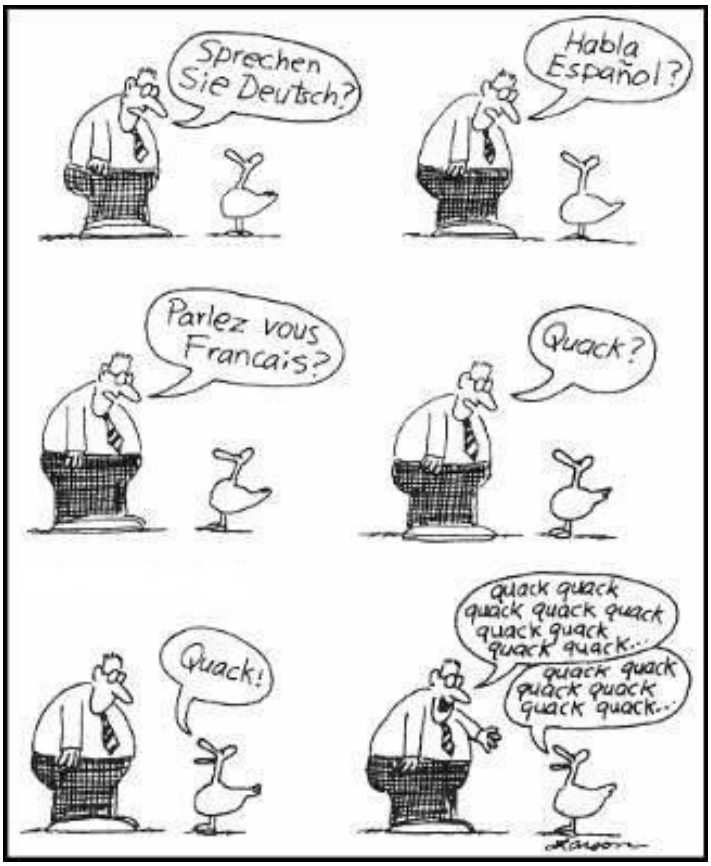
Datakommunikation

Hvis to personer skal kommunikere med hinanden, bliver det kun vellykket, hvis de **både** taler samme sprog, **og** er enige om ikke at tale i munden på hinanden. På samme måde er det indenfor datakommunikation i elektronikkens verden. Når to eller flere "enheder" skal kommunikere sammen, skal de både "tale" et fælles sprog, og der skal være nogle regler om hvem der må tale og hvornår, så ingen taler i munden på hinanden.



Figur 1 Man skal som minimum tale samme sprog for at føre en samtale. © Gary Larson, 1984.

Indhold

Datakommunikation

Helt overordnet er datakommunikation egentlig bare en beskrivelse for, at man foretager en kommunikation med noget data. Data kan så være alt fra at sige ”tænd” eller ”sluk”, til at kunne sende kommandoer, tekst, lyd og/eller billede.

Datakommunikation – helt simpelt

Den mest simple form for datakommunikation er, at man har en LED som man enten tænder eller slukker. Modtageren af kommunikationen, kan være en person der ser lysdioden og ved hvad ”LED tændt” og ”LED slukket” betyder.

Man kan vælge at udvide fra én LED til flere LED (evt. bruge forskellige farver) og på den måde signalere et budskab (tænk på lyssignaler i lyskryds).

Datakommunikation – send et budskab til alle der lytter

I afsnittet ovenfor blev et simpelt LED kommunikationssystem beskrevet. Med det vil alle der kender til de forskellige LEDs betydninger kunne modtage og forstå budskabet. Folk der ikke kender farvernes betydning, vil bare kunne trække på skulderen, og gå videre.

Der findes flere forskellige datakommunikationssystemer der er baseret på denne måde at kommunikere på. F.eks. er fjernbetjeninger til TV og musikanlæg ofte lavet som én-vejs LED kommunikation; i stedet for at have en lysdiode der udsender synlig lys, anvendes en infrarød lysdiode (IR LED), og i stedet for at have lige så mange lysdioder som der er funktioner på TV’et, så ”blinker” lysdioden på fjernbetjeningen sin besked til TV’et. Blinkene laves i et bestemt mønster (næsten ligesom morsekode), hvor producenten af TV’et har fastlagt hvordan der skal blinkes for at signalere de enkelte funktioner (tænd/sluk/skru op/ned…). Man kalder sådan et kommunikationssprog for *en protokol*.

Figur 2 Fjernbetjening til TV, som kommunikerer via IR.

Da blinkene jo er digitale (enten er IR lysdioden tændt eller også er den slukket [1 eller 0]), foregår kommunikationen digitalt. Hver producent af TV og audioudstyr har sin egen måde at kommunikere på (kommunikationsprotokol), så derfor vil et Philips TV ikke fungere med en fjernbetjening fra Samsung.

Datakommunikation – Styr på hvem der taler sammen

Der findes typer af kommunikation, hvor man har brug for at kunne tale begge veje (to-vejs kommunikation), og dermed have brug for at vide hvem man præcist vil tale til. Sådan er det f.eks. med IP kommunikation (netværk) eller Bluetooth kommunikation. Her sender afsenderen besked om hvem modtageren er, og også hvem afsenderen er. På den måde kan modtageren svare tilbage til afsenderen.

Data sendes i "pakker", hvor hver datapakke indeholder information om hvem den sendes fra og hvem der er modtager, samt hvilket pakkenummer der er tale om (og meget mere). Derudover er der også selve indholdet i pakken – det data man vil sende (del af lyd, billede, tekst, eller lignende).

Protokoller

Når to enheder skal kommunikere sammen, skal de som tidligere beskrevet "tale" samme sprog. Sproget der tales, kaldes for en *protokol*. Både afsender og modtager kender protokollen, og dermed vil enhederne kunne forstå hinanden. Det svarer til, at hvis en dansker vil sige "Hej" til en engelsktalende person, så siger han "Hello"; han bruger protokollen "*Den engelske ordbog*" til at vælge ord fra, som han ved vil kunne forstås af modtageren.

Indenfor digital elektronik (altså der hvor vi arbejder med 0'er og 1'er (eller "High" og "Low")) ved vi, at vi skal have et antal bits til rådighed, for så kan vi repræsentere en række tal. Hvis vi for eksempel har 4 bit til rådighed (Hvert bit kan enten være 0 eller 1), så kan vi opskrive $2^4 = 16$ forskellige kombinationer af 0 og 1. Det giver f.eks. tallene fra 0 til 15:

Binært	Decimalt
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Vi har nu 16 forskellige værdier eller kommandoer til vores protokol, fordi vi jo kan beslutte, at de 16 kombinationer skal betyde noget i protokollen. Det kunne f.eks. være som følger til brug i en fjernbetjening:

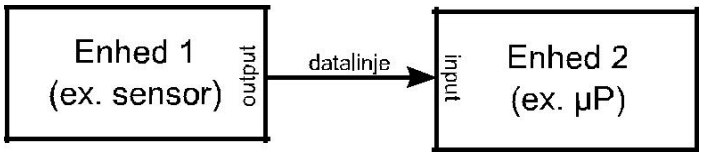
Binært	Betydning
0000	
0001	
0010	Tænd for TV
0011	
0100	Vælg TV
0101	Vælg USB
0110	Vælg DVD
0111	Sluk for TV
1000	
1001	Skru op for volumen
1010	
1011	Skru ned for volumen
1100	Skift kanal, op
1101	
1110	Skift kanal, ned
1111	

Kommandoerne, det vil sige de 4 digitale bits, kan så sendes enten som trådet (dvs. med ledning mellem sender og modtager) eller trådløs (dvs. uden ledning mellem sender og modtager). I de kommende afsnit beskrives både trådet og trådløs kommunikation.

Trådført datakommunikation

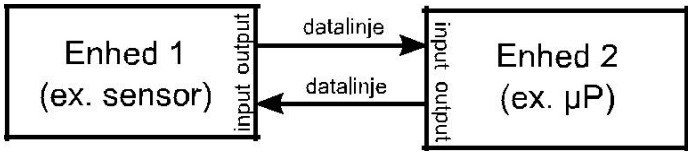
Trådført datakommunikation dækker over kommunikation mellem enheder der er forbundet via en fast forbindelse (ledning, optisk fiber el. lign.). En stor fordel ved en trådført kommunikationsforbindelse er, at det signal der afsendes, også kommer frem til modtageren (med mindre der er én der har afbrudt forbindelsen).

Kommunikationen kan enten være én-vejs eller to-vejs. Er den én-vejs sendes der kun data den ene vej fra én enhed til en anden (se Figur 3). Dette kunne for eksempel være en temperatursensor der sender data til en mikroprocessor, uden at mikroprocessoren har mulighed for at "tale tilbage" til temperatursensoren.



Figur 3 Én-vejs kommunikation: Data sendes fra enhed 1 til enhed 2

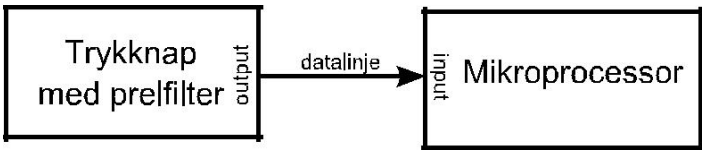
Kommunikationen kan også foregå to-vejs hvor begge enheder kan sende data til hinanden (se Figur 4). Det kunne for eksempel være en batterisensor, hvor mikroprocessoren kan spørge sensoren hvor meget energi der er tilbage i batteriet, eller den aktuelle spænding og strømtræk; når mikroprocessoren spørger, svarer sensoren tilbage.



Figur 4 To-vejskommunikation: Data sendes både fra enhed 1 til enhed 2, men også fra enhed 2 til enhed 1

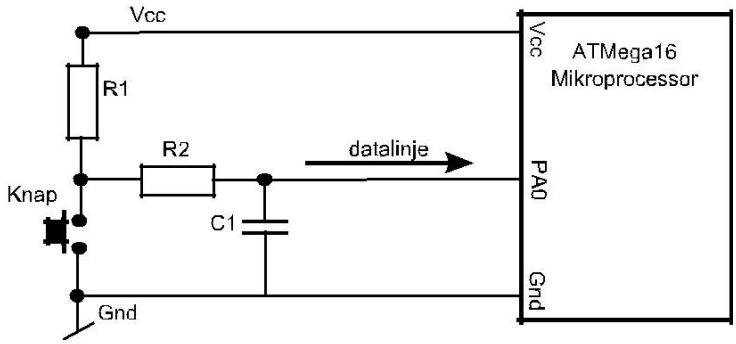
Simpel en-vejskommunikation: Et trykknapinput

Et eksempel på en simpel én-vejskommunikation er et trykknapsinput til f.eks. en mikroprocessor. Figur 5 viser et blokdiagram med trykknap og mikroprocessor hvor det ses, at data går fra "trykknap med prelfilter"-blokken til "mikroprocessor"-blokken.



Figur 5 En trykknap med prelfilter er forbundet til en mikroprocessor. Det er outputet fra prelfilteret der sender input til mikroprocessoren

Figur 6 viser el-diagrammet over konstruktionen. Her ses det, at datalinjen "*bare*" er spændingsoutputtet fra prel filteret, der er forbundet til en port på mikroprocessoren. Porten er i mikroprocessoren sat op som et input, og når knappen trykkes ned, vil datalinjen (inputtet til mikroprocessoren) gå fra spændingen *Vcc* [V] til 0 [V], og dette kan mikroprocessoren registrere som skiftet fra "digital høj" til "digital lav".



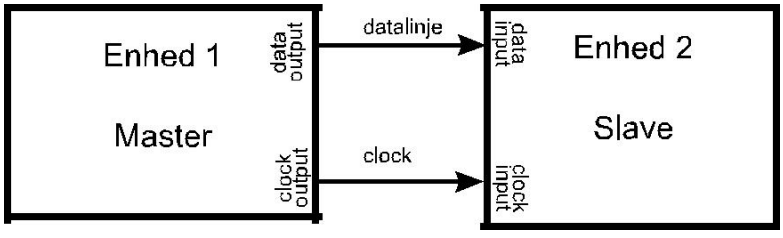
Figur 6 Diagram over trykknap med prelfilter, der er forbundet til mikroprocessoren

Kommunikationsbus

Hvis man har behov for at kunne sende forskellige kommandoer og data over én enkelt datalinje, så er det ikke nok bare at kunne signalere at datalinjen er "høj" eller "lav" – dette giver jo kun 2 tilstande. I stedet for anvender man en kommunikationsbus. Der findes flere forskellige typer kommunikationsbusser, og deles op i enten *serielle busser* eller *parallelle busser*.

Seriel databus

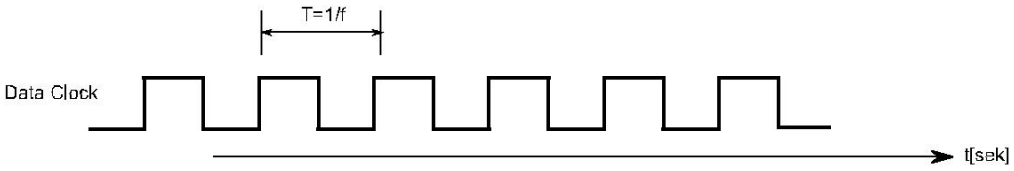
Hvis man har en seriel databus vil det sige, at data der udveksles på bussen *kommer i serie* altså "efter hinanden". Da datalinjen enten kan være "digital høj" eller "digital lav", skal data sendes som binære værdier. Ønsker man at sende tallet 8d (dvs. decimaltallet 8), skal der sendes 4 bit, da den binære værdi for 8d er 0b1000. Man aftaler på forhånd om man starter fra MSB (Most Significant Bit) eller fra LSB (Least Significant Bit), og så "clockes" data ud, hver gang *clock'en* pulser (se afsnittet "Data clock" side 8). Et eksempel på en seriel databus ses på blokdiagrammet Figur 7.



Figur 7 Seriel databus, hvor data sendes serielt på en datalinje

Data clock

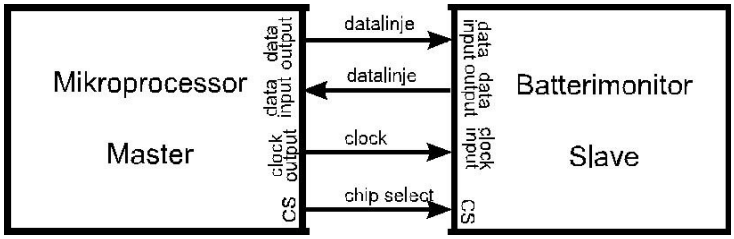
Der skal være "noget" der driver data ud på datalinjen. Dette er data clock'ens arbejde. Data clock'en er et digital output der skifter mellem "digital lav" og "digital høj". Dette ses på Figur 8. Tiden, *T*, er den tid det tager for en clock periode, og er givet som: 1/f, hvor tiden *T* måles i [sek] og frekvensen *f* måles i [Hz].



Figur 8 Data clock'en skifter mellem "digital høj" og "digital lav" med en given frekvens, *f*.

SPI Bus – Master og slave enheder

En *SPI bus* deler enhederne op så der er en *master* der bestemmer, og et antal *slaver* der gør hvad de bliver beordret til. SPI står for "*Serial Peripheral Interface Bus*", og er en seriel bus. Det er master-enheden der bestemmer, når der skal sendes og modtages data. Figur 9 viser en master-enhed koblet sammen med en slave-enhed via en SPI bus. Det ses, at der er 4 forbindelser imellem enhederne: 2 datalinjer, 1 clock og 1 chip select.



Figur 9 En master-enhed og en slave-enhed sammenkoblet via en SPI Bus. Pilene angiver retningen af dataflow

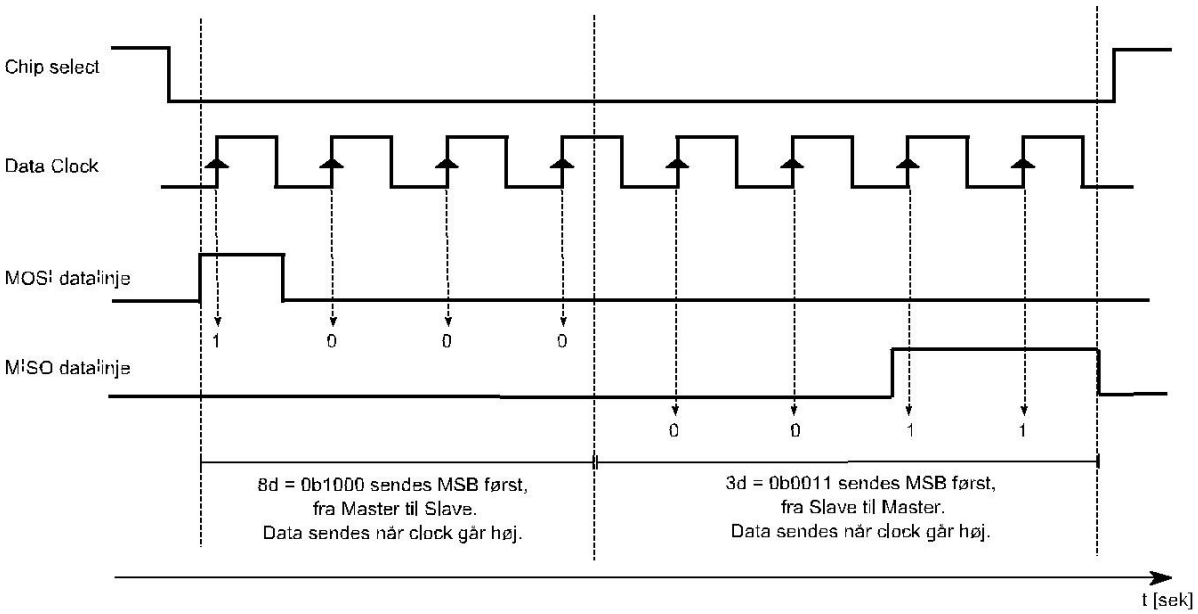
Når master-enheden ønsker at sende data til slave-enheden, sættes chip select (CS) til lav; så ved slave-enheden, at den skal lytte. Herefter pulser clock'en, mens de data der skal sendes fra master-enheden "clock'es" ud på datalinjen fra "data output" på masteren. Hvis slave-enheden skal svare tilbage, gøres dette efterfølgende på den datalinje som er data output fra slaven.

De to datalinjer kaldes henholdsvis for MOSI (Master Output Slave Input) og MISO (Master Input Slave Output).

Et eksempel:

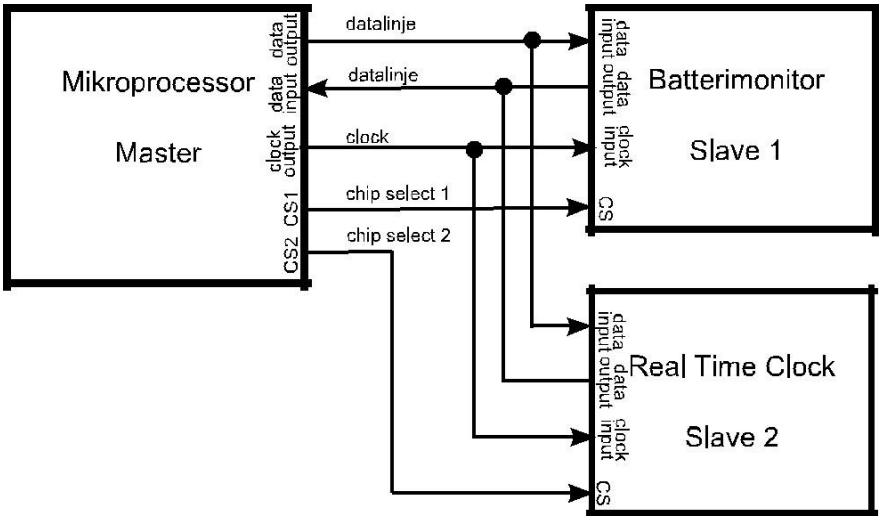
Hvis Masteren ønsker at sende tallet 8d til slaven, og slaven svarer 3d tilbage, så ser det ud som følger. Opsætningen er, at data sendes med MSB først, og datalængden er 4 bit. Data sendes når clock'en går fra lav til høj. Figur 10 viser timingen, hvor man kan se, hvad der sker på de to datalinjer i forhold til chip select og clock'en.

1. Chip select går lav, hvorved slave-enheden gør sig klar til at modtage data på MOSI porten.
2. Det første databit ud af de 4 bit som masteren ønsker at sende, gøres klar. Dvs. da bit'et er "digital høj" (det var jo den binære værdi "1" der skal sendes), sættes MOSI datalinjen "digital høj".
3. Data clock'en starter, og slave-enheden ser at MOSI datalinjen er "digital høj" idet clock'en går fra lav til høj.
4. Det andet databit der skal sendes fra masteren til slaven er den binære værdi "0". Derfor sættes MOSI datalinjen "digital lav".
5. Idet Data clock'en igen går fra lav til høj, registrerer slave-enheden at MOSI datalinjen er lav, og har dermed modtaget et "0".
6. Dette fortsætter indtil alle fire bit er afsendt.
7. Herefter sætter slave-enheden MISO datalinjen "digital lav", da den skal sende den binære værdi 0b0011, og det første bit (MSB) er et "0".
8. Når clock'en går fra lav til høj, registrerer master-enheden at MISO datalinjen er "digital lav", og har dermed modtaget et "0".
9. Efter slave-enheden har afsendt alle fire bit, trækker master-enheden chip select høj igen, og samtalen imellem de to enheder er overstået.



Figur 10 Et eksempel hvor en master-enhed sender 4 bit data (8d) til en slave-enhed, og efterfølgende svarer slave-enheden tilbage med tallet 3d. Transmissionen foregår via SPI bussen.

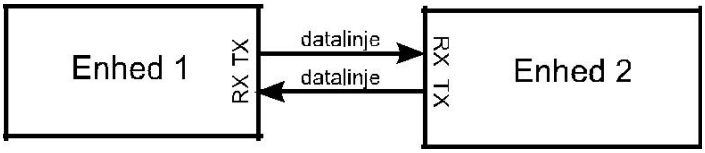
Når man arbejder med SPI bussen, kan man sagtens have flere slaver på den samme bus. For hver slave, skal der være en separat chip select forbindelse fra master-enheden, og ud til den enkelte slave-enhed. Dette ses på Figur 11 hvor to slave-enheder er koblet til den samme master-enhed, via SPI bussen.



Figur 11 En master-enhed og to slave-enheder forbundet via SPI Bus

UART – Universal Asynchronous Receiver/Transmitter

En simpel trådført seriel kommunikation er ved at benytte en UART. A'et i U**A**RT står for *Asynchronous* (asynkron), og betyder, at der *ikke* sendes et clock signal mellem afsender og modtager, men at der forud er aftalt hvilken hastighed der sendes med.



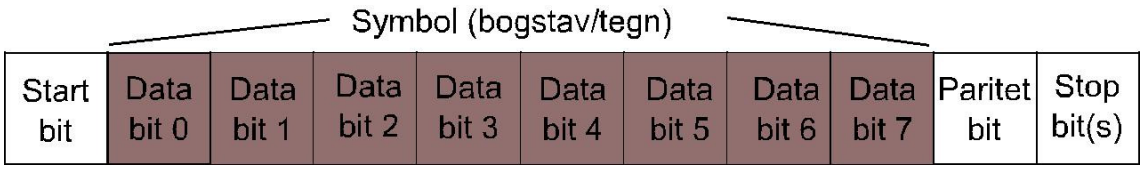
Figur 12 To enheder kommunikerer via UART

Figur 12 viser to enheder der kommunikerer via UART, hvor TX betyder "Transmit" og RX betyder "Receive".

UART protokol

Når man sender data via en UART, benytter man en særlig protokol. En protokol er en måde hvorpå man "pakker data ind" når det skal sendes over datalinjen. Både afsender og modtager skal være klar over hvilken protokol der anvendes.

Protokollen ser ud som vist på Figur 13, hvor der først sendes et start bit. Herefter sendes databit'ene med det mindst betydende bit først (LSB). Til sidst kommer et paritets bit og stop bit'et (eller bit'ene).



Figur 13 UART protokol

Start bit er **lav** (digital 0).

Data er oftest 8 bit langt, men kan også være 5, 6, 7 eller 9 bit lang.

Paritet bit bruges som et check bit, der er genereret ud fra alle data bit er blevet XOR'et sammen. Dette kan vælges som enten ingen/even/odd.

Stop bit er enten en eller to **høje** bit (digital 1'ere).

Protokollen sender et symbol (bogstav/tegn) ad gangen.

Eksempel:

Hvis datakommunikationen er sat op til at sende 8 data bit, ingen paritet bit og 2 stop bit, så skal der sendes i alt 11 bit pr symbol. Hvis man ønsker at sende bogstavet "H", slår man først op i en ASCII tabel, hvor man kan se hvilken tal-værdi bogstavet "H" svarer til. Her ses, at det er tallet 72d (= 54h) der betyder "H". Det 8-bit binære tal for 72d er: 0100 1000b (MSB…LSB).

Der skal dermed sendes følgende bit:

Bit nr	Bit	Binær værdi
1	Start bit	0
2 - 9	Data bit	00010010 (LSB…MSB)
	Paritets bit	-
10 - 11	Stop bit	11
	Alle bit i rækkefølge:	00001001011

Transmissions hastighed

Ud over antal data bit, paritets bit og stop bit, skal transmissionshastigheden også vælges. Transmissionshastigheden angives som antal symboler pr. sekund, og angives i enheden *baud*. Typiske baud rater er: 9600, 19200, 38400, 57600 og 115200 baud.

Fra eksemplet i det foregående afsnit ses, at der hvor hvert symbol sendes 11 bit. En baud rate på 9600 vil dermed sende 9600 · 11 bit = 105.600 bit/sek = 105,6 kbps.

RS-232

PC'ere har (især for nogle år tilbage i tiden) været udstyret med UART kommunikationsporte. Da PC'ere benytter sig af signalleringsspændinger på +/- 12V, har man lavet en standard til UART kommunikation i forhold til PC'ere kaldet RS-232. RS-232 er lidt forsimplet UART hvor der anvendes +12V som digital 1, og -12V som digital 0.

Der findes IC'ere der kan håndtere konverteringen mellem UART og RS-232. Det er f.eks. kredsen MAX232 fra producenten Maxim, der på Figur 14 ses på et printboard.

Figur 14 UART <-> RS-232 konverter kreds fra Maxim, monteret på printboard.

Trådløs kommunikation

For rigtig mange mennesker, er en trådløs kommunikationsforbindelse at foretrække. Man slipper for at have en ledning forbundet mellem enhederne, der skal snakke sammen, hvorved ens frihedsgrader forøges. Men som elektronikkonstruktør gør det ikke jobbet lettere, hvis kommunikationen skal foregå trådløs; man kan nu ikke længere regne med, at det afsendte data bliver modtaget.

Datatransmission uden et clock-signal

Da vi tidligere så på trådførte serielle kommunikation (se afsnittet "Seriel databus", side 7) var der en data clock til stede. Data clock'en var det der "drev" data fra afsenderen til modtageren, og det som modtageren brugte til at vide hvornår data var klar til at blive læst.

Når man sender data trådløst, er det nødvendigt for både afsender-enheden og modtager-enheden at have den samme timing; dvs. modtager-enheden skal vide hvornår afsendt data er klar til at blive læst – en opgave data clock'en i den trådførte serielle kommunikation varetog. Man kunne forestille sig, at både afsender og modtager kørte efter hver sit "lokale" clock-signal der selvfølgelig skulle være helt ens. For at det ville virke, kræver det, at clock'ene i både afsender-enhed og modtager-enhed startes på nøjagtigt samme tidspunkt… Det er jo lidt svært, når de ikke fysisk er forbundet til hinanden. I praksis er det umuligt at få til at virke. Derfor kan man "indbygge" clock'en i selve datasignalet. Det gøres f.eks. ved brug af Manchester codning.

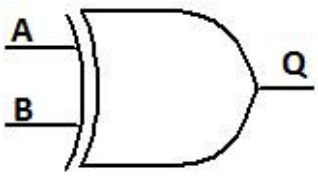
Manchester codning

Når man ikke har mulighed for at synkronisere afsender og modtager 100 % i forhold til at vide hvornår data er afsendt og skal læses, kan man benytte Manchester coding. Kort fortalt går det ud på, at man encoder de data man ønsker at sende, med data clock'en. Modtageren kan så – helt uafhængig af den data clock der kører på afsendersiden – udtrække de sendte data alene ved at se på hvornår de encodede data skifter fra digital lav til høj, eller fra høj til lav.

Et eksempel på dette gives her. Vi ønsker at sende det 8-bit lange tal 0b10011010 (det svarer til decimaltallet 154d). Disse 8 bit er vores data der skal sendes, og det er disse data der skal Manchester encodes.

Til Manchester encodningen anvendes den logiske XOR operation. XOR står for eXclusive OR, og Figur 15 viser symbolet for en XOR gate, mens Figur 16 viser sandhedstabellen for XOR gaten.

Det Manchester encodede data er resultatet af **data clock** XOR'et med **data**.

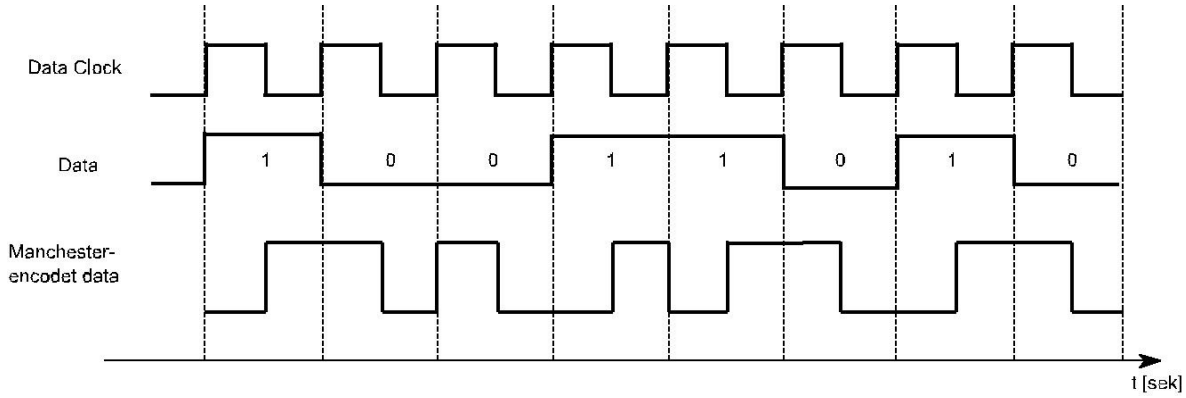


Figur 15 XOR gaten, også kendt som eXclusive OR.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

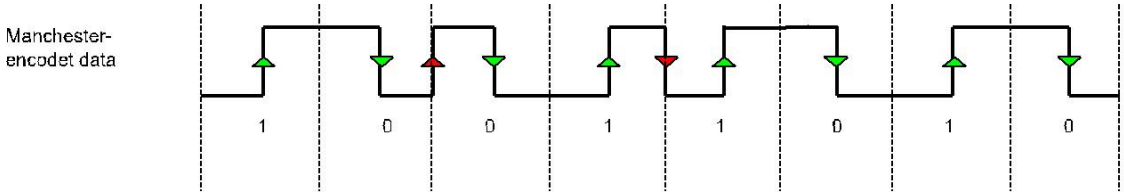
Figur 16 Sandhedstabel for XOR gaten.

Figur 17 viser timingen, og hvordan data clock og data encodes, som resultatet af data clock XOR'et med data. Det er de Manchester encodede data der sendes.



Figur 17 Manchester encoding af data. Her encodes 8 bit: 0x10011010.

Men hvordan får vi data ud igen? På modtagersiden kender man frekvensen af data clock'en på afsender-enheden, og har dermed en data clock kørende. Figur 18 viser data der modtages.



Figur 18 Det modtagede Manchester encodede data skal decodes; Et "lav til høj" skift midt i en periode afkodes til et "1", mens et "høj til lav" skift midt i en periode, afkodes til et "0". Skift i starten eller slutningen af en periode må ikke registreres.

Når data skal dekodes på modtagersiden, ses der på niveauskift midt i en clock periode; går niveauet fra "lav til høj", er data et "1" og går niveauet fra "høj til lav", er data et "0". Niveauskift i starten eller slutningen af en clock periode må ikke registreres (markeret med en rød pil på Figur 18).

Fejl detektering

Når man sender data trådløs, kan man ikke være sikker på, at det der sendes, også er det som modtageren modtager. Det skyldes forstyrrelser (f.eks. påvirker en tændt microbølgeovn en WiFi kommunikation, da begge opererer i 2,4 GHz frekvensområdet).

Når data bliver "ødelagt" ved en forstyrrelse, så betyder det, at 0'er kan blive til 1'ere og 1'ere kan blive til 0'er. Dermed kan det afsendte opfattes helt forkert. Hvis der for eksempel sendes kommandoen: 0110 (som i protokollen måske betyder "Skru op for lyden"), og data bliver "ødelagt" så modtageren modtager 0010 i stedet. Hvis 0010 betyder "Sluk for TV", så får vi et problem! Man bør derfor altid have indbygget en måde hvorpå man kan opdage, hvis de afsendte data er blevet "ødelagt" undervejs.

En måde at gøre det på, er ved at benytte såkaldte *checksums*. En checksum er et tal, der udregnes ud fra de data der skal sendes. Tallet kan f.eks. være 8 bit langt, og udregnes ligesom en funktion man kender fra matematikken. Inputtet til funktionen er det data man vil sende. Checksums resultatet sendes sammen med data til modtageren, og modtageren udregner også en checksum

for de modtagne data. Herefter sammenlignes med den modtagede checksum. Hvis begge checksums er ens, så er data valid (dvs. ikke blevet ødelagt under transport). Er de to checksums derimod *ikke* ens, så må det modtagne data ikke anvendes, da det med stor sandsynlighed er ødelagt.

Der findes flere forskellige checksums metoder. Det kan være CRC-8, CRC-16 og MD5.

Årsager til fejl-data ved trådløs kommunikation

Der er flere årsager til at det afsendte data ikke når (korrekt) frem til modtageren:

1. Hvis modtageren er udenfor rækkevidde, således data aldrig når frem.
2. Hvis en anden sender data samtidig, således de data der modtages er en blanding af data fra flere sendere.
3. Hvis data undervejs "ødelægges" som følge af pludselige forhindringer på transmissionsvejen (ex. en person der går mellem senderen og modtageren og "skygger" for signalet).

Når man skal designe et trådløst system, skal man indtænke hvorledes disse fejl minimeres, opdages og håndteres. Dette er et teoritungt område, og er udenfor dette dokuments område.

Transmissionstyper

Data kan sendes trådløs på flere forskellige måder. Det kan gøres som:

1. Infrarød optisk kommunikation (IR) som er den mest udbredte metode i TV fjernbetjeninger.
2. Ultralyds kommunikation, der tidligere har været anvendt til fjernbetjeninger.
3. Radiobølger (RF) hvor der kommunikeres på en eller flere frekvenser.