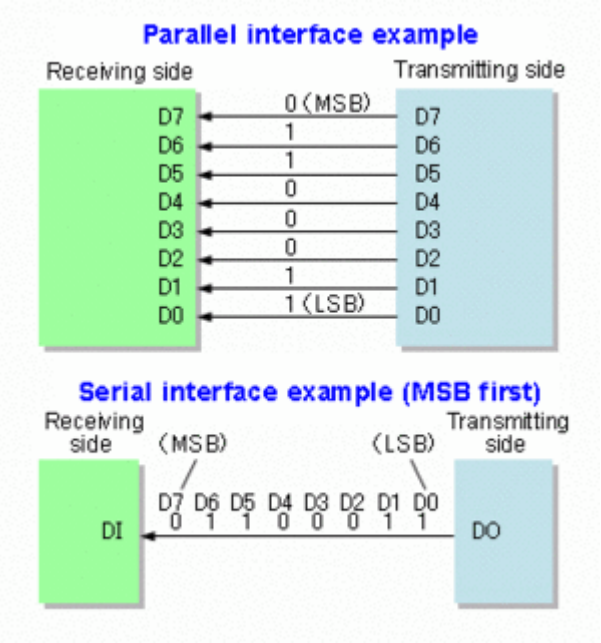
Synkronisering i datakommunikasjon

Når datamaskiner sender data til hverandre, må hver databit komme riktig fram til mottager. Synkronisering vil si at databit nr.0 fra sender blir registret som databit nr.0 hos mottager. Tilsvarende for databit nr.1, 2, 3, 4, 5, 6 og 7. (D0, D1, D2, D3….D7)

Parallell kommunikasjon

Ved parallell kommunikasjon gjøres denne synkroniseringen ved at hver databit i en byte, D0, D1, D2, … D7, får hver sin ledning i en kabel, pluss fellesledning (jord). Dette er kun mulig å bruke hvis det er korte forbindelser, kanskje et par meter. For mange år siden (1980-tallet) ble parallelle grensesnitt brukt til skrivere og instrumenter, som var direkte koblet til en datamaskin, over meget kort avstand. En slik kabel hadde mange ledninger i seg, og det var meget upraktisk og dyr. Man gikk derfor over til seriell kommunikasjon, hvor hver databit brukte samme ledningspar. Det er (kun) det som brukes til datakommunikasjon i dag.



Seriell kommunikasjon

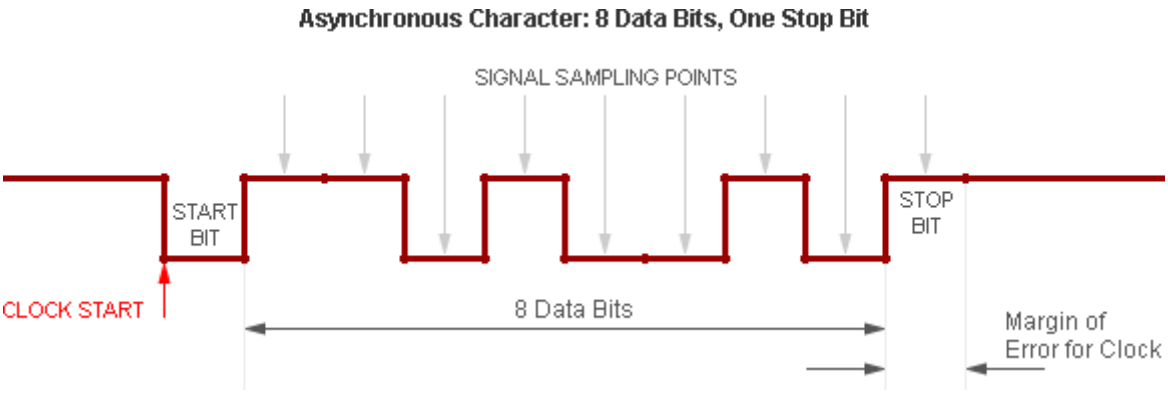
Ved seriell kommunikasjon bruker alle databitene samme linje (ledningspar, forbindelse).. Bitene sendes etter hverandre i tid. Dvs at hver databit har linjen en liten tid. Datahastigheten sier noe om hvor lang tid hver bit har linjen. F.eks 1 Mbit/s sier at i løpet av et sekund har 1 million bit (10^6) passert. Hver bit har da linjen 1 µs (10^{-6} sekund).

Da disse bitene kommer etter hverandre i tid, må mottager vite hvilken av de bitene som er henholdsvis D0, D1, D2 osv. Da må sender og mottager synkroniseres. Det finnes to hovedmåter det kan gjøres på. Det er asynkron og synkron dataoverføring.

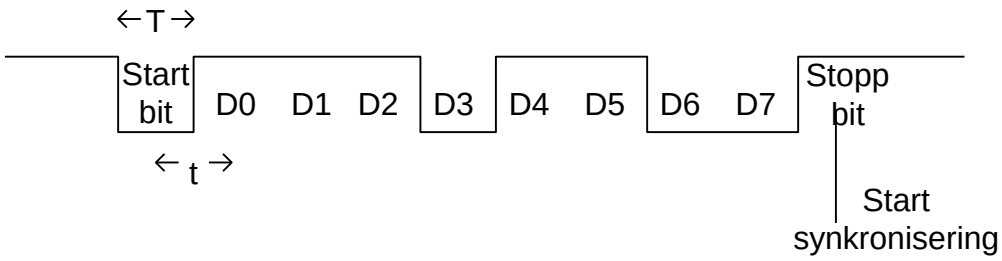
Asynkron dataoverføring

Her foregår synkroniseringen for hver byte. Det sendes en startbit før databitene sendes. En forbindelse, hvor det ikke går data, har tilstanden logisk 1. Mottageren vil hele tiden lete etter startbiten, som er logisk 0. Det gjør den ved å punktprøve inngangsignalet veldig mye raskere (16x eller 64x) enn datahastigheten. Med en gang mottageren oppdager at det er logisk 0, antar den at det er begynnelsen på startbiten. Mottageren (vet = er innstillt på) datahastigheten, og vil da punktprøve startbiten en gang til, denne gang mitt i startbiten. Da denne verdi også er logisk 0, er sender og

mottager synkronisert. Mottager punktprøver deretter mitt i hver bit, som kommer etter hverandre, og finner da ut om det er en 0 eller 1 i de forskjellige databitene. Mottager vet også hvilken databit den punktprøver, fordi første bit etter startbiten er D0. Neste er D1 osv fram til D7. Stop-bit er logisk 1, og kommer rett etter D7. Da stop-bit er detektert, vil mottager begynne med denne oversamplingen (16x eller 64x) igjen, for å lete etter startbit på neste byte.



Tiden for hver bit bestemmes av senderens klokke. Det er store **T** i figuren under. Mottagerens klokke bestemmer når bitene skal punktprøves. Det er lille **t** i figuren under. Senderklokka og mottagerklokka skal være lik hverandre, men de er ikke eksakt like. En liten forskjell i senderklokka og mottagerklokka vil medføre at bitene ikke blir punktprøvet eksakt i mitten. Avviket blir større for hver bit. Den første biten D0, vil bli punktprøvet så å si i mitten, mens for D7 er avviket størst. Men så lenge avviket ikke er større enn at biten blir punktprøvet innefor tiden for den bit'en, går det greit. I asynkron dataoverføring starter synkroniseringen på nytt igjen for hver byte.

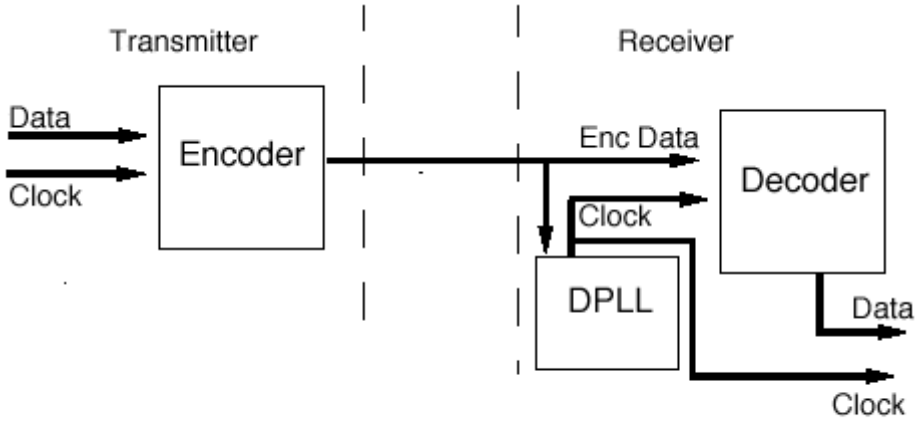


Synkron

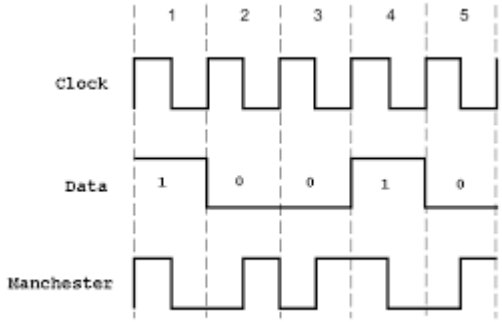
Ved asynkron overføring må det overføres (minst) to ekstra bit for hver byte som overføres. Dvs 10 bit må sendes for å få over 8 bit med data. De 2 ekstra bitene, start bit og stopp bit, må overføres ekstra for hver byte, bare for å oppnå synkronisering. Det er mye «overhead».

Ved synkron dataoverføring er disse to ekstra bitene tatt bort. Der sendes 8 bit for hver 8 bit med data. Synkroniseringen gjøres i starten av hver blokk, i stedet for i starten av

hver byte. Når synkronisering først er oppnådd, er det ingen synkronisering igjen før neste blokk. Det er da viktig at senderklokka og mottagerklokka er eksakt like, derfor overføres senderklokka sammen med dataene. Det gjøres ved å kode inn klokkesignalet inn i datastrømmen. Det gjøres i «encoder». Ut fra denne encoderen er databitene kodet om, slik at de inneholder mye klokkesignal. På mottagersiden blir dette klokkesignalet dekodet ut av datastrømmen, ofte i en DPLL. På den måten blir mottagerklokka og senderklokka eksakt like.



En mye brukt kodeform er Manchesterkode. En slik koding av databitene gjør at kravet til båndbredden på overføringsmediet blir dobbelt så stort.



Synkroniseringen kan foregå på to måter:

- Det ene er bytesynkronisering. Da er det en bestemt byte, synkroniseringsbyte, som er lagret i mottageren. I starten av blokka er det en slik synkroniseringsbyte. Mottageren sjekker det mottatte bitmønsteret med denne byten for hver bit mottatt. Da bitmønsteret er likt, er synkronisering oppnådd. Hvis denne synkbyten er en del av dataene, setter senderen inn en ekstra byte, som betyr at etterfølgende byte ikke er synkroniseringsbyte, men en databyte. Denne ekstra byten blir tatt ut på mottagersiden.

- Den andre måten er bitsynkronisering. Der ser mottageren etter et bestemt bitmønster. Når det har kommet 6 enere, etter en nuller, og det deretter kommer en nuller, er synkronisering oppnådd. Hvis det samme synkroniseringsmønsteret kommer som en del av dataene, vil senderen sette inn en ekstra 0 i datastrømmen, slik at det ikke kommer 6 1'ere etter hverandre i dataene. Denne ekstra 0'er blir tatt ut igjen ved mottageren