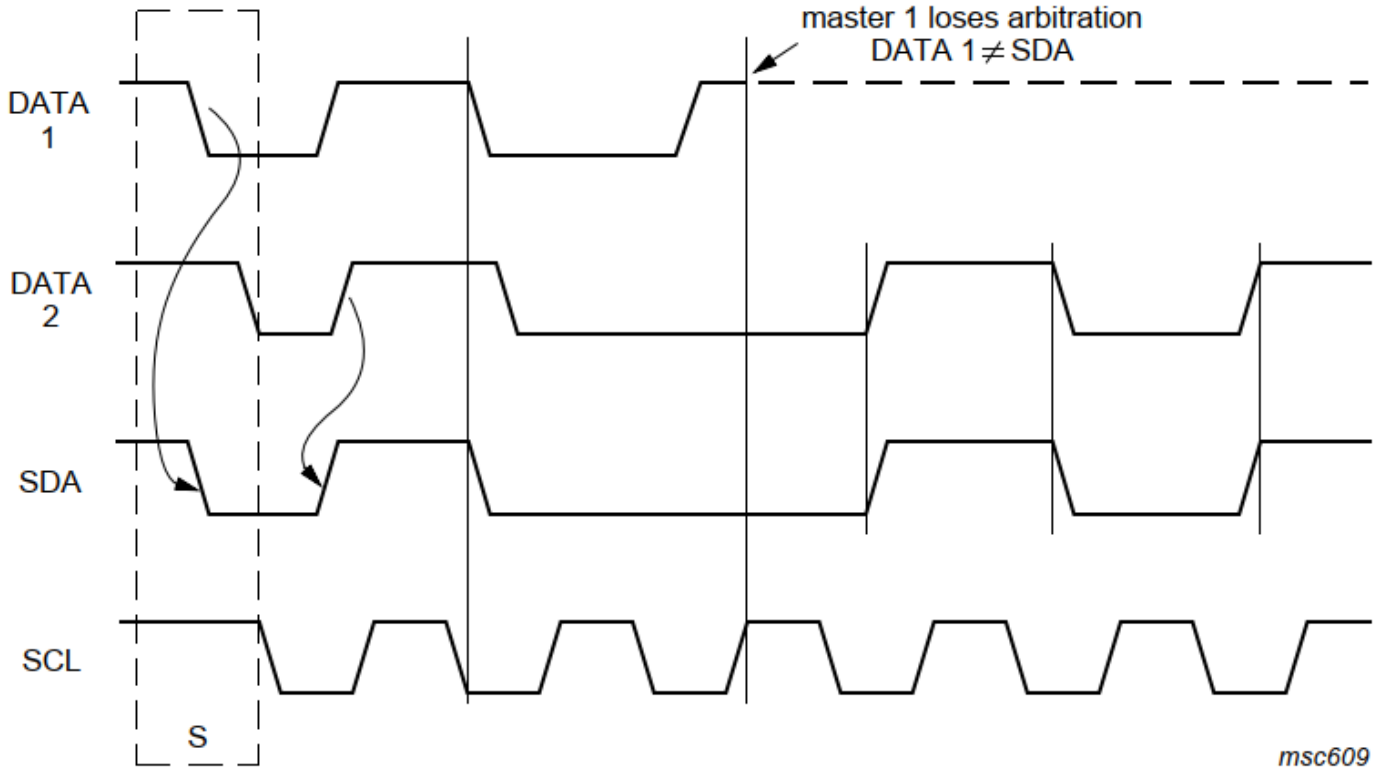
- Mehrere Master müssen sich auf eine gemeinsame Low/High Periode für die Clock einigen (Clock-Synchronisation)
  - Dazu ziehen alle die SCL zunächst auf Low und geben sie nach ihrer Clock-Low-Periode wieder frei. Danach zählen sie, bis das Signal auf High zurückkehrt
  - Alle Master starten von dort aus mit ihrer Clock-Rechnung. Der Master, der als erstes die High-Periode beendet, zieht die Clock-Leitung auf Low.



Praktisches Beispiel I2C

Kollisionsvermeidung bei 2 Mastern

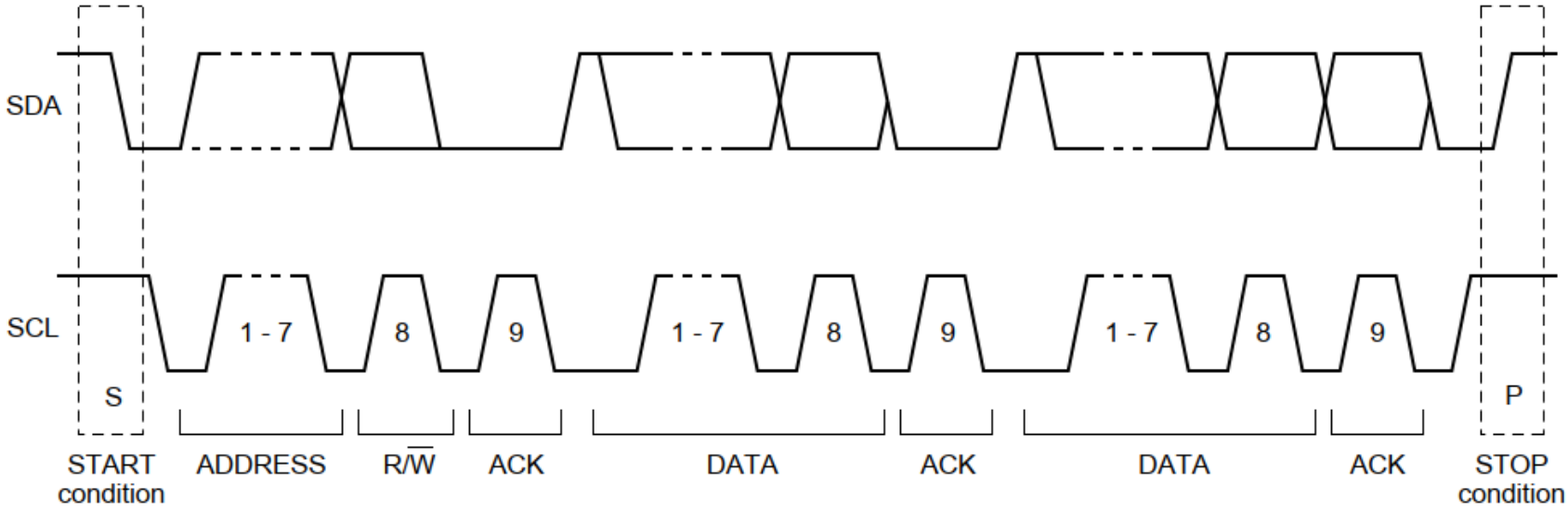
- Wie bereits beschrieben, können 2 Master gleichzeitig senden, prüfen aber den Bus und wenn ein Master feststellt, dass er ein High sendet, während der Bus Low ist, erkennt er eine Kollisiono und zieht sich für diese Übertragung zurück



Praktisches Beispiel I2C

Übertragung von Adresse und Lese / Schreiboperation

- Nach der Start-Condition wird ein Byte gesendet, das aus 7 Bit Adresse und 1 Bit Read/Write Indikation besteht
- Bei Write schreibt der Master weiter auf den Bus, bei Read schreibt der Slave auf den Bus
- Das Acknowledgement kommt vom jeweils anderen Partner, der Master beendet die Übertragung mit einem Not-Acknowledge vor der STOP Condition



Praktisches Beispiel I2C

Weitere Details

- Adressierung von 10 Bit Adressen
- Adressierung aller Devices (Broadcast) durch Geräteadresse 0
- Reservierte Adressen (0000 XXX und 1111 XXX)
- Software-basierte Zuweisung von Adress-Teilen an Slaves
- Transfer in höheren Bandbreiten-Modi
- ...

Praktisches Beispiel I2C

TECHNISCHE HOCHSCHULE DEGGENDORF

Danke für Ihre Aufmerksamkeit!