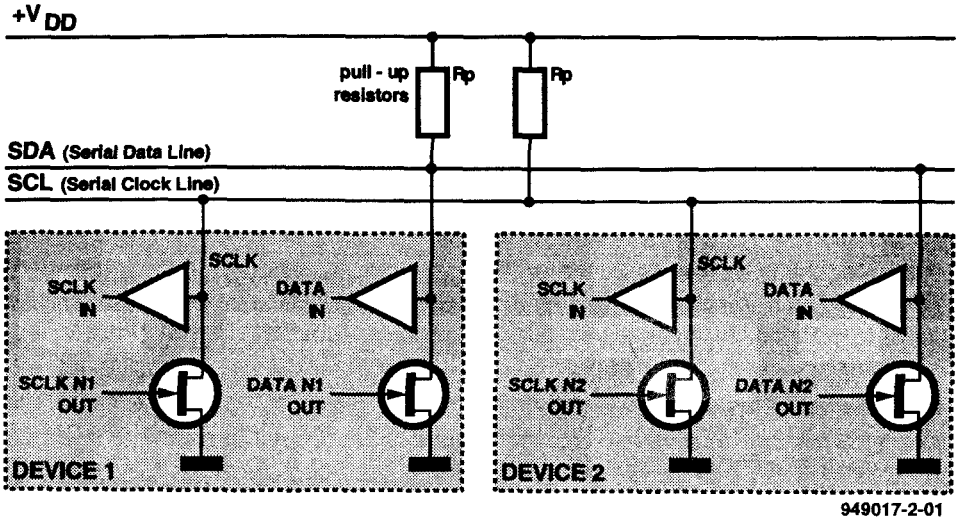
Specificatie van de I²C –bus. *Het I²C-concept*

De I²C-bus ondersteunt IC's van de logische families. Twee leidingen, SDA (Serial DAta) en SCL (Serial CLock), verzorgen de uitwisseling van informatie. Iedere busdeelnemer heeft een exclusief, uniek adres. Daarbij is het om het even of het gaat om een geheugenbouwsteen, een LC -display of een microcontroller. Alle busdeelnemers kunnen zend - en ontvangstfuncties uitvoeren, afhankelijk van hun toepassing. Zo zal een toetsenbordinterface doorgaans uitsluitend data verzenden en een LC -display uitsluitend data ontvangen.Een apparaat is *master* als het een datatransport initieert, met begrip van de transportrichting, en hiertoe de relevante klokpulsen op de SCL -leiding genereert. Een apparaat dat door de *master* wordt aangesproken (geadresseerd), en daardoor deel uitmaakt van het datatransport, wordt *slave* genoemd. De afzonderlijke apparaten zijn in principe altijd in staat om zowel master- als slave-functies uit te voeren. Het feit dat iedere busdeelnemer zowel master als slave kan zijn wil niet zeggen dat zoiets ook daadwerkelijk het geval is voor iedere bouwsteen. Aangezien een willekeurig aantal IC's op de bus een datatransport kan initiëren, is de I²C -bus een multimasterbus. Uiteraard kan op enig tijdstip slechts één master actief zijn en de bus gebruiken. Het I²C –busprotocol kent verscheidene maatregelen om conflicten tussen potentiële masters en de feitelijke master te voorkomen. De master test voordat datatransport plaatsvindt of de bus vrij is. Zelfs in de dat twee masters gelijktijdig beginnen met het iniëren van datatransport, wordt dit ondervangen door twee procedures met de namen *kloksynchronisatie* en *arbitrage*. Kloksynchronisatie zorgt voor synchronisatie van de kloktijden van beide masters. Arbitrage leidt ertoe dat de masterdie als eerste een “1” verzendt als de andere master tegelijkertijd een “0” verzendt, de bustoekenning verliest. Daarom wordt het logisch-nulniveau bij de I²C –bus ook wel het dominant niveau genoemd.



Algemene eigenschappen

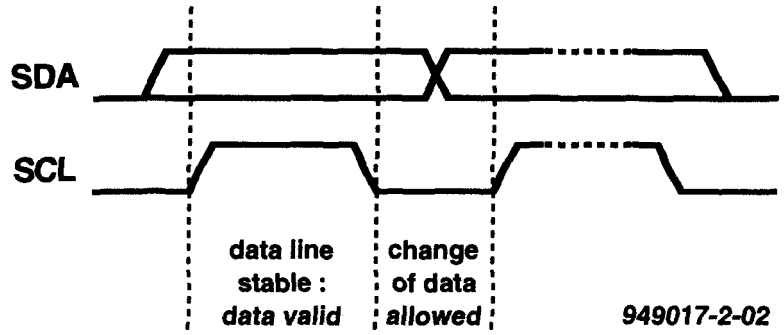
Zowel de leiding SCL als de leiding SDA werkt bidirectioneel, in beide richtingen. Beide leidingen zijn via een pullupweerstand met de positieve voedingsspanning verbonden. Bij een vrije bus zijn beide leidingen dus '1'. Chips die op de I²C -bus zijn aangesloten moeten open-collectoruitgangen (transistroren, stroomsturing) of open-drainuitgangen (fet’s,spanningsturing) hebben. Als er een 'O' op een leiding moet komen te staan, wordt zo'n uitgang “omlaag getrokken”. Het geheel van chips die op SDA en SCL zijn aangesloten vormt dus een wired-AND-schakeling. Zie onderstaande figuur. De maximale datatransportsnelheid

bedraagt doorgaans 100 kbits/s. In de zogenaamde *fast-modus,* een uitbreiding van het I²C -protocol, bedraagt die snelheid 400 kbits/s. Het maximale aantal busdeelnemers wordt bepaald door de maximale buscapaciteit van 400 pF.

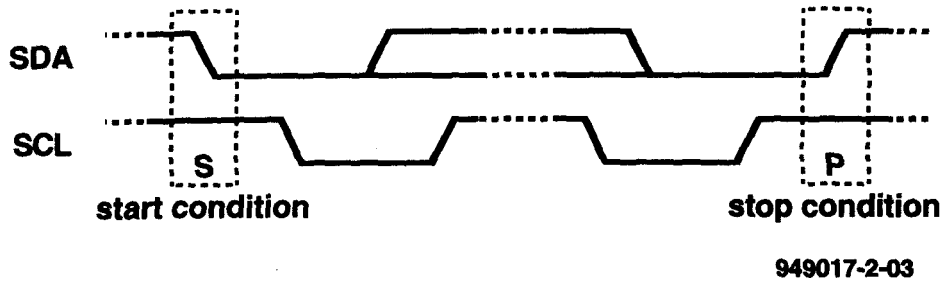
Dataoverdracht op bitniveau

Algemeen
De basis van de overdracht van woord-data is de overdracht in afzonderlijke bits. Dit is de taak van de bitlevel-laag van het I²C -busprotocol. In principe worden de afzonderlijke bits gecodeerd via een niveau op de kloklijn. SCL fungeert daarbij als seriële klok, net als bij een schuifregister. Speciale niveau/flankcombinaties van de beide leidingen coderen een zogenaamde startconditie respectievelijk een stopconditie. Daarmee kunnen het begin en het einde van het datatransport met behulp SDA en SCL leidingen worden gededecteerd. Voor beide lijnen geldt dat een niveau lager dan 1,5 V als laag ("0") wordt geïnterpreteerd en een niveau hoger dan 3 V als hoog ("1").

Geldigheid van de data
Data op de SDA -lijn moet stabiel zijn zo lang de SCL -lijn hoog is. Verandering zouden anders namelijk als een start- of stopconditie kunnen worden geïnterpreteerd. Dat betekent dat apparaten die de SDA -lijn moeten zetten terwijl de SCL -lijn laag is. Apparaten die data lezen doen dat dan uiteraard met SCL hoog.



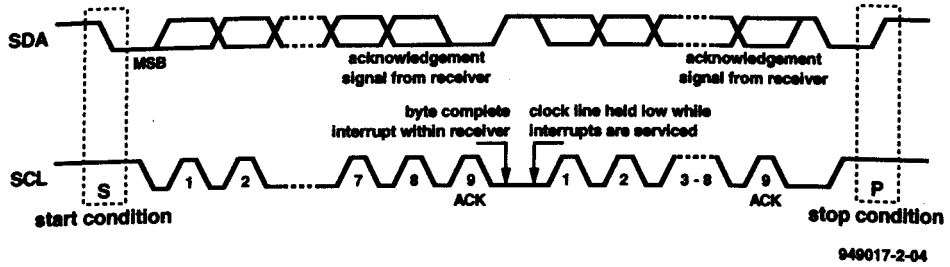
Star/stopconditie
De start- en stopconditie wordt door flanken van de SDA -lijn gecodeerd waarbij de SCL -lijn hoog is. Voor de startconditie is een dalende flank (van "1" naar "0") op SDA nodig, voor de stopconditie een stijgende flank (van '0' naar '1'). Volgens de specificatie mogen deze condities op ieder tijdstip tijdens een lopend datatransport plaatsvinden. Het is echter wel zaak om daarbij voorzichtig te werk te gaan.



Na een startconditie is de bus bezet *(busy)*. Alleen de master die de startconditie maakte mag de stopconditie creëren. Na deze conditie is de bus leeg *(idle)*. Een bijzonderheid is de zogenaamde *repeated start condition.* Hierbij beëindigt de master het datatransport niet door een stopconditie, maar realiseert simpelweg een nieuwe startconditie. Dat spaart wat tijd. Ook hier geldt: kijk uit bij zuivere software -implementaties van het I²C -busprotocol.

Dataoverdracht op byteniveau

De kleinste geheugeneenheid voor dataoverdracht tussen apparaten is het byte, acht bits. Het aantal over te dragen bytes is in principe onbegrensd. Veel bouwstenen kunnen echter afhankelijk van hun functie slechts een bepaald aantal bytes verzenden en ontvangen. De overdracht van een byte vergt 8 klokpulsen op de SCL -lijn. Die worden door de master gegenereerd. De geadresseerde slave mag echter de laagtoestand van de SCL -lijn naar eigen goeddunken verlengen. Deze procedure wordt *clock stretching* genoemd. Zo kunnen langzame slaven wachttijden inlassen. De master moet hiermee rekening houden. Bij het transport wordt het hoogstwaardige bit (MSB) van een byte als eerste verzonden, en het laagstwaardige bit (LSB) als laatste.



Na het achtste bit wordt aansluitend een zogenaamd *acknowledge-bit ver*zonden. Daartoe wordt een negende klokpuls gegenereerd. Met dit bit kan handshaking worden geïmplementeerd. Als de master data naar een slave zendt, moet die slave na ieder ontvangen byte een acknowledge verzenden, die door de master moet worden gelezen. Is dit bit '0', dan kan de overdracht verdergaan. Een acknowledge-bit '1' betekent dat de slave niet of niet meer deelneemt aan de overdracht. Als anderzijds de master de slave uitleest, moet de master na ieder ontvangen byte een acknowledge-bit verzenden. Men moet ervan doordrongen zijn dat het gebruik van het acknowledge-bit verplicht is. Geen zender mag doorgaan met zenden na de ontvangst van een bit '1'. (Als het acknowledge-bit '1' is wordt dit ook wel NACK genoemd; de N staat voor 'not'.) Evenzo moet een master het laatste in te lezen en ingelezen byte met een NACK, dus een '1', bevestigen, en niet het inlezen domweg beëindigen door het genereren van een stopconditie. Uiteraard mag het ontvangende IC ten behoeve van het verzenden van het acknowledge-bit de SDA-lijn slechts laag maken terwijl de SCL -lijn hoog is, net zoals dat bij databits geldt.

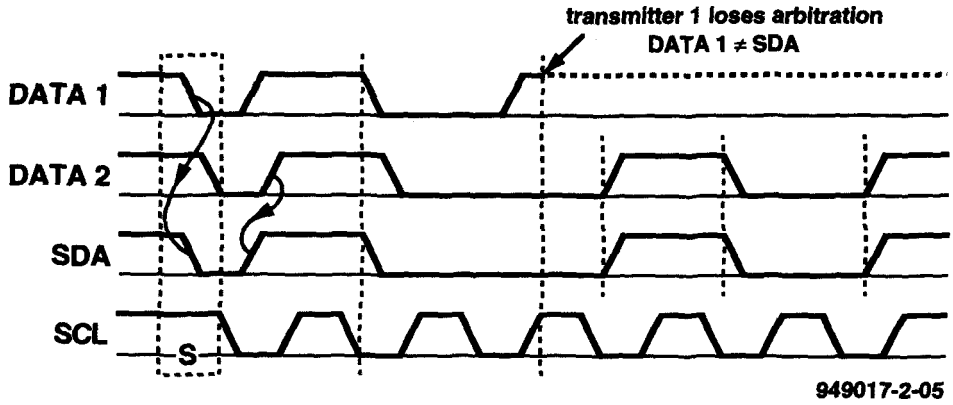
Arbitrage en kloksynchronisatie

Kloksynchronisatie

Alle masters genereren hun klokpulsen autonoom. Door de wired-AND-opzet van de buslijnen bestaat echter de mogelijkheid dat concurrerende masters die klokpulsen met elkaar kunnen synchroniseren. Een master die een '0' op de SCL-lijn zet, zal ooit daarvan weer een 'l' maken. Vanaf dat tijdstip gaat de master niet simpelweg door, maar leest hij herhaald de SCL-lijn totdat deze 'l' wordt. Dat betekent dat. zolang een ander apparaat de SCL-lijn nog laag houdt, de master zich hierop synchroniseert. Dat andere apparaat kan of een andere master zijn of een

slave die een wachttijd zou willen afdwingen. Al met al prevaleert dus de langste laagperiode, en dus het traagste apparaaat.

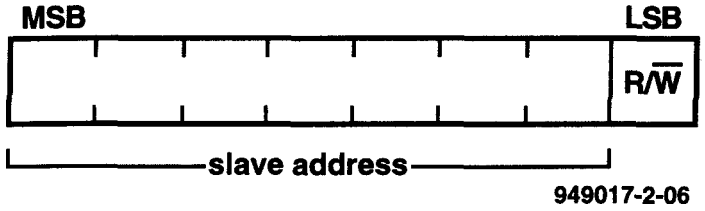
Arbitrage
Onder arbitrage wordt het proces verstaan dat de toekenning van de bus aan de diverse masters regelt. Op de I²C-bus is een speciale procedure met de naam *CSMA/CD (Carrier Sense, Multiple Acces with Collision Avoidance)* geimplementeerd. Carrier Sense betekent dat een master die iets wil lezen of schrijven, de lijnen scant om vast te stellen of de bus vrij is of niet De I²C-bus wordt als bezet geïnterpreteerd als één van de beide lijnen laag is. De tijd, gedurende welke beide lijnen hoog moeten zijn om een vrije bus vast te stellen, wordt in de parameterspecificaties van Philips $t_{HD;STA}$ genoemd. Bij het standaard protocol bedraagt die tijd minstens 4,7 s. Als de master een vrije bus heeft vastgesteld, genereert hij een startconditie en verzendt het eerste byte, dat het adres van de slave van bestemming bevat. Als een andere master dit gelijktijdig zou willen doen, dan worden eerst de pulsen op de SCL-lijn uitgelijnd, zie het hoofdstuk 'kloksynchronisatie'. Gedurende de gelezen klok-hoogfasen vergelijken de masters nu het indertijd verzonden SDA-bit met het door hen teruggelezen bit. Dat functioneert net zo als bij de conditie van de kloklijn. Een master die een 'l' verzendt en een'0' ontvangt (omdat de '0' vanwege de wired-AND dominant is), ziet dan dat een ander apparaat de bus bedient en breekt de verzending af. Deze procedure heet *collision avoidance,* omdat de 'zegenrijke' master van de botsing met de concurrerende master geen bitschade oploopt. Dit in tegenstelling tot een andere procedure met de naam *collision detection,* waarbij beide masters de bus moeten vrijgeven. Door collision avoidance wordt het transportvermogen van de bus verhoogd, omdat er altijd een master is die de bus krijgt toebedeeld.



De adressering van I²C-componenten

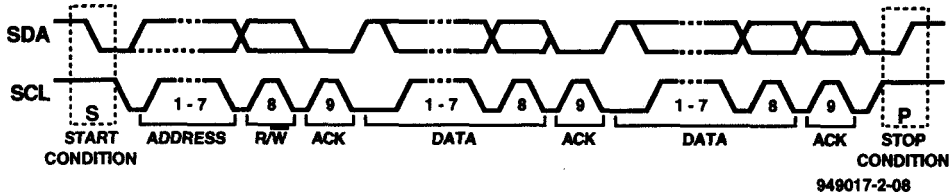
Algemeen
Iedere bouwsteen die vanuit de bus moet kunnen worden aangesproken (geadresseerd), heeft ter onderscheiding van andere bouwstenen een uniek adres: het slave-adres. Bouwstenen die uitsluitend als master optreden hebben geen adres nodig. Oorspronkelijk voorzag het I²C-concept in 7-bit adressen. Hiermee kunnen maximaal 128 bouwstenen worden geadresseerd. Er zijn echter bepaalde adressen gereserveerd, zodat er in feite minder zijn. Philips heeft een uitbreiding van de adresruimte gespecificeerd, waarbij de adreslengte 10 bits is (wordt in dit dossier niet besproken).
7-bit adressering

Als een master met een slave wil communiceren, moet hij eerst de bus claimen via een startcon-ditie (arbitrage). Dan zendt hij als eerste een byte waarvan de eerste 7 bits het adres van de gewenste slave vormen. Het laatste bit legt de transportrichting vast ('0' = schrijven, '1' = lezen).



Na een startconditie moeten alle aangesloten slave de bus bewaken om dit byte te lezen. Hierbij kunnen langzame slaves de master tot wachten dwingen. Nadat alle 8 bits zijn ontvangen vergelijkt iedere slave het ontvangen adres met zijn eigen adres. Dit adres ligt vast voor de bouwsteen of kan zijn ingesteld met dipswitches, jumpers of draadbruggen. Bouwstenen met een afwijkend adres nemen afscheid van de bus en wachten op een volgende startconditie. De bouwsteen met het juiste adres genereert een acknowledge-bit. Nu weet de master dat de geadresseerde slave tot datatransport in staat is. Komt er geen acknowledge-bit, dan is de bouwsteen beschadigd, of er is geen bouwsteen met dat adres.

Adresindeling
Zoals al gezegd vormen de eerste 7 bits van het eerste byte het adres van de te adresseren I2C-bouwsteen. Een bijzonder geval betreft het adres 0. Als het adres 0, gecombineerd met het richtingsbit '0' (schrijven) wordt uitgezonden, dan is er sprake van een *general call.* Het adres 0 heet daarom *general call address.* Via dit adres wordt een *broadcast* ingeleid, dus data die voor meer dan één slave bedoeld is. Dergelijke data kan uiteraard uitsluitend van de master naar de slaves worden geschreven. Het adres 0, gecombineerd met een richtingsbit 'l', stelt een zoge-naamd *start byte* voor. Dit byte mag voorafgaand aan het eigenlijke datatransport worden ver-zonden en dient ter ondersteuning van trage I2C-bouwstenen, die het busprotocol via software afwikkelen. De adrescombinaties 0000 001 en 0000 010 zijn gereserveerd voor de onder-steuning van busdeelnemers die ook andere dan I2C-protocollen kunnen afwerken.



Transfer bij 7-bit adressering
Het principiële verloop van een transfer is in bovenstaande figuur te zien. Na een startconditie volgen de 7 adresbits en het richtingsbit. De geadresseerde slave zendt een acknowledge-bit. Daarna kan een willekeurig aantal bytetransfers plaatsvinden. Als laatste genereert de master een stopconditie en geeft de bus vrij. Er is echter nog een andere mogelijkheid om een transfer te beëindigen. De master kan na de behandeling van een compleet byte een nieuwe startconditie genereren, gevolgd door een adresbyte. Dit is de *repeated start.* Het nieuwe adres kan een

andere slave betreffen of dezelfde slave met gewijzigd richtingsbit. Daarmee zijn er drie nader toe te lichten transfertpen.

1. **De master zendt data naar de slave**
De richting van de transfer blijft gehandhaafd. De transfer eindigt met een stopconditie (of een repeated start met nieuw slave-adres).

Van master naar slave

S	SLAVE ADDRESS	R/W	A	DATA	A	DATA	A	P

2. De master leest data van de slave
Nadat de slave het adresbyte heeft ontvangen, schakelt deze naar de zendmodus en de master naar de ontvangmodus. Dat betekent dat de slave alleen nog de acknowledge voor het adresbyte genereert; vervolgens worden alle acknowledges door de master gegenereerd.

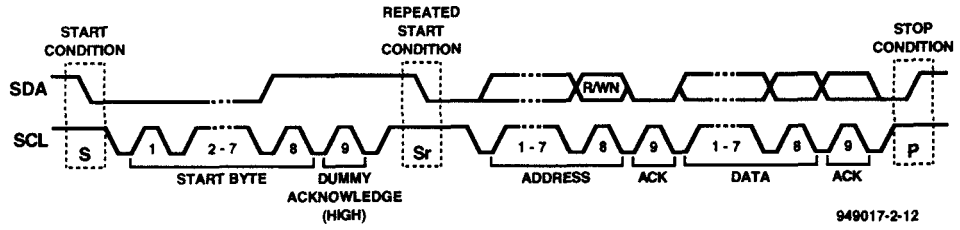
Van slave naar master

S	SLAVE ADDRESS	R/W	A	DATA	A	DATA	A	P

3. **Gecombineerd formaat**
De master begint lezend/schrijvend zoals gebruikelijk. Via een repeated start condition spreekt hij de slave opnieuw aan, maar met tegengestelde richting. Richtingswisselingen kunnen willekeurig vaak optreden.

S	SLAVE ADDRESS	R/W	DATA	A/A	S r	SLAVE ADRESS	R/W	A	DATA	A/A	P

Startbyte-procedure
In principe zijn er twee verschillende soorten apparaten die je op de I2C-bus kunt aansluiten. Apparaten met een hardware-interface zijn voldoende snel om de bus correct te behandelen. Is daarentegen de slavemodus via software op. Het idee achter de startbyte-procedure is om een startconditie te hebben die eenvoudiger kan worden herkend en die minder vaak hoeft te worden gescand. De procedure ziet er in detail als volgt uit.



Na een startconditie wordt een startbyte (0000 0001) verzonden. Dat uit zich tevens door het feit dat de SDA-lijn 7 klokpulsen lang constant laag blijft. De master verwacht geen acknowledge en hij genereert een negende klokpuls met aansluitend weer een repeated start condition. De slave heeft daarvoor een tot éénzevende deel gereduceerde klokfrequentie nodig. Hij moet

echter na het herkennen van het startbyte in staat zijn om alsnog met een hogere klokfrequentie te werken.

Let op: de startbyteprocedure wordt niet toegepast bij een repeated start. Een transfer moet voor een startbyte met een stopconditie worden beëindigd (uitzondering:richtingswissel bij een bouwsteen).