BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
MÉRÉSTECHNIKA ÉS INFORMÁCIÓS RENDSZEREK TANSZÉK

Digitális technika
VIMIAA01

Fehér Béla
BME MIT

A kommunikációs technológiák

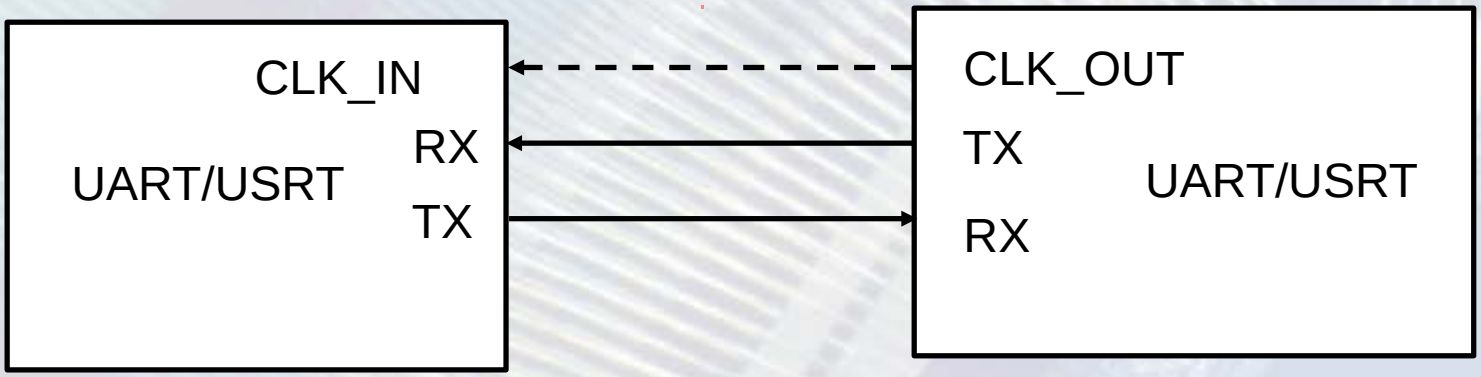
- **A mikroprocesszorok a környezetükkel folyamatos kapcsolatban állnak**
- **A környezet kettős jelentésű**
  - A rendszerben lévő közvetlen belső részegységek
  - A rendszeren kívüli állandó vagy időszakos kommunikációs kapcsolatok
- **A kapcsolatok alatt alapvetően az eszköz-eszköz ill. gép-gép közötti (M2M) kommunikációt értjük**
- **A digitális technika tárgy keretében a legegyszerűbb, leggyakoribb soros adatátviteli interfészeket tekintjük át és csak VEZETÉKES kapcsolatokkal foglalkozunk**

A kommunikációs technológiák

- **Megkülönböztetjük a belső és külső kommunikációt**
- **Belső kommunikáció:**
  - Eszközön, rendszeren belüli adatátvitel
  - Azonos logikai szintek, áramköri viselkedés
  - Kapcsolat normál jelekkel
- **Külső kommunikáció**
  - Egyedi fizikai előírások, speciális jelszintek, áramköri meghajtókra, dedikált csatlakozókra
  - Zavarvédelem kérdése, aszimmetrikus/szimmetrikus jelvezetés
  - Megbízható működési távolság

UART/USRT Soros kommunikáció

- **Univerzális aszinkron/szinkron adó vevő egység**
  - Kis sebességű adatátvitel, aszimmetrikus vonalakon
  - Nincs megkülönböztetett szerep, egyenrangú kommunikáció két egység között
  - Külön TX/RX adatvezeték, egy időben akár két irányú átvitel → full duplex kommunikáció
    - Órajel csak USRT esetében

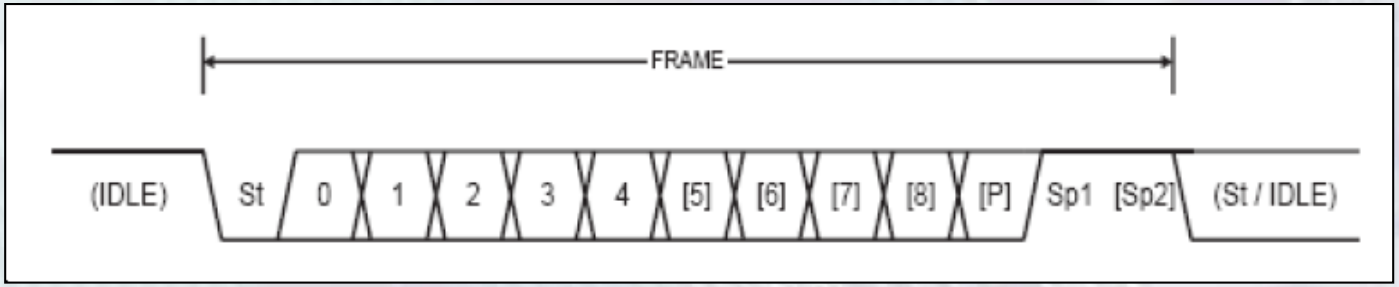


UART/USRT Soros kommunikáció

- **Univerzális aszinkron/szinkron adó vevő egység**
  - Aszinkron esetben órajel nélkül, de előre egyeztetett, szabványos bitsebességgel (2400, 4800, ..115200kb/s
    - A bitidőzítés elvárt pontossága az elfogadható biztonságú adatátvitelhez legalább 1-2%
    - Nagyobb eltérés esetén a hiba gyakoriság megnőhet
  - Szinkron esetben külön órajel adja a bitsebességet, ezért lehetne tetszőleges a sebesség → megszokás alapján tipikus a szabvány sebességek választása
    - +1 vezeték, viszont jelentősen egyszerűbb hardver felépítés

UART/USRT Soros kommunikáció

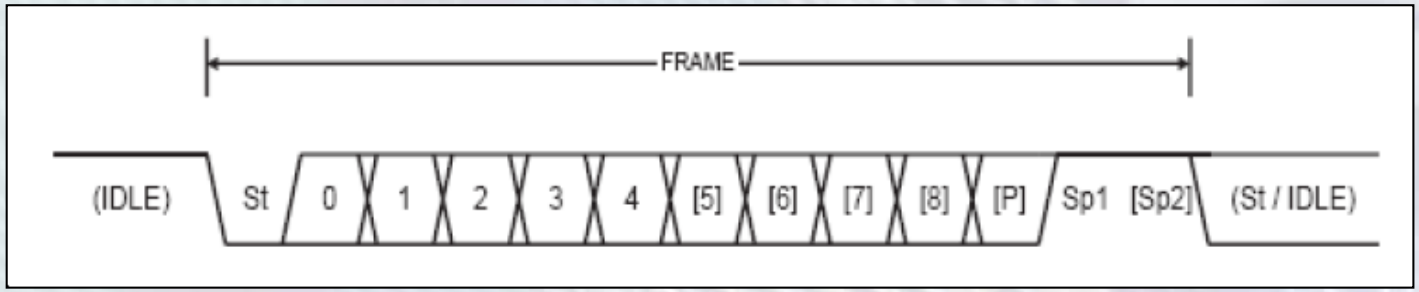
- **Univerzális aszinkron/szinkron adó vevő egység**
  - Keretezett adatátvitel: Miden bájt adatbitjeit egy-egy START és STOP bitpáros keretezi



  - 1 START + [5-6-7-8] ADAT + 1Par + 1-1,5 STOP
  - Tipikus beállítás, amit a leggyakrabban használunk: **8N1**, azaz **1 START** bit + **8 ADAT** bit + **1 STOP** bit
  - Adatbitek sorrendje: LSB → MSB (D[0], D[1], ..D[7]
  - 8 adatbit 10 bitidő alatt → bitsebesség, sávszélesség kihasználás (veszteség ↔ felismerhetőség)

UART/USRT Soros kommunikáció

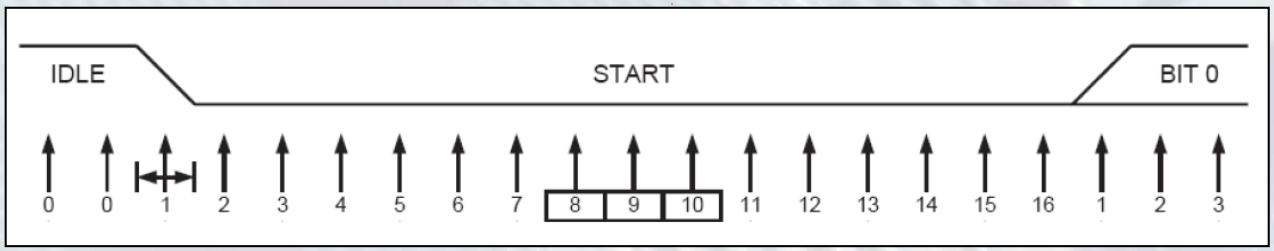
- **Aszinkron adó: Saját (pontos) órajelének ütemezése alapján kiadja a biteket**
  - Tetszőleges időben elkezdhet adni a TX vonalán
  - A STOP bit vége után azonnal kezdheti a következő bájt START bitjét → Back-to-back átvitel



  - A vevő figyeli az RX vonalát (Adó_TX ↔ Vevő_RX)
  - Ha a vonalon lefutó élet ↓ érzékel, az egy aktív átvitel elindítását jelentheti (vagy egy tüske szerű zavart!)

UART/USRT Soros kommunikáció

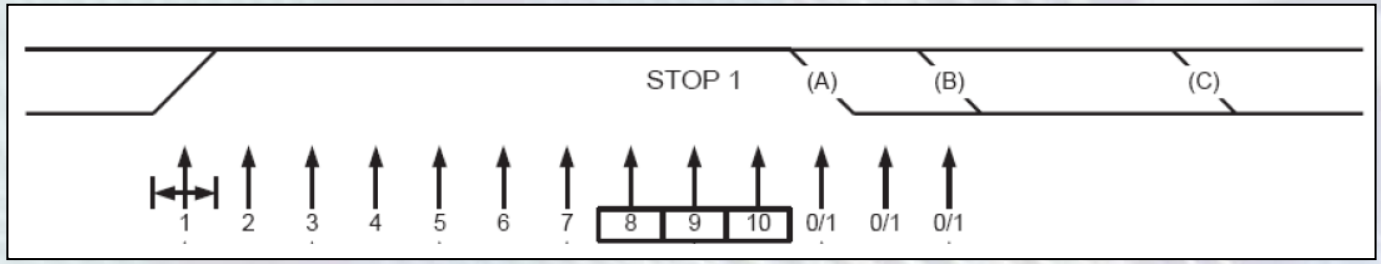
- **Aszinkron vevő: Saját (pontos) órajelének ütemezése alapján a bitfrekvencia 16-szorosával mintavételezi a vonalat → Várja START lefutó ↓ élét**
  - Ezután a fél bitideig folyamatosan mintavételez, vagy a START bit közepén végrehajt néhány mintavételt→ Csak stabil START jelre szabad indulni



  - Ha a START bit nem stabil, a vételt leállítja
  - Hasonlóan a többi bitnél is lehet 3x-os mintavételt használni → javíthatja a zavarérzékenységet

UART/USRT Soros kommunikáció

- **Nem foglalkozunk minden részlettel**
  - Stabil START jel után az adatbiteket belépteti egy shift regiszterbe (és ellenőrzi a paritást, ha van)
  - Mintavételezi a STOP bit értékét is
  - Ha STOP = 1, akkor a vétel érvényes



  - A fogadott adatbájt tovább adható és a vevő azonnal újra vételi módra állhat
  - Az időzítések esetleges kisebb hibái kezelhetők

UART/USRT Soros kommunikáció

- **UART adó/vevő tervezése bonyolult**
- **Az adó egyszerű**
  - 9/10 bites tölthető SHR + ütemező/léptető jel
- **A vevő már összetettebb rendszer**
  - Szokásos módszerünkkel: Adatstruktúra + vezérlő
  - Adatstruktúra: 8/9 bites shiftregiszter, 3 vagy 4 bites bitszámláló, vevőben mintavételi számláló
  - Vezérlő: IDLE-START-VÉTEL-(PAR)-STOP végén RDY vagy ERR, majd újra IDLE állapotok
  - Az ERR esetén nem sokat tehetünk, a hibakezelés a magasabb rétegek feladata.
    - Nem jött meg az összes bájt
    - Rossz az ellenőrzőösszeg

UART/USRT Soros kommunikáció

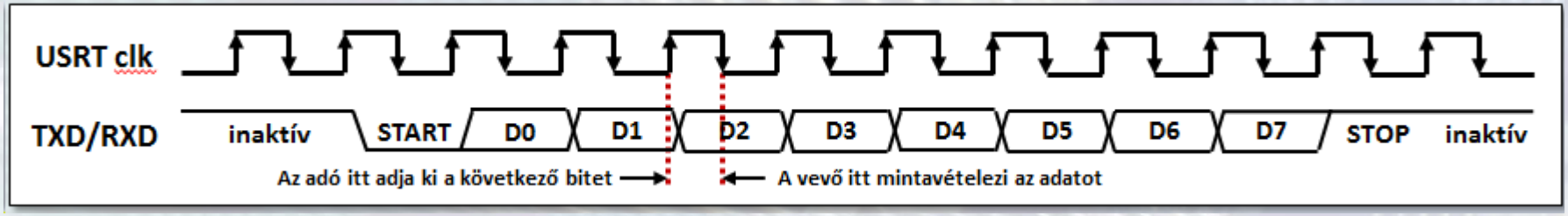
- **UART átvitel időviszonyai**
- **Beállítás: 8N1 (10 bit) Bitidő számítása: 1/bitsebesség**

Bit/s	2400	4800	9600	19200	38400	57600	115200
T_{bit}	417us	208us	104us	52us	26us	17,4us	8,68us
$T_{karakter}$	4,17ms	2,08ms	1,04ms	0,52ms	0,26ms	0,174ms	0,087ms

- **Pl. 9600 b/s sebességnél 1 karakter átviteli ideje (1ms) alatt a MiniRISC kb. 5300 utasítást hajt végre**
- **UART/USRT Periféria kezelés?**
  - Lekérdezéssel sok időveszteség
  - Megszakítással esetleg túl gyakori
  - Speciális kiegészítés a soros adó/vevőkhöz: **FIFO** Kisméretű (pl. 16 bájt) adatpuffer

USRT Soros kommunikáció

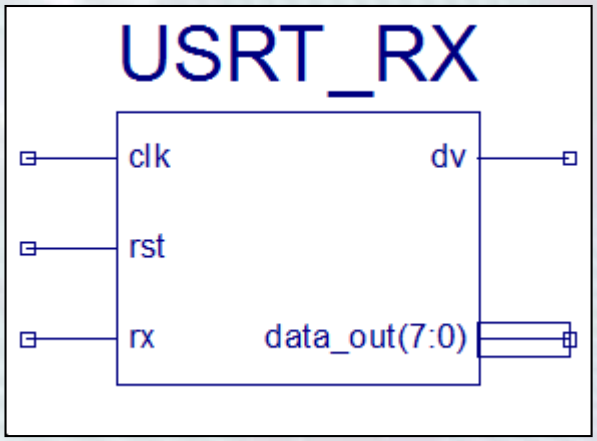
- **USRT paraméterek ugyanazok, mint UART esetén**
  - A MiniRISC rendszerben ezt használjuk a PC terminál felé (8N1, 9600b/s)
  - Átviteli jelek és vezérlésük



  - Az órajelet bármelyik egység adhatja, valójában csak szinkronizációs célokat szolgál
  - Az USRT adó vevő hardver realizációja egy-egy tölthető, engedélyezhető shiftregiszter

USRT Soros kommunikáció

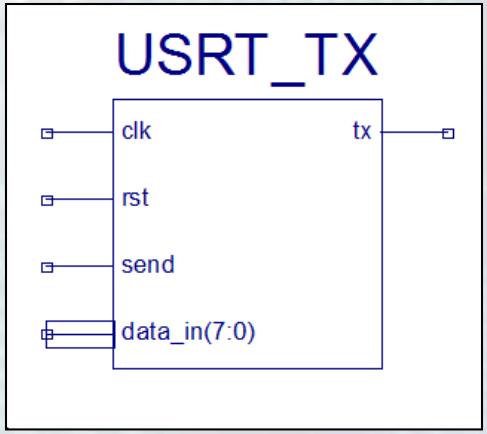
- **Egyszerű USRT vevő Verilog HDL kódja**
  - A 8 adatbittel együtt a START és STOP biteket is beléptetjük
  - Adat érvényes feltétel: DV = ~START & STOP




```
`timescale 1ns / 1ps
module USRT_RX(
   input clk,
   input rst,
   input rx,
   output dv,
   output [7:0] data_out
   );

   reg [9:0] shr;                          // USRT vételi shift regiszter keret + info
   wire start;
   wire stop;

   assign start = shr[0];                  // Balról léptetünk be LSb-vel kezdődő adattal
   assign stop = shr[9];
   assign dv = ~start & stop;              // Ha a keret helyes, akkor jó az adat

   assign data_out = shr[8:1];             // A 10 bitból 8 bit az adat

   always @(posedge clk)
      if (rst)
         shr <= 10'b1_1111_1111_1;         // Csupa 1 bit, alapállapot
      else if (dv)                         // Az előző karakter után, azonnal jöhet az új
               shr <= {rx, 9'b1_1111_1111}; // karakter, ezért azonnal rx beléptetés
           else
               shr <= {rx, shr[9:1]};      // Beléptetés balról, ....D2D1D0START


endmodule
```

USRT Soros kommunikáció

- **Egyszerű USRT adó Verilog HDL kódja**
  - A 8 adatbittel együtt a START és STOP biteket is betöltjük
  - Ezután jobbra léptetve kiléptetjük a teljes karaktert, újra csupa „1” értékű bittel feltöltve




```
`timescale 1ns / 1ps
module USRT_TX(
    input clk,
    input rst,
    input send,
    input [7:0] data_in,
    output tx
    );

   reg [9:0] shr;                          // USRT adó shift regiszter

   always @(posedge clk)
   if (rst)
      shr <= 10'b1_1111_1111_1;            // Csupa 1 inaktív tartalom
   else
      if (send)                            // Küldés parancs pulzus
         shr <= {1'b1, data_in, 1'b0};     // betölti a STOP-8D-START bitmintát
      else
         shr <= {1'b1, shr[9:1]};          // majd jobbra kilépteti
                                           // sorrendben
   assign tx = shr[0];                     // Kimeneti TX jel

endmodule
```

UART használata

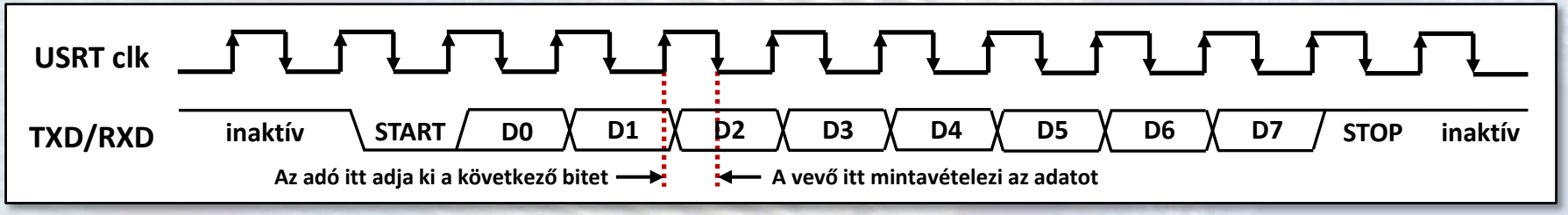
- **Külső adatkapcsolatokban**
  - Nem használhatók a közvetlen logikai jelek
  - Követelményektől függően speciális meghajtók és vezetékezés
- **RS232 szabvány:**
  - Aszimmetrikus jelvezeték,
  - ± 5 …±15 V jelszintek („1” = negatív, „0”=pozitív)
  - Távolság 10..100 m, 19,6kb/s
- **RS485/RS422 szabvány:**
  - Szimmetrikus jelvezeték
  - +5V feszültség ellenfázisú vezérlés Out_A – Out_B
  - Távolság akár 1200m, 10Mb/s

MiniRISC USRT periféria használata

***slave_usrt:* USRT soros kommunikációt biztosító periféria**

USRT adatátvitel

- Keretezett formátum:
  - 1 START bit, melynek értéke mindig 0
  - 8 adatbit (D0 – D7)
  - 1 STOP bit, melynek értéke mindig 1
- USRT órajel: az adatátviteli sebességet határozza meg
  - A master egység adja ki a slave egység felé
- Adás: a bitek kiadása az USRT órajel felfutó élére történik
- Vétel: a mintavételezés az USRT órajel lefutó élére történik
  - Csak a kerethiba mentes (STOP bit = 1) karakterek kerülnek tárolásra



MiniRISC USRT periféria használata

slave_usrt: **USRT soros kommunikációt biztosító periféria**

Vezérlő regiszter (UC): BASEADDR + 0x00, írható és olvasható

7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
0	0	0	0	RXCLR	TXCLR	RXEN	TXEN
R	R	R	R	W	W	R/W	R/W

Bit	Mód	Funkció
TXEN	R/W	0: az USRT adó tiltott 1: az USRT adó engedélyezett
RXEN	R/W	0: az USRT vevő tiltott 1: az USRT vevő engedélyezett
TXCLR	W	1 beírásának hatására az adási FIFO törlődik
RXCLR	W	1 beírásának hatására a vételi FIFO törlődik

Adatregiszter (UD): BASEADDR + 0x03, írható és olvasható

- Írás: adat írása az adási FIFO-ba (ha TXNF=1)
- Olvasás: adat olvasása a vételi FIFO-ból (ha RXNE=1)

7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

MiniRISC USRT periféria használata

slave_usrt: **USRT soros kommunikációt biztosító periféria**

FIFO státusz regiszter (US): BASEADDR + 0x01, csak olvasható

7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
0	0	0	0	RXFULL	RXNE	TXNF	TXEMPTY
R	R	R	R	R	R	R	R

Megszakítás eng. regiszter (UIE): BASEADDR + 0x02, írható és olvasható

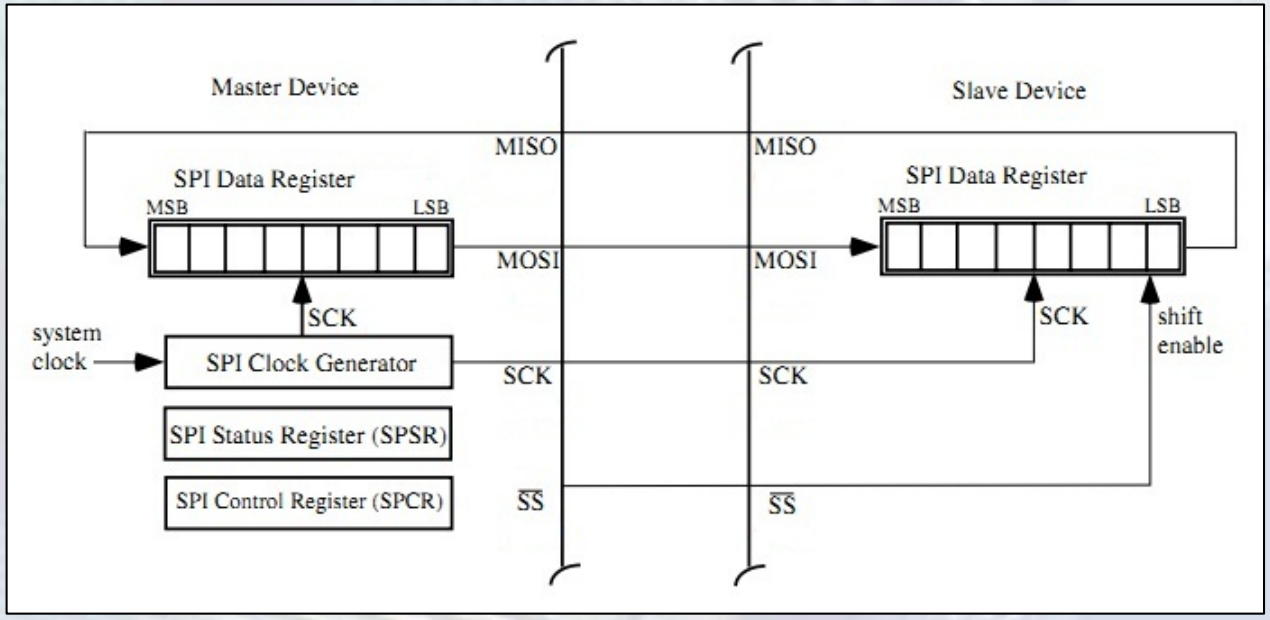
- A FIFO státusz megszakítások engedélyezhetők/tilthatók vele
- A megszakításkérés az engedélyezett események fennállásáig aktív

7. bit	6. bit	5. bit	4. bit	3. bit	2. bit	1. bit	0. bit
0	0	0	0	RXFULL	RXNE	TXNF	TXEMPTY
R	R	R	R	R/W	R/W	R/W	R/W

Bit	Jelentés	
TXEMPTY	0: az adási FIFO-ban van adat	1: az adási FIFO üres
TXNF	0: az adási FIFO tele van	1: az adási FIFO nincs tele
RXNE	0: a vételi FIFO üres	1: a vételi FIFO-ban van adat
RXFULL	0: a vételi FIFO nincs tele	1: a vételi FIFO tele van

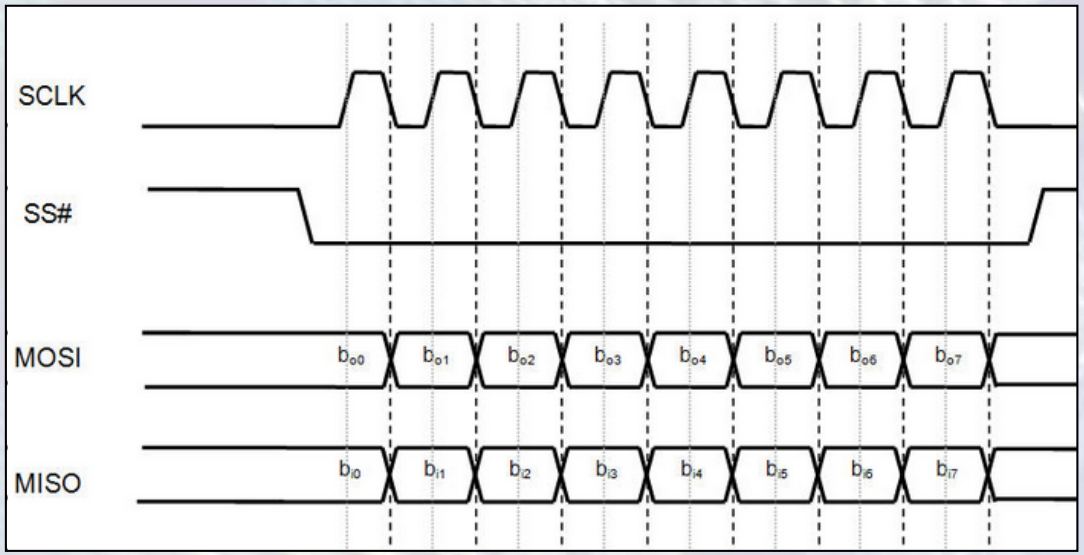
SPI soros protokoll

- **Az SPI soros periféria interfész kizárólag rendszeren belül használható**
  - Integrált áramkörök, részegységek között
- **4 vezetékes szinkron interfész:**
  - CSn, SCK, MOSI, MISO
  - Egyidejű, szimultán adatátvitel (szinkron full-duplex)



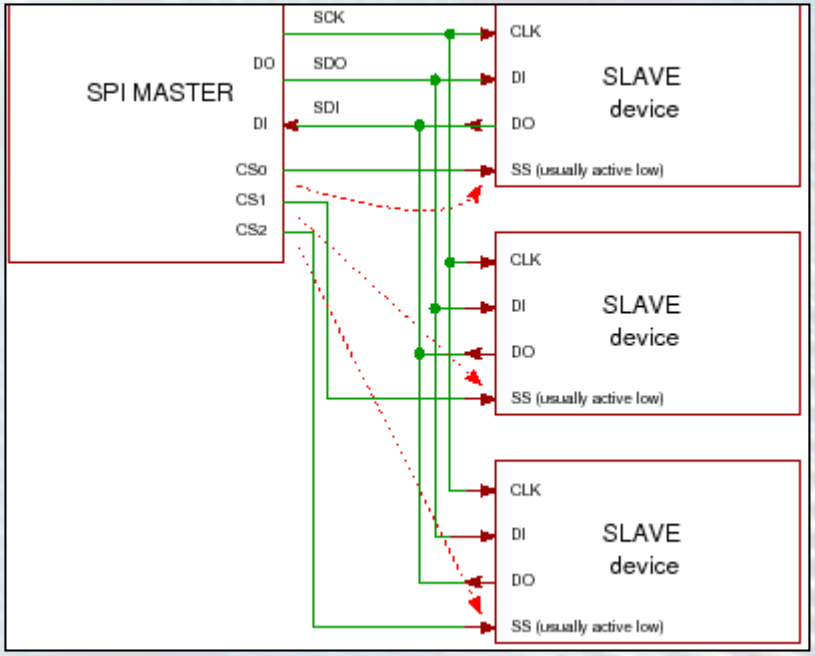
SPI soros protokoll

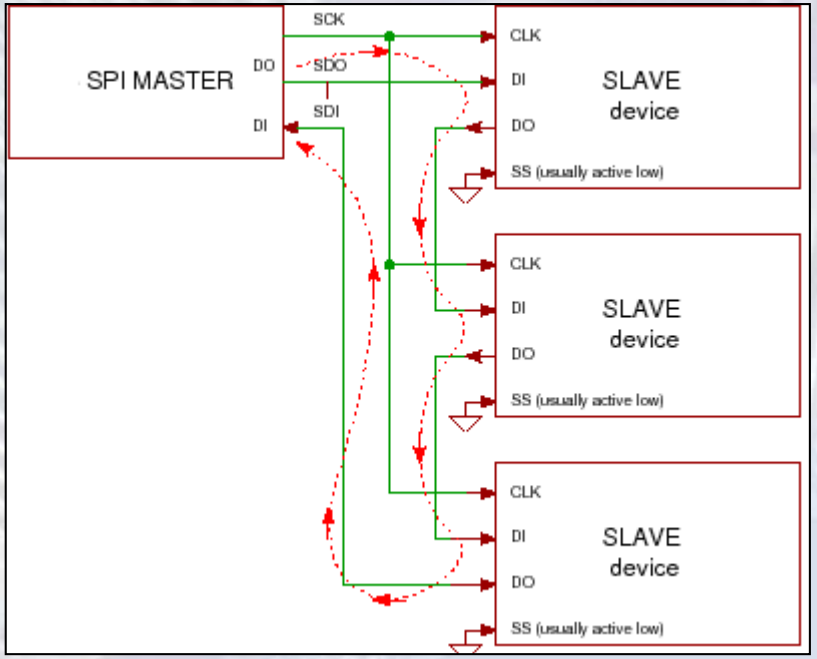
- **Az egyszerű modell szerint „sorosan csatolt” shiftregiszterekből álló hurok**
  - Közös órajelre, szinkron működés
    - 4 lehetséges működési opció az órajel polaritása és az aktív órajel él választása alapján
    - Az SPI periféria egységeket ezekhez tetszőlegesen konfigurálhatjuk (itt most nem ismertetünk minden verziót)



SPI soros protokoll

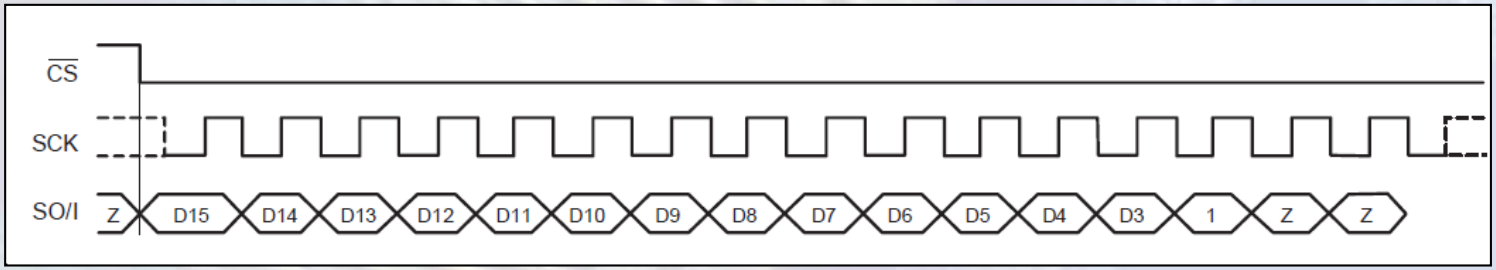
- **Az SPI interfész busz kommunikációt is biztosít**
- **A tipikus rendszer 1 MASTER – több SLAVE**
- **Két lehetséges topológia**
  - Csillag topológia
    Független elérés
  - Kaszkád, lánc topológia
    Egyetlen adatlánc





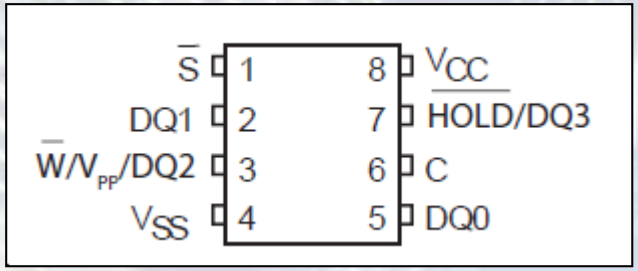
SPI soros protokoll

- **Az SPI interfész kialakítása következtében viszonylag nagy átviteli sebességre képes a soros interfészek közt**
  - SCK akár néhány MHz is lehet
  - SPI esetén nincsenek „szabvány” bitsebességek
  - Adatméret nem korlátozott 8 bitre, aktuális igények szerint egy-egy ciklusban akár 16-24-32 bit is átvihető
  - Gyakran használják csak egyirányú kommunikációra
    - Pl. A/D átalakító, szenzorok: csak bemenet
    - Pl. D/A átalakító, kijelzők: csak kimenet
    - Ilyenkor egyszerűen az adott számú adatbit szinkron ki vagy beléptetése történik (pl. TMP121 hőmérő szenzor)



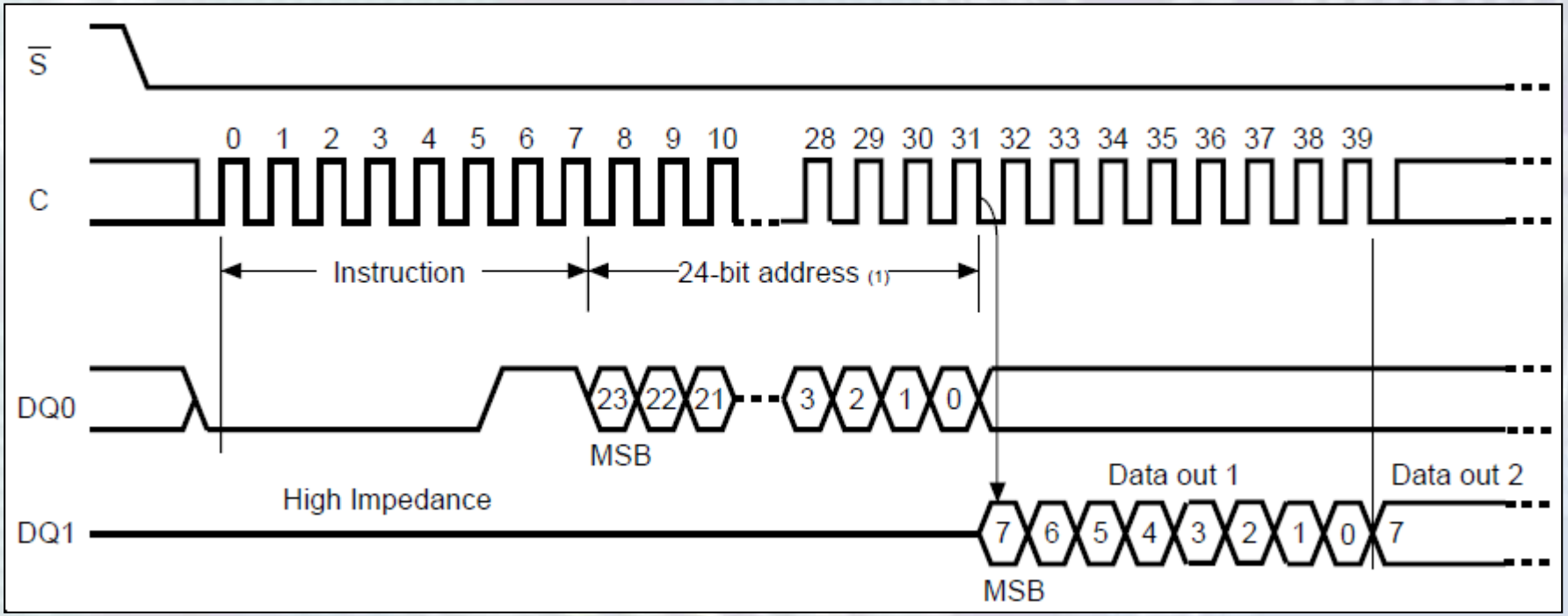
SPI soros protokoll

- **Az SPI interfész fontos a flash memóriák használatánál**
- **Nagy kapacitás, viszonylag kevés láb, (kis fizikai méret) és elfogadható sebesség**
- **Gyakran „kibővített, Dual/Quad (2/4 vezetékes) SPI**
- **Pl. N25Q128 128Mbit SPI flash memória**
  - 108 MHz SPI órajel
  - Hagyományos SPI és 2 vagy 4 vezetékes SPI I/O
  - Kis méretű, 8 lábú tokozás, jelentős I/O sávszélesség
  - Szokásos IC lábak több funkciót valósítanak meg!
  - Extra adatátviteli lehetőség ~400Mb/s !



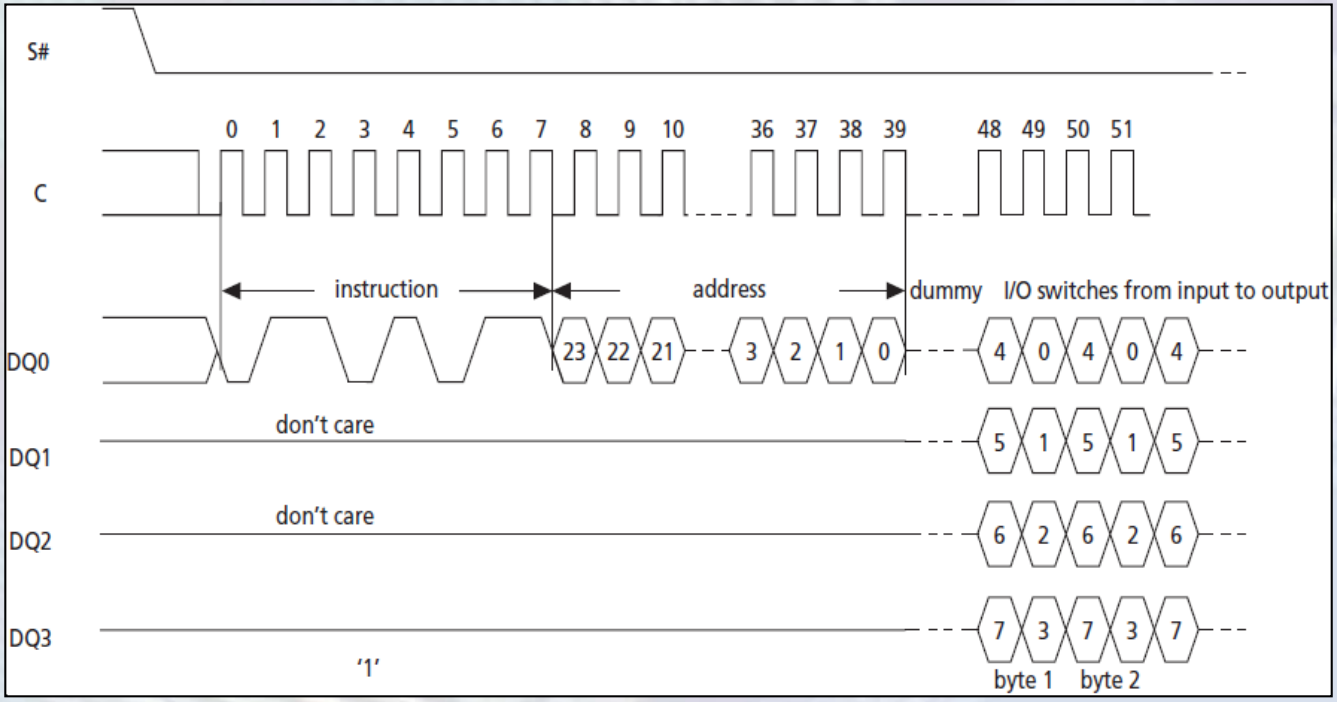
SPI soros protokoll

- **Hagyományos SPI protokoll soros memóriáknál**
  - Parancs, cím kivitel DQ0 outputon
  - Adat beolvasás DQ1 inputon
    - Erre minden mikroprocesszor képes, esetleg <100MHz-en



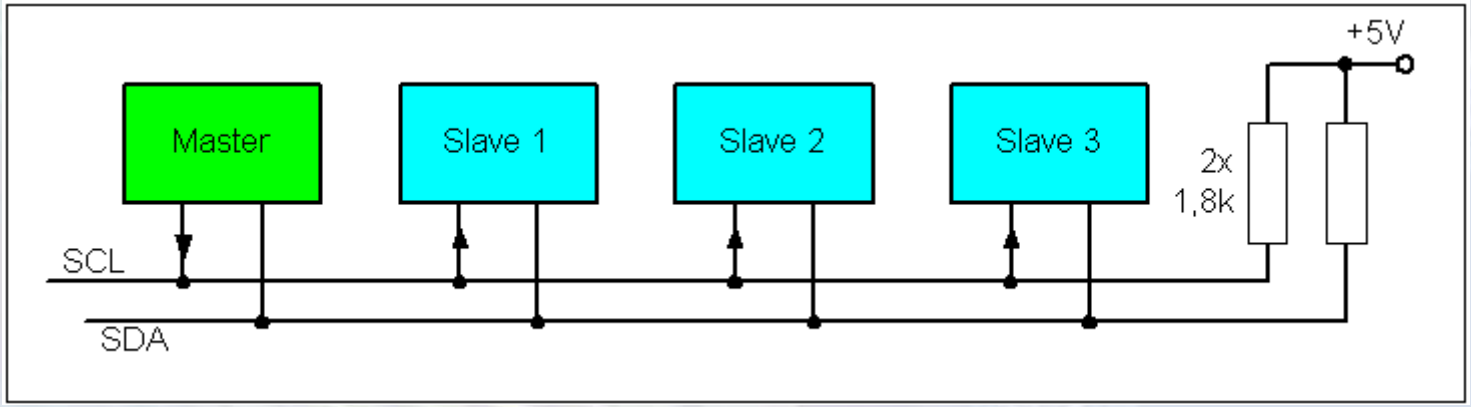
SPI soros protokoll

- **Kiterjesztett SPI protokoll soros memóriáknál**
  - Parancs, cím kivitel D0 outputon (mint előbb)
  - Adat beolvasás DQ0, DQ1, DQ2, DQ3 inputon
    - DQ0 I/O irányt váltott!
    - Ezt nem minden mikroprocesszor támogatja még!



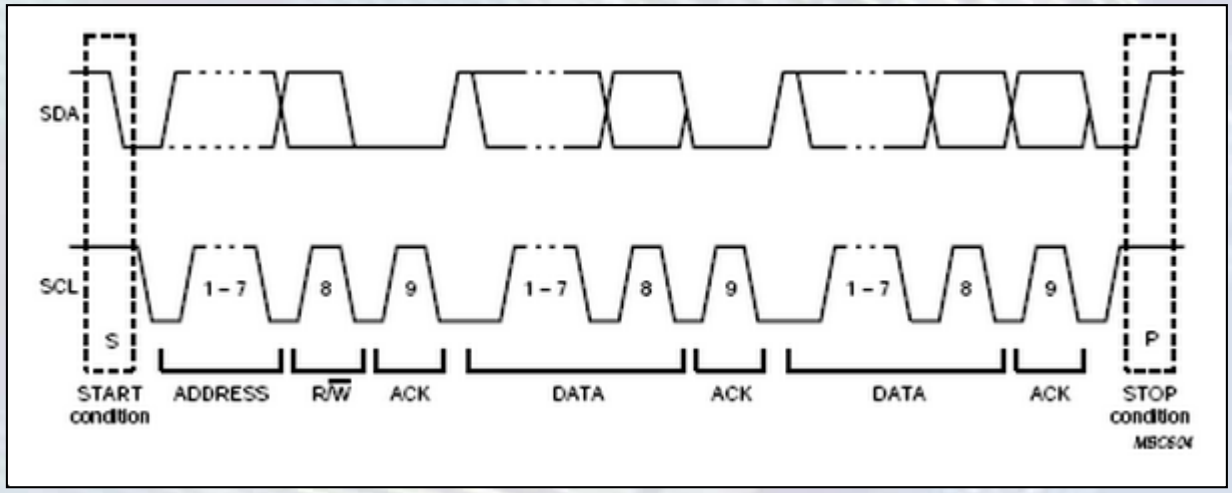
I2C soros protokoll

- **I2C: Inter Integrated Circuit: Integrált áramkörök közötti kommunikáció (SMBus, PMBus, DDC)**
- **Nagyon elterjedt, a legszélesebb körben használják**
- **Teljes, szabványos protokoll**
  - Két vezetékes (SCL, SDA), valódi busz kommunikáció
  - Több MASTER- több SLAVE
  - Viszonylag lassú (100kHz-400kHz-1MHz-)
  - Alkalmazása: szenzorok, memóriák, stb.
  - I2C alapú protokollok: SMBus, PMBus, IPMI, DDC



I2C soros protokoll

- **I2C adatátvitel**
- **Speciális START és STOP feltétel**
- **SCL és SDA is kétirányú**
- **MASTER kezdeményez, és a SLAVE válaszol, de kérhet várakozást, ha lassabb a belső működése**
- **Bájtonkénti nyugtázás a teljes átvitel során**
- **Bonyolult protokoll, nem tárgyaljuk**
- **DE kis sebességen GPIO porton is lejátszható**



Digitális technika
13. EA vége