I2C- Grundlagen

Anfang der 80er Jahre entwickelte die Firma Philips das I2C-Busprotokoll. Ausschlaggebend war der steigende Einsatz von Mikroprozessoren in Konsumergeräten. Gerade die Elektronikbranche ist darauf angewiesen, möglichst preiswert zu produzieren. Einsparungen waren somit dringend notwendig. Und genau hier setzt der I2C-Bus an. Bisher wurden teure CPU's mit 8- oder 16-Bit breiten Datenbussen in z.B. Fernsehgerät eingesetzt, obwohl sich der Datenfluss fast auf die Interaktion des Konsumers beschränkt, der gelegentlich den Sender wechselt.
Aber warum soll nun der Einsatz eines neuen Bussystems Kosten einsparen ? Ganz einfach: Beschränkt man die Datenübertragung auf ein serielles Verfahren, fallen immens viele Leitungen weg. Die IC's können nun auch kleiner produziert werden, da nicht mehr so viele Anschlüsse nach aussen geführt werden müssen. Auch der Platzbedarf auf der Platine verringert sich stark. Zudem müssen weniger Platinenbohrungen und Lötpunkte angebracht werden. Addiert man die ganzen Einsparungen, sollte der ernorme Platzgewinn und der geringere Fertigungsaufwand erkennbar sein.
I2C steht für "Inter Integrated Circuit". Und genau dort wird er auch eingesetzt, als Bus zwischen Mikroprozessoren, IC's und Ein-/Ausgabeeinheiten jeglicher Art, die in der Regel ihren Platz auf einer gemeinsamen Platine finden.
Die Daten werden mit Hilfe zweier Leitungen als serielle Information übertragen. Vorraussetzung ist nur noch ein gemeinsames Massepotential, was nur in seltenen Fällen zu einer dritten Leitung führt.

Konzept

Die gesamte Kommunikation wird über nur zwei Leitungen abgewickelt. SCL (Serial Clock) trägt das notwendige Taktsignal und SDA (Serial Data) die Information. Im I2C-Bus erfolgt die Datenübertragung bitseriell und synchron, d.h., dass jedes Datenbit auf der SDA-Leitung mit dem Takt der SCL-Leitung synchronisiert wird. Beide Leitungen sind bidirektional ausgeführt, was es jedem IC ermöglicht, Daten zu verschicken oder anzufordern. Um zu gewährleisten, dass die Information zum richtigen Empfänger gelangt, erhält jedes am Bus beteiligte IC eine eindeutige Adresse, mit der es identifiziert werden kann.
Das Konzept sieht eine Unterscheidung von Geräten (IC's) in Master und Slave vor. Als Master wird ein Gerät bezeichnet, das einen Datentransfer initiiert und das notwendige Clocksignal generiert. Geräte, die vom Master adressiert, also zum Transfer aufgefordert werden, bezeichnet man als Slave. Desweiteren wird noch eine Unterscheidung in Transmitter und Receiver gemacht. Der Transmitter sendet Daten und der Receiver empfängt sie. Anhand zweier Beispiele soll dieses nun verdeutlicht werden:

Mikroprozessor (MP) A sendet Daten an MP B

- A (Master) adressiert B (Slave)
- A (Master-Transmitter) sendet Daten an B (Slave-Receiver)
- A beendet den Transfer

Mikroprozessor (MP) A fordert Daten von MP B an

- A (Master) adressiert B (Slave)
- A (Master-Receiver) empfängt Daten von B (Slave-Transmitter)
- A beendet den Transfer

Prinzipiell ist jedes Gerät in der Lage, Master- bzw. Slave-Funktionen zu übernehmen, da die Zuordnung dynamisch erfolgt und keinen Regeln unterliegt.
Da beliebig viele IC's am Bus einen Transfer initiieren dürfen, wird er auch als Multimaster-Bus bezeichnet. Es muss daher gewährleistet sein, dass nur ein Master zur Zeit aktiv ist. Um Problemen vorzubeugen, bietet das I2C-Protokoll mehrere Massnahmen zur Vermeidung von Kollisionen, auf die im weiteren Verlauf näher eingegangen werden soll.

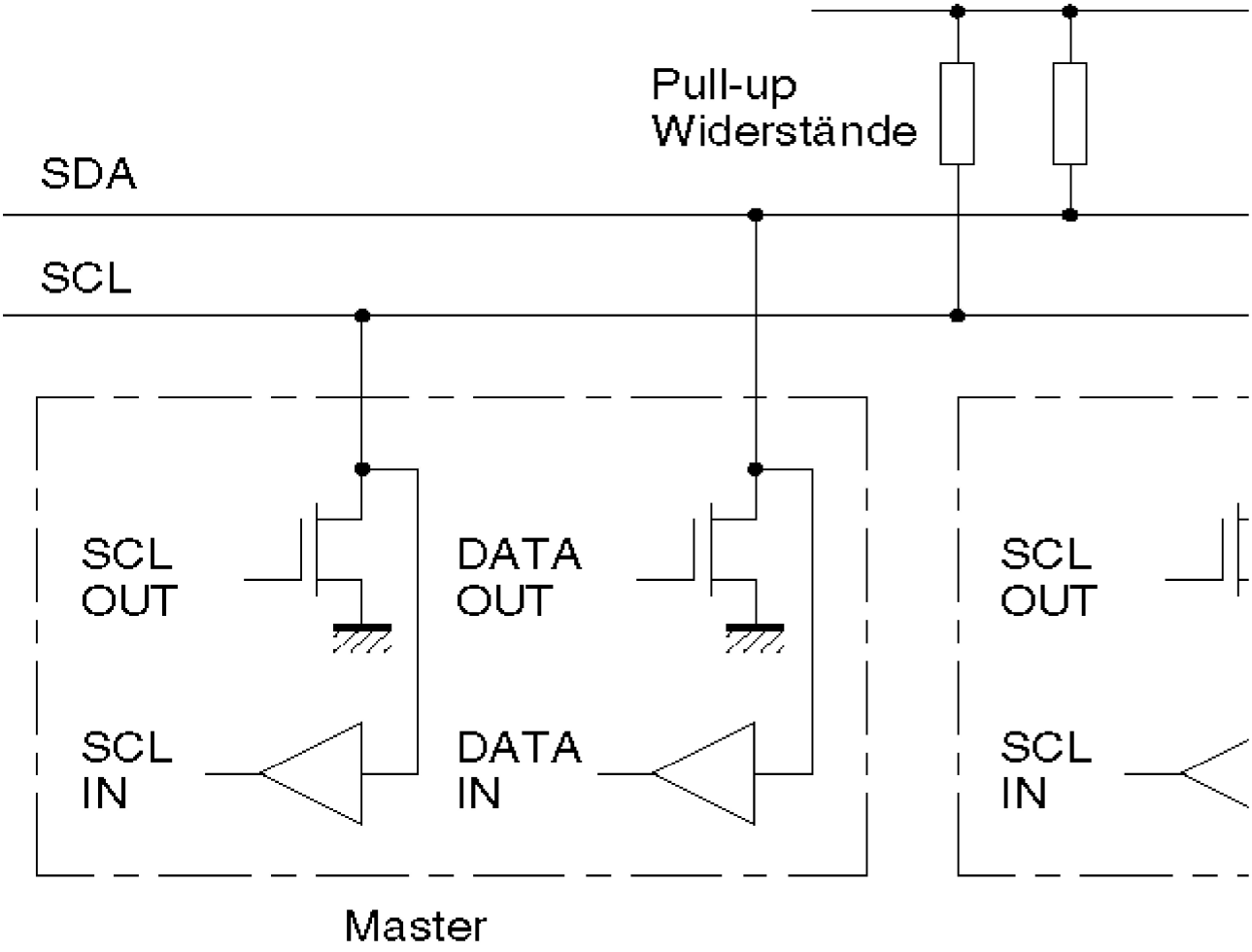
Elektrische Eigenschaften

- SCL (Serial Clock) und SDA (Serial Data) arbeiten bidirektional und werden durch Pullupwidestände auf High-Potential (logisch "1") gezogen
- Transferraten:
  - Low Speed Mode 10 kBit/s (inzwischen bedeutungslos)
  - Standard Mode 100 kBit/s
  - Fast Mode 400 kBit/s
  - High Speed Mode 3,4 MBit/s (1998 eingeführt; keine oder kaum IC's, die diesen Standard unterstützen)
- Beschränkung der maximalen Buskapazität auf 400 pF
- kompatibel zu den gängigen Logikfamilien

Hardware-Grundlagen

Wie bereits beschrieben, besteht der I2C-Bus aus zwei aktiven Leitungen SDA und SCL sowie der Masseleitung als Bezugspotential. Die Leitungen SDA und SCL sind über Pull-up-Widerstände mit der Betriebsspannung verbunden.
Im Ruhezustand befinden sich also beide Leitungen auf High-Pegel (positive Logik). Die unten stehende Abbildung zeigt schematisch die Eingangs- und Ausgangskreise von Master- und Slave-Bausteinen. Bezeichnet wird die Gesamtheit dieser Chips als Wired-AND-Schaltung. Im Ruhezustand sind die als Open-Collector ausgeführen Transistoren gesperrt und der Bus befindet sich auf High-Pegel.
Zur Ausgabe einer logischen 0 wird der Transistor leitend und zieht die entsprechende Leitung auf Masse. Im umgekehrten Fall bleibt die Basis des Transitors stromlos und die Leitung liegt,

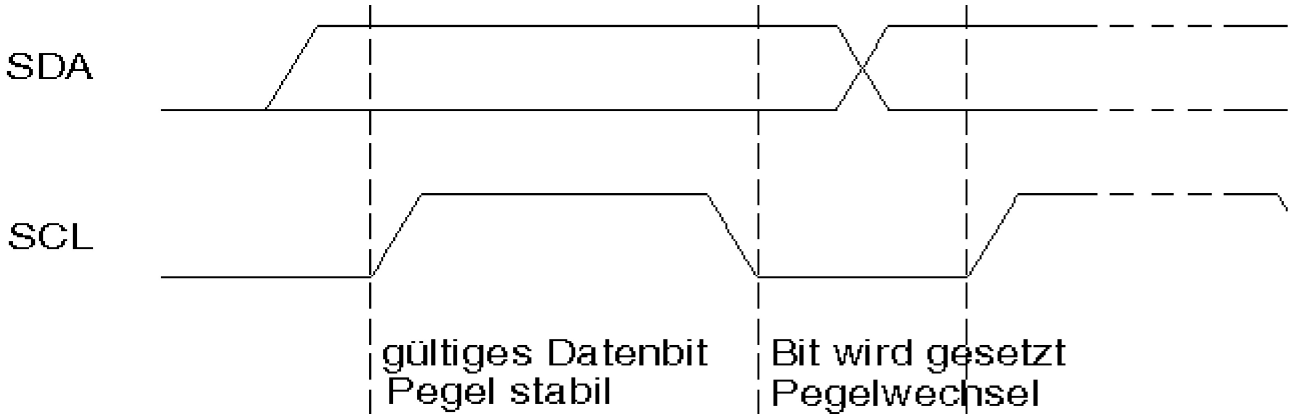
bedingt durch den Pull-up-Widerstand auf logisch 1. Die Eingangskreise erkennen die Pegeländerungen auf den Busleitungen und können sie dann entprechend auswerten.
Der High-Pegel ist definiert als NICHT-LOW-Pegel, d.h., es kann auch mit entsprechenden Bausteinen an einem System mit Spannung ungleich 5V (z.B. 3,3V) gearbeitet werden.



Datenübertragung

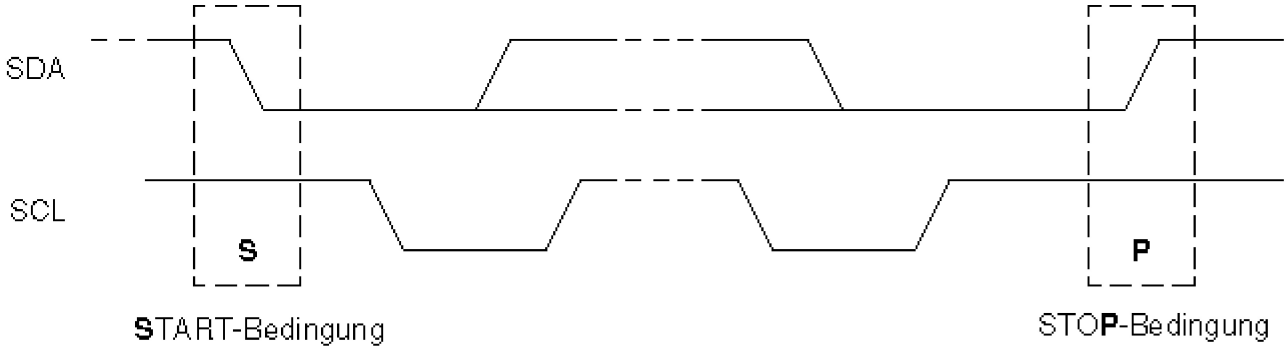
Um die Kompatibiltät zu den Mikroprozessorsystemen zu wahren, werden die Daten im 8-Bit-Format übertragen. Die Anzahl der zu übertragenden Bytes in einem Transfer ist dank der SCL-Leitung, die das Clocksignal zur Synchronisation trägt, unbegrenzt. Allerdings können einige Bausteine abhängig von ihrer Funktion nur eine bestimmte Anzahl Bytes senden und empfangen. Das Byte wird mit dem höchstwertigen Bit MSB (Most Significant Bit) voran übermittelt. Wichtig ist, dass die Daten auf der SDA-Leitung stabil sein müssen, solange die SCL-Leitung High führt, ansonsten werden die Signale als Steuersignale interpretiert. Schreibende Geräte (Transmitter) müssen ihre Daten auf die Leitung legen, wenn SCL Low ist. So erreichen sie einen stabilen Zustand in der High-Phase. Nun können die lesenden Geräte die Daten bei SCL High aufnehmen.
Im Timingdiagramm kann man gut die Übertragung eines gültigen Datenbits erkennen. Nun wird auch schon deutlich, wie simpel und dennoch effizient das I2C-Protokoll arbeitet. Ist SCL

High, wird die Datenleitung SDA abgefragt und das stabil anliegende Signal als Information interpretiert.

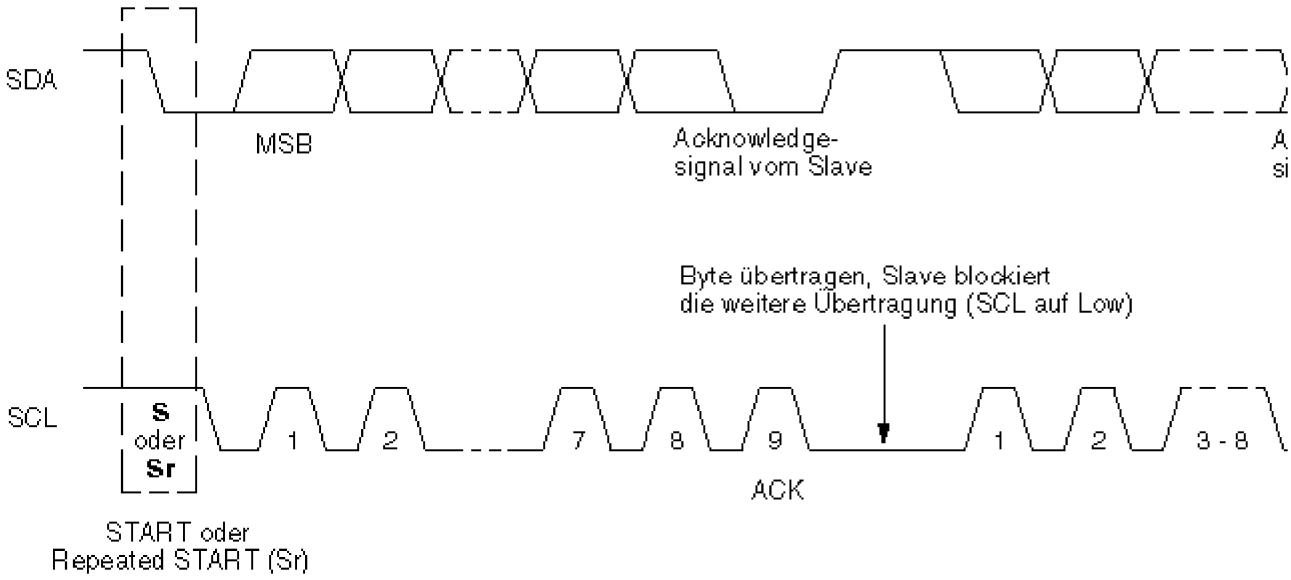


Steuersignale

Leider ist es mit der reinen Datenübertragung noch nicht getan, vielmehr muss der gesamte Ablauf noch koordiniert werden. Damit nicht jedes anliegende Gerät zu Senden oder Empfangen beginnt, wird eine Identifikation der IC's in Form einer Adresse den Daten vorangestellt. Desweiteren wird z.B. der Beginn und das Ende einer Übertragung markiert. Diese wichtigen Signale, die das ganze Verhalten auf den Leitungen bestimmen, werden zusammengefasst als Steuersignale bezeichnet.
Doch wie unterscheidet man nun die Daten- von den Steuersignalen ?
Einige lassen sich einfach an der Reihenfolge erkennen. Sinnigerweise muss erst die Adresse mitgeteilt werden und dann können die Daten folgen. Doch es gibt noch eine andere Möglichkeit. Man verändert die Signale auf der Datenleitung. Um nun nicht mit den Regeln zu brechen, die unter anderem besagen, dass alles, was im High-Zustand von SCL als Pegel auf SDA liegt, ein Datenbit ist, hat man sich nun noch etwas Besonderes einfallen lassen. Bestimmte Steuersignale werden anhand ihrer Pegeländerung während der High-Phase der Clockleitung identifiziert. Von den Receivern wird nun nicht mehr der Pegel, sondern die Flanke gelesen. Elementar sind die Start- und Stopbedingung, die eine zu übermittelnde Sequenz kennzeichnen. Das Startsignal wird durch eine fallende Flanke auf SDA herbeigeführt. Wichtig ist, dass während des Flankenwechsels SCL High führen muss. Die Stopbedingung wird analog kodiert, nur dass sie eine steigende Flanke darstellt. Start- und Stopbedingung dürfen zu jedem Zeitpunkt eines laufenden Transfers erzeugt werden. Beide Signale werden vom Master erzeugt. Nach jedem Start wird der Bus als Busy (Belegt) angesehen. Nur der Master, der den Transfer initiiert hat, darf den Bus wieder auf Idle (Frei) setzen.
Eine Besonderheit ist die Möglichkeit einer Repeated Start Condition (erneute Startbedingung), hier wird der Transfer nicht durch ein Start beendet. Es wird vielmehr durch einen wiederholten Start ein neuer Transfers eingeleitet. Man verspricht sich so einen leicht erhöhten Datendurchsatz, da das Stopsignal unterdrückt wird.



Ist das Byte übermittelt, wird häufig ein optionales Handshaking vollzogen. Der Slave muss nun nach jedem empfangenen Byte ein Acknowledge an den Master senden. Antwortet der Slave auf dieses Bestätigungssignal mit einem High-Bit, gibt es zu erkennen, dass es an der Kommunikation nicht mehr teilnimmt, bzw. diese beendet.
Will nun ein Gerät einen Slave auslesen, z.B. einen Speicher, muss nun der Master nach jedem Byte das Acknowledge senden. Ist die Kapazität des Speichers überschritten, kann der Master nach Empfangen des High-Pegels eine entsprechende Fehlerroutine starten. Es ist jedoch zu beachten, dass kein Master weitersenden darf, wenn er ein High-Acknowledge-Bit, auch als NACK (Not Acknoledge) bezeichnet, empfangen hat. Ansonsten gehen sämtliche nachträglich gesendete Informationen verloren.
Auch wenn das Acknoledge-Bit ein Steuersignal ist, wird es praktisch wie ein Datenbit gehandhabt, der Master erzeugt hierfür noch ein zusätzlichen neunten Taktimpuls auf SCL. Der einzige Unterschied ist, dass sich das Verhalten von Transmitter und Receiver kurzfristig umdreht, schliesslich muss der Slave die folgende Übertragung mit einem Low quittieren, wenn die Übertragung fortgesetzt werden soll. Sendet der Slave dagegen ein High, gibt er das Ende der Kommunikation bekannt.

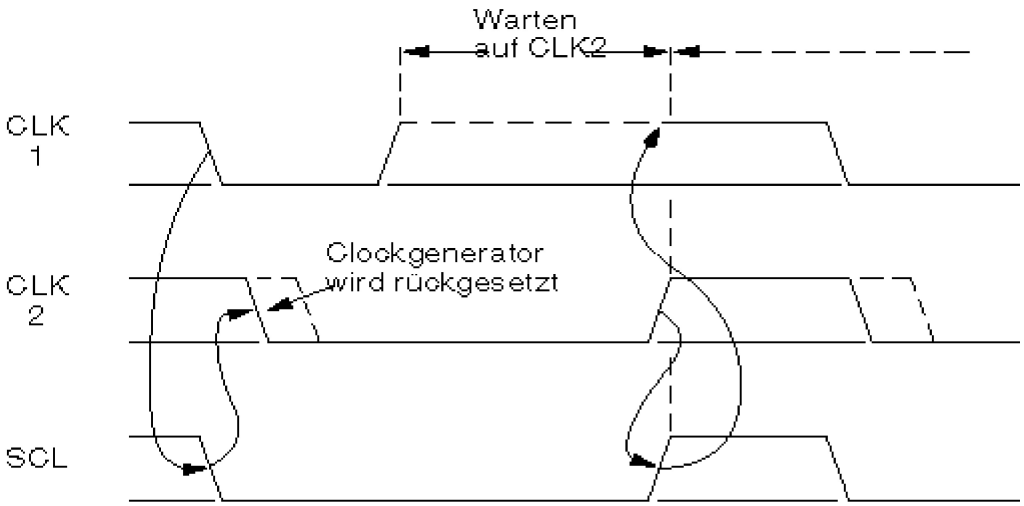


Interessant für langsamere Bausteine ist noch das Clock-Stretching (Taktverlängerung). Slaves können, bedingt durch die Pullupwiderstände an den Datenleitungen, die Lowphase des Clocksignals künstlich verlängern. Damit geben sie dem Master zu verstehen, dass sie die

Taktfrequenz nicht verarbeiten können und ermöglichen sich selbst so das korrekte Lesen des aktuellen Datenbits.
Notwendig ist hierbei die Überwachung der Taktleitung durch den Master, da dieser sonst nicht auf das Clock-Stretching reagieren kann.

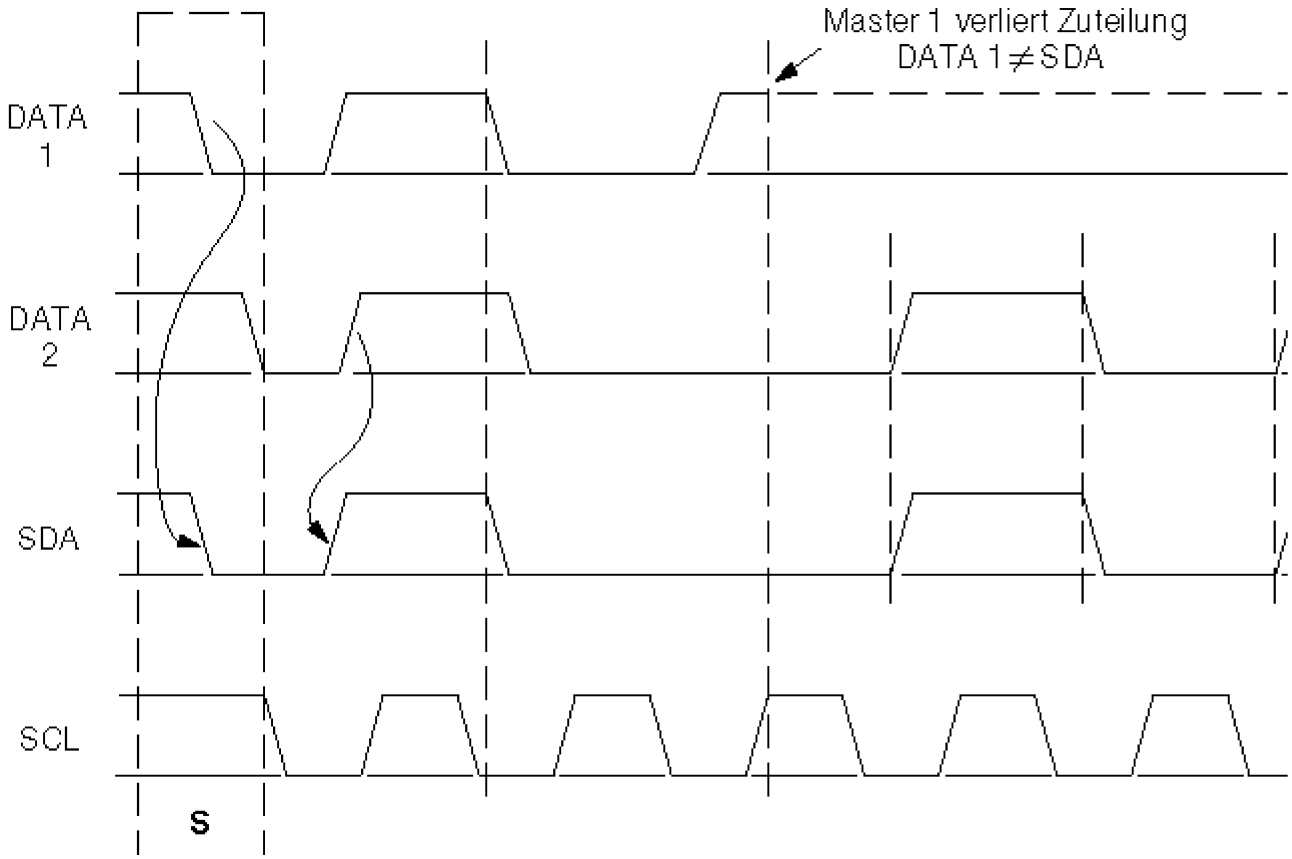
Clock-Synchronisation und Arbitrierung

In Multimastersystemen wird unter bestimmten Bedingungen vorausgesetzt, dass das Taktsignal der Master sich völlig synchron verhält. Dies ist nur dann möglich, wenn die Master eine gewisse Eigenintelligenz mitbringen, die man in der Regel nur in Mirkroprozessoren findet. Synchronisiert man zwei Master, stimmen danach Frequenz und Tastverhältnis überein. Ein Kompromiss aus beiden Clocksignalen ist durchaus vorstellbar, wenn beide Systeme zur gleichen Zeit eine Synchronisation durchführen.



Durch die Realisierung der Busleitung als Wired-AND besteht die Möglichkeit, das anliegende Taktsignal zu lesen. Ein Master, der auf der SCL-Leitung einen Low-Pegel ausgibt, wird diesen irgendwann zurücknehmen und High ausgeben. Nun schaut der Master auf die Clockleitung. Liegt nun immer noch eine logische “0” an, bedeutet dies, dass ein weiterer Master eine längere Low-Phase erzeugt. Anhand der Beobachtung von SCL kann der erste Master sein Taktsignal solange verlängern, bis es mit dem langsameren übereinstimmt.
Analog dazu kann auch der Beginn des Clocksignals verschoben werden. Legt ein Master die SCL-Leitung auf logisch “1” und ein anderer, viel langsamerer, hält sie nach wie vor auf “0”, behält der weniger schnelle Master die Dominanz. Wird SCL auch schon in dieser Zeit überprüft, kann sich der schnellere Master auf die längere Low-Phase einstellen.
Allgemein setzt sich die langsamste Low-Periode und damit das langsamste Gerät durch.
Die Zuteilung des Busses an die verschiedenen Master wird als Arbitrierung bezeichnet. Implementiert ist ein Verfahren mit dem Namen CSMA/CD (Carrier Sense, Multiple Access with Collision Avoidance). Sobald ein Master eine Transaktion auf dem Bus durchführen will, muss kontrolliert werden, ob er überhaupt frei ist oder eine Übertragung noch nicht abgeschlossen. Das Verfahren ist relativ leicht umzusetzen, da lediglich die beiden Leitungen

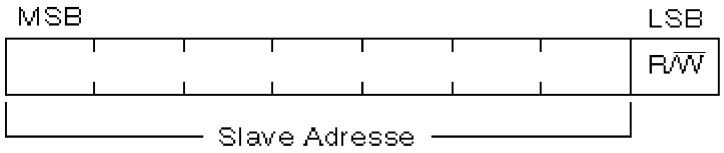
SDA und SCL für eine definierte Zeit, die etwas unter der halben Periode des Taktsignals liegt, beobachtet werden. Trägt nur eine Leitung ein Low-Signal, ist dies ein sicheres Zeichen für eine bestehende Transaktion, der Bus wird nun als belegt angesehen.
Erkennt der Master den Bus als frei, kann er seine Übertragung beginnen. Gelegentlich kann es dennoch zu einer gleichzeitigen Transaktion kommen, z.B. dann, wenn zwei oder mehr Master im gleichen Moment den Bus abfragen und ihn als frei interpretieren. Interessanterweise kommt es in solchen Situationen erst zu einer im oberen Teil beschriebenen Synchronisation des Taktes. Doch damit nun nicht beide Master gleichzeitig senden, setzt die sogenannte Kollisionsvermeidung (Collision Avoidance) ein. Sobald einer der Master eine logische “1” sendet, also die Leitung auf High setzt, aber eine “0” empfängt, erkennt er, dass zur Zeit ein anderes Gerät den Bus bedient, und bricht die Transaktion ab. Die “0” kann nur dann empfangen werden, wenn zur gleichen Zeit ein anderes IC die Datenleitung für sich beansprucht und den entsprechenden Pull-up-Widerstand auf Masse legt.
Nachteilig an diesem Verfahren ist, dass keine Prioritäten verteilt werden können. Es ist somit dem Zufall bzw. der Übertragung überlassen, welcher Master die endgültige Kontrolle über den Bus erlangen kann. Als Kollisionsvermeidung wird das Verfahren benannt, da der Master, der sich durchsetzt, von seinen Konkurennten nichts mitbekommt.



Adressierung

Will nun ein Master mit einem bestimmten Gerät über den Bus kommunizieren, muss als erstes eine Adresse übermittelt werden, die das IC eindeutig identifiziert. Die logische Konsequenz der

Adresse ist, dass sie einmalig und dem Master bekannt sein muss.
Im Normalfall ist der Master das steuernde Gerät auf dem Bus, er allein kann einen Transfer initiieren. Herrschen mehrere Master über den Bus, ist eine Adressierung der einzelnen Master unabdingbar. Ansonsten ist eine vollständige Kommunikation zwischen den Mastern nicht möglich.
Die ursprüngliche Grösse der Adresse beträgt 7 Bit, damit liessen sich im Höchstfall 128 IC's im Bus einsetzen. Da allerdings noch einige Adressen reserviert sind und nicht jedes Gerät jede beliebige Adresse annehmen kann, wurde eine Erweiterung auf eine Länge von 10 Bit spezifiziert. Nun bleibt es dem Anwender überlassen, für welches Format er sich entscheidet, nur gibt der Markt zur Zeit kaum bis keine IC's mit 10 Bit langen Adressen her.



Nach der gesendeten Startbedingung wird unmittelbar die Adresse übertragen. Im 7-Bit-Format, auf welches hier näher eingegangen werden soll, wird die Adresse in dem nächsten Byte angegeben. Da hier ein Bit übrig bleibt, das LSB, wird hier noch zusätzlich die Richtung des Transfers (R/W) verschlüsselt (Abb. 7). Eine logische "0" bedeutet "Schreiben", eine "1" steht für den Lesebetrieb. Hier entscheidet sich die Funktion von Master und Slave, ob das Gerät nun als Receiver (Lesen) oder Transmitter (Schreiben) arbeitet.
Ist nun die Adresse übertragen worden, wartet der Master auf eine Bestätigung. Hat ein Slave die 8 Bit empfangen, vergleicht er sie mit seiner eigenen Adresse. Bei Misserfolg verhalten sie sich ruhig und warten auf die nächste Startbedingung. Stimmt die Adresse überein, meldet sich der Baustein mit einem Acknowledge und gibt zu erkennen, dass er bereit für die folgende Übertragung ist.
Die Art der Adresskodierung der Bausteine ist recht vielfältig. Es gibt zum einen reine Softwarelösungen, die aber nur in Mikroprozessoren realisierbar sind. In der Regel stehen externe Pins zur Verfügung, die über Jumper gesetzt werden können. Um die Bausteine pysikalisch klein und billig zu halten, werden häufig nur die niedrigsten drei Bits der Adresse nach aussen zur Konfiguration geführt. Die restlichen vier Bits sind dann fest im Gehäuse verdrahtet und somit nicht zugänglich, was den verfügbaren Adressraum noch kleiner werden lässt.
Auf die reservierten Adressen soll nur kurz eingegangen werden, da sie nur in seltenen Fällen von Belang sind.
Es wurde in den vorherigen Abschnitten behauptet, dass eine Übertragung immer nur zwischen zwei IC's möglich ist. Aber es gibt eine Ausnahme. Mit der General Call Adress, Adresse "0" und Richtungsbit auf "Schreiben" (R/W = 0), lässt sich eine Art Rundruf gestalten. Ein Master kann so einer ganzen Gruppe von Slaves Daten übermitteln.
Es wurde auch an die Kompatibiltät zu anderen synchronen Bussystemen gedacht. Um diese nutzen zu können, gibt es z.B. die Adressen 0000 001 und 0000 010, sie würden dann die

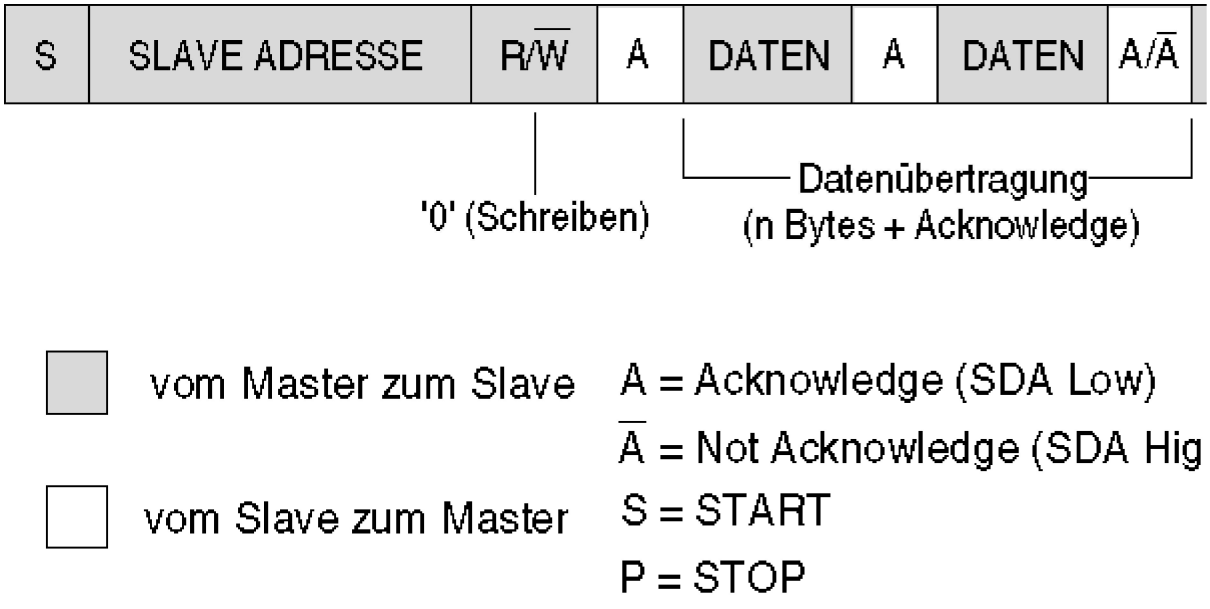
differenten Systeme aktivieren.
Eine wichtige Adresse ist die 10-Bit-Slave-Adresse, die wie der Name schon sagt, eine 10 Bit lange Adresse einleitet. Hier sind nur die höherwertigen fünf Bits reserviert, die restlichen enthalten schon die ersten zwei Bits der eigentlichen Adresse und natürlich das Richtungsbit.

SLAVE ADDRESS	R/$\overline{W}$ 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

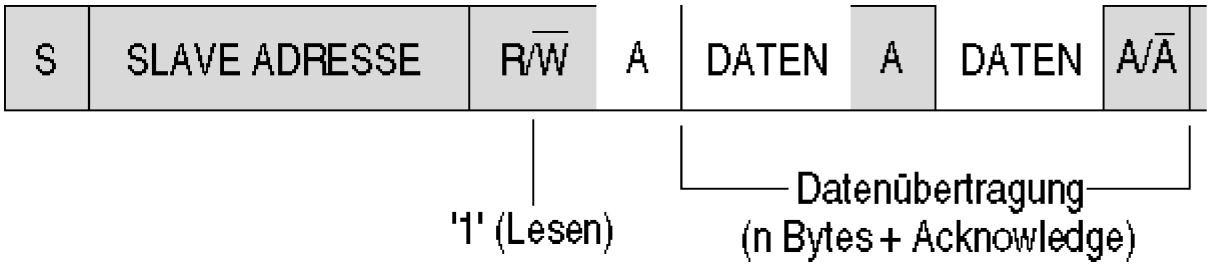
Transferarten

Prinzipiell gibt es drei verschiedene Arten des Transfers. Sie geben die Richtung der zu übertragenden Daten an. Um einen kleinen Überblick zu gewähren, sind die Betriebsarten noch einmal aufgezeichnet.

- Master sendet Daten an Slave
  Die Richtung des Transfers bleibt erhalten, Daten werden vom Master auf den Bus gebracht. Der Slave antwortet nur, wenn überhaupt gewünscht, mit einem Acknowledge.



- Master fordert Daten vom Slave an
  Hier wechselt die Transferrichtung. Der Slave sendet die angeforderten Daten und der Master quittiert gegebenenfalls mit einem Acknowledge. Beendet wird die Verbindung dennoch vom Master.



- Kombiniertes Format
  Nach einer Repeated-Start-Bedingung erfolgt ein Richtungswechsel von anfangs schreibend auf lesend, aus Sicht des Masters.