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

Digitális technika
VIMIAA02
13. hét

Fehér Béla, Benesóczky Zoltán
BME MIT

A kommunikációs technológiák

- **A mikroprocesszorok a környezetükkel folyamatos kapcsolatban állnak**
- **A környezet kettős jelentésű**
  - A rendszerben lévő közvetlen belső részegységek (készüléken belül)
  - A rendszeren kívüli állandó vagy időszakos kommunikációs kapcsolatok (készülékek közötti)
- A digitális technika tárgy keretében a legegyszerűbb, **leggyakoribb soros adatátviteli interfészeket** tekintjük át és **csak VEZETÉKES kapcsolatokkal** foglalkozunk

A kommunikációs technológiák

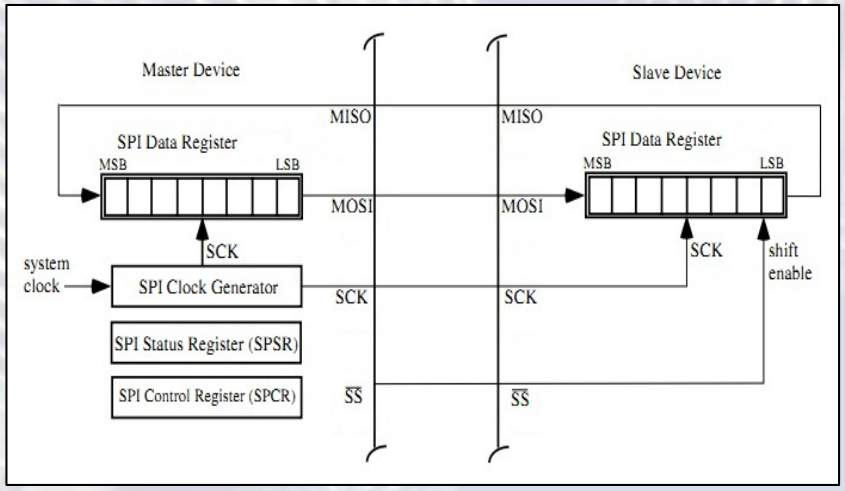
- **Megkülönböztetjük** a **belső** és **külső** kommunikációt
- **Belső kommunikáció:**
  - Készüléken, nyomtatott áramkörön (NYÁK) belüli, kis távolságú adatátvitel (pl. mikrokontroller és szenzor között)
  - Általában azonosak a logikai szintek (a low és hi szintekhez tartozó feszültség) ezért többnyire nem kell szintillesztés
  - Megkülönböztetünk **szinkron** és **aszinkron** kommunikációt:
    - **Szinkron** (órajel jelvezeték is van): SPI, I2C, USRT
    - **Aszinkron**: nincs külön órajel jelvezeték: UART

A kommunikációs technológiák

- **Külső kommunikáció**
  - **Speciális jelszintek** (speciális meghajtó és vevő áramkörök kellenek)
  - Egyedi fizikai előírások és csatlakozók
  - **Nagy távolság esetén** különösen fontos a **zavarvédelem**
  - A **megbízható működési távolság** a zavarvédelemtől is függ (pl. a speciális meghajtó és vevő áramkörök tulajdonságaitól), akár >1km is lehetséges.

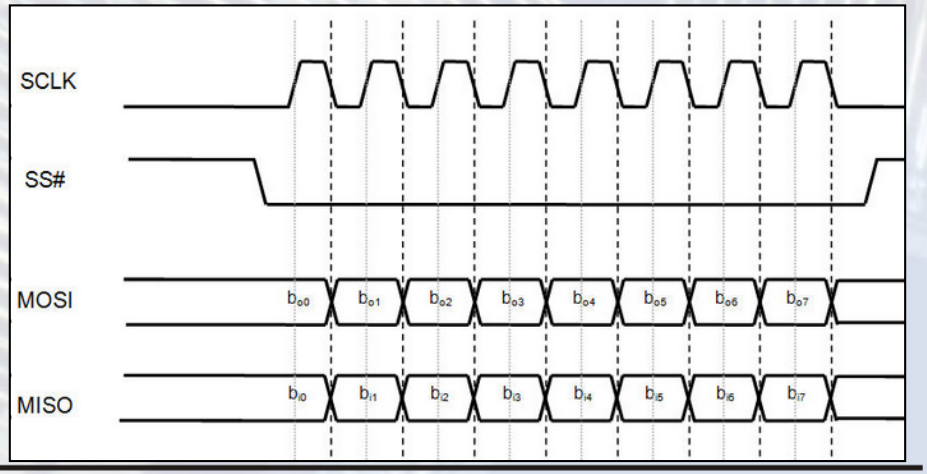
SPI soros protokoll

- **Az SPI soros periféria interfész kizárólag rendszeren belül használható, IC-k, részegységek között**
- **4 vezetékes szinkron interfész, a Master vezérli a kommunikációt a SLAVE-el:**
  - **/SS** (Slave select), **SCK** (Serial clock), **MOSI** (Master Output Slave Input), **MISO** (Master Input Slave Output)
  - **Egyszerre 2 irányú** (full-duplex, adás és vétel) kommunikáció.
  - **A master shiftregisztere és a slave shiftregisztere** sorba kapcsolódnak és együtt **egy hurkot alkotnak**. Ebben **a master és a slave adata 8 órajel alatt helyet cserél**.
  - **/SS=0 választja ki a slave-et**, a **slave 3 állapotú MISO kimenete ennek hatására engedélyeződik**.
  - Ezután kezdődhet meg a kommunikáció.



SPI soros protokoll

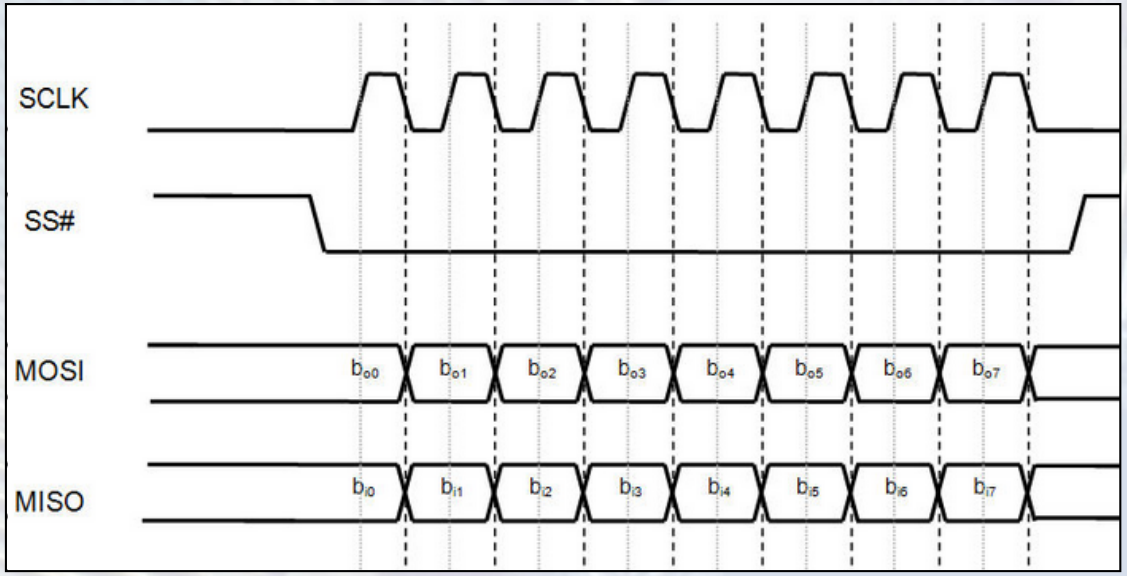
- **Az egyszerű modell szerint „sorosan csatolt” shiftregiszterekből álló hurok**
  - Közös órajelre, szinkron működés
    - Mindkét órajel élet használja. A shiftregiszterek **kimenetén az újabb adat bit** megjelenése **ellentétes élre történik**, mint a **bemenetén a mintavételelezés**.
    - 8 lehetséges konfiguráció van, az SPI perifériát ezekhez tetszőlegesen konfigurálhatjuk (nem ismertetünk minden verziót).
      - **az órajel kezdeti szintje:** (low vagy high),
      - **az aktív kimeneti órajel él:** (lelfutó, felfutó)
      - **a bitek sorrendje:** MSB-vel vagy LSB-vel kezdődik
    - A **master SS=0-val kijelöli a slave-t, 8** (vagy több) **órajel alatt az adatok átshiftelődnek a masterből a slave-be és fordítva.**



SPI soros protokoll

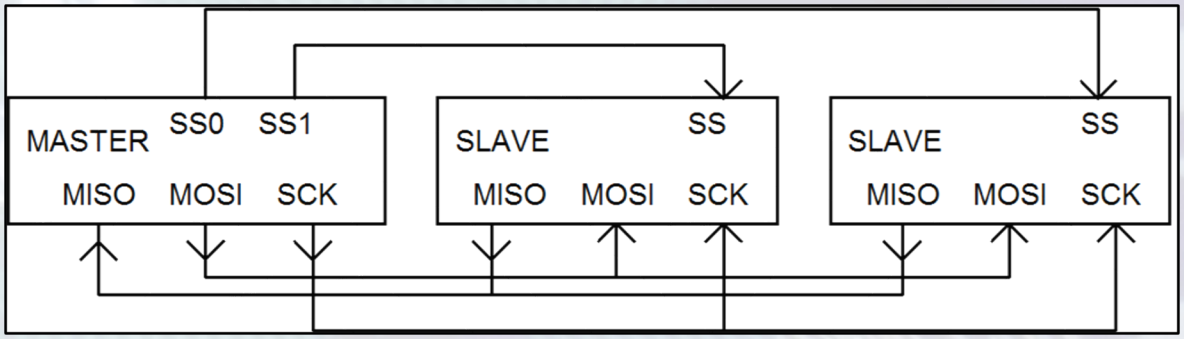
Az alábbi példában:

- Az **órajel kezdeti szintje low.**
- Az órajel **lelfutó élére változik az adat** és **felfutó élnél** íródik be (mintavételeződik) a shiftregiszterekbe.
- A bitek sorrendje LSB-vel kezdődik
- **Az átvitel után a master** /SS=1-gyel **megszünteti a slave kiválasztását**.



SPI soros protokoll

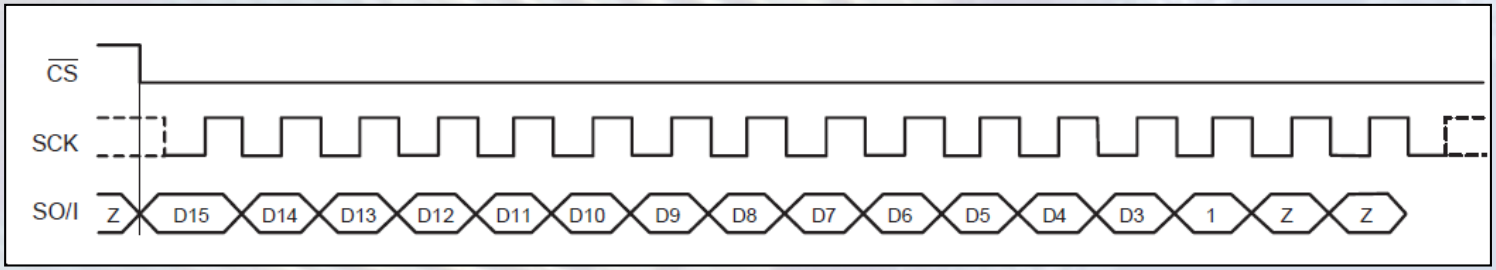
- **Az SPI interfész busz kommunikációt is biztosít**
- A tipikus rendszer **1 MASTER – több SLAVE**



- **/SSi 0-ba állításaval választja ki a master azt a slave-et**, amellyel kommunikálni akar (SSi alacsony aktív). A **kiválasztott slave MISO kimenete aktivizálódik, a többi slave MISO-ja 3-adik állapotban** marad.
- **Az SPI adatregiszterébe írásra indul a kommunikáció**.
- Ennek hatására **8 (vagy több) órajel alatt a masterbe és a slave adata „helyet cserél”.**
- A kommunikáció a kiválasztott slave SSi-jének 1-be állításával fejeződik be. A **standard SPI kommunikáció 8 órajel alatt** zajlik le.

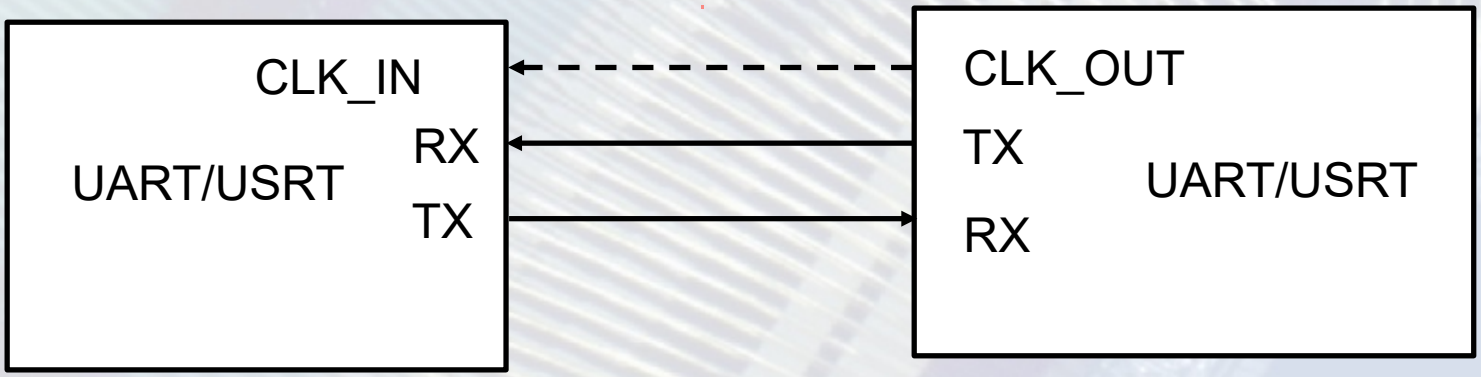
SPI soros protokoll

- **Az SPI interfész kialakítása következtében viszonylag nagy átviteli sebességre képes a soros interfészek közt**
  - SCK akár néhányszor 10 MHz is lehet
  - Adatméret nem korlátozott 8 bitre, aktuális igények szerint a 8 bit egész számú többszöröse is lehet.
  - 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)



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 adatvezetékek, egy időben akár két irányú átvitel → full duplex kommunikáció
    - Órajel csak USRT esetében

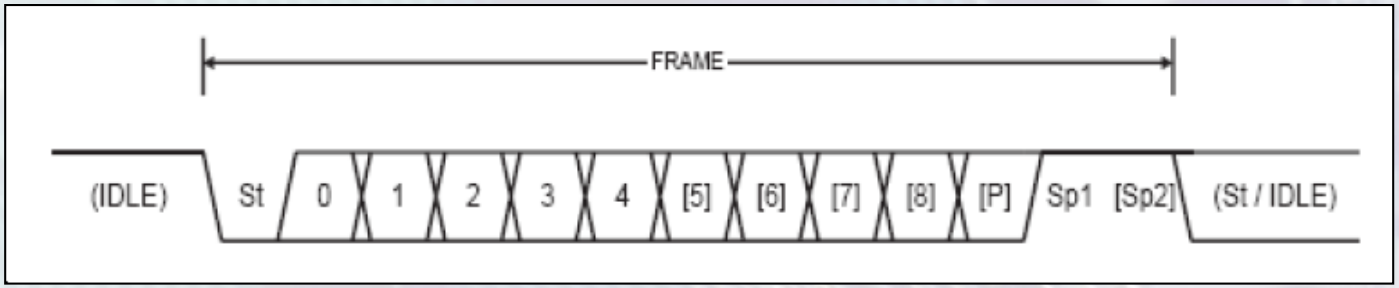


UART Soros kommunikáció

- **Univerzális aszinkron adó vevő egység**
  - Aszinkron esetben ***órajel nélkül***, de előre egyeztetett, szabványos bitsebességgel (2400, 4800, ..115200 b/s)
  - Készüléken belüli kommunikáció esetén közvetlenül összeköthetők.
  - Külső kommunikáció esetén a zavarvédelem miatt speciális meghajtókra és vevőkre (asszimetrikus vagy differenciális, szintkonvertálás).

UART/USRT Soros kommunikáció

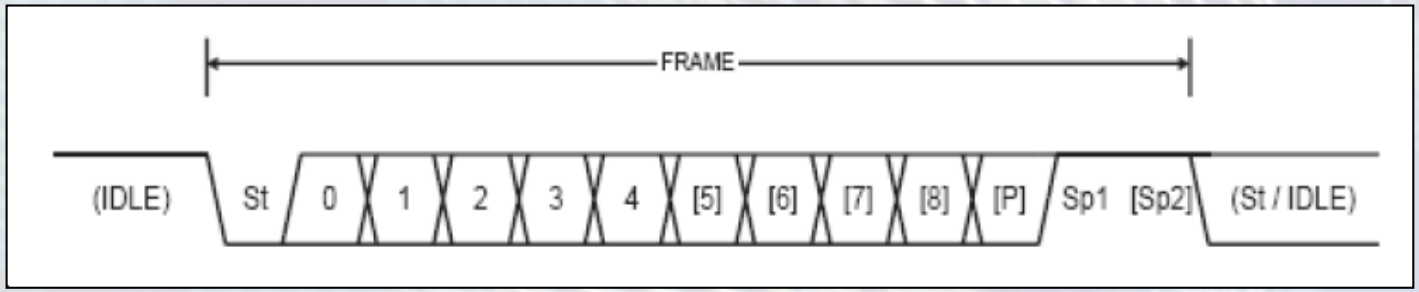
- **Univerzális aszinkron/szinkron adó vevő egység**
  - **Keretezett adatátvitel**: Minden bájt adatbitjeit egy-egy **START** és **STOP** bitpáros keretezi (FRAME)



  - **1 START** + [5-6-7-**8**] **ADAT** + (1**P**ar/**N**o par.) + **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)

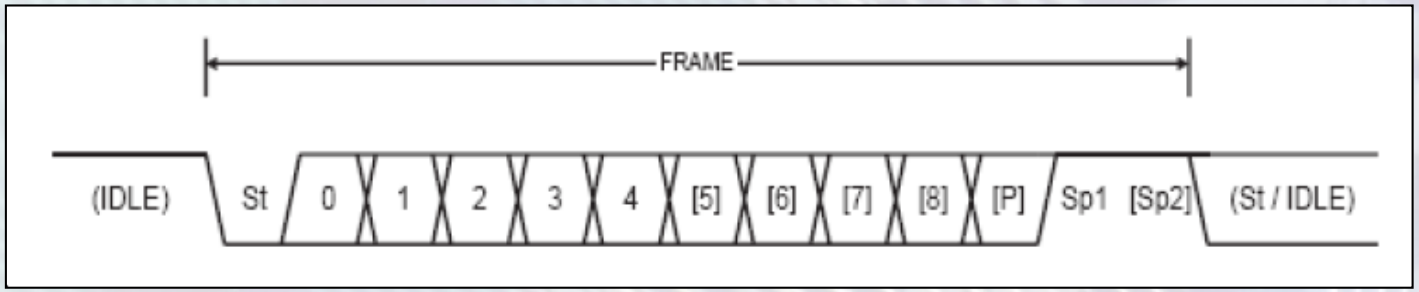
UART Soros kommunikáció

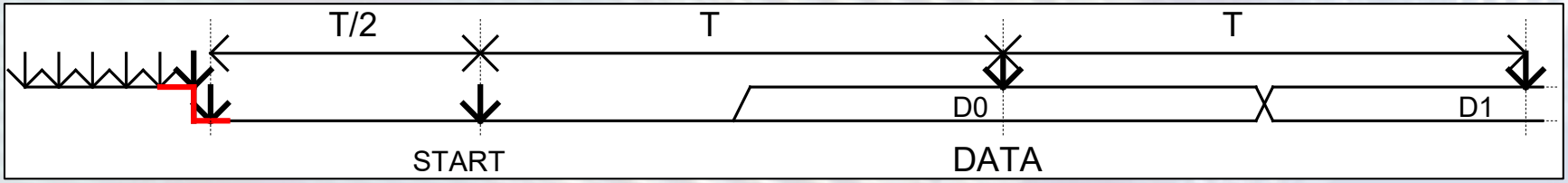
- **Aszinkron adó:** Saját (pontos) órajelének ütemezése alapján kiadja a biteket
  - Tetszőleges időben elkezdhet adni a TX vonalán
  - A STOP bit vége után azonnal kezdheti a következő bájt START bitjét



UART Soros kommunikáció

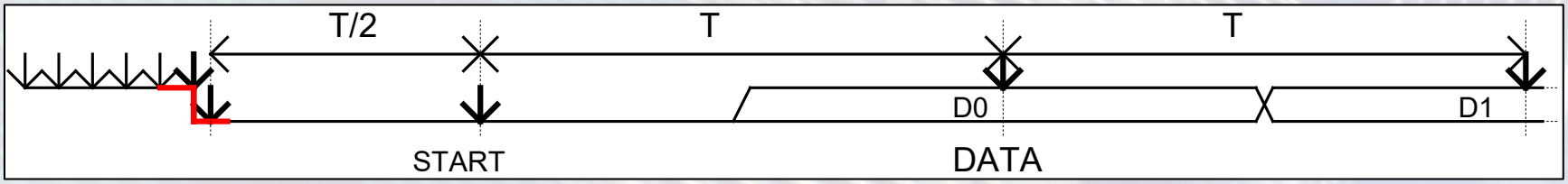
- Aszinkron vevő: Saját (pontos) órajelének ütemezése alapján **a bitfrekvencia 16-szorosával mintavételezi az RX** jelet → Várja START lefutó ↓ élét
- Ha **lefutó élet ↓ érzékel, az egy aktív átvitel elindítását jelentheti** (vagy egy tüske szerű zavart!)





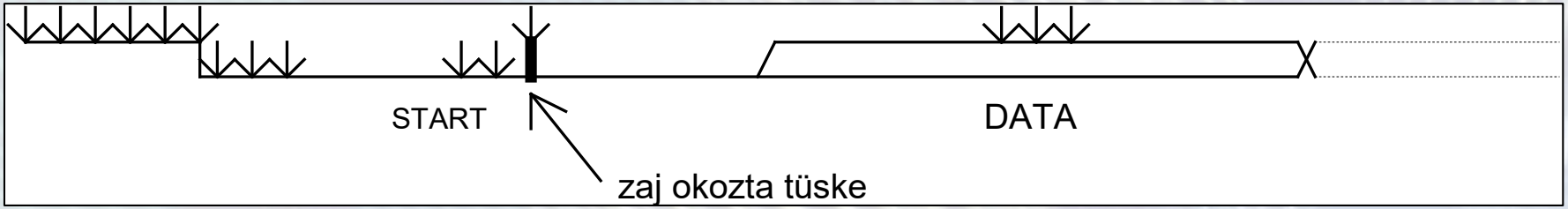
UART Soros kommunikáció

- Ezután a **fél bitideig vár** és a START bit közepén mintavételez. Ha a START bit közepe nem 0 (zaj), akkor újra a START lefutó élére vár. Ha a START bit rendben, akkor ezután bitidőnként, a bitidő közepénél mintavételez amíg az összes bit (a STOP bit is) be nem jön. Ezután újra a START bitre vár. (Lefutó él és fél bitidő múlva 0 mintavétel.)



UART Soros kommunikáció

- A **zavarok** (zaj) hamis START és egyéb bit érzékelést okozhatnak.
- **Zavarvédelem**: 3 egymást követő mintavétel alapján **többségi szavazással**. Ha legalább 2 minta azonos a 3-ból, azt elfogadja. (Pl. Az alábbi tüske zaj nem okoz hibát.)



UART Soros kommunikáció

- **A megvalósítás** (nem megyünk részletekbe)
  - Stabil START jel után **az adatbiteket belépteti egy shiftregiszterbe**
  - Ellenőrzi a paritást, ha van. A paritás hibát (**parity error**) státus regiszteben jelzi.
  - Mintavételezi a STOP bit értékét is
  - Ha **STOP = 1**, akkor **a vétel érvényes,** egyébként hibás. A STOP bit hibát a státus regiszterben jelzi (**framing error**)
  - A fogadott adatbájtot átírja egy regiszterbe vagy FIFO-ba és a vevő azonnal újra vételi módra áll. **A vett karaktert a státus regiszterben jelzi**.

UART Soros kommunikáció

- Az adó és vevő saját órajeléből előállított bitsebességek csak kicsit eltérhetnek el az előírt frekvenciától. Túl nagy eltérés a hibához, bitvesztéshez vezethet. (Pl. STOP bit helyett még az előtte levő bitet mintavételezi.)

- A bitidőzítés elvárt pontossága az elfogadható biztonságú adatátvitelhez legalább 1-2%

- Nagyobb eltérés esetén a hiba gyakoriság megnőhet.

UART/USRT Soros kommunikáció

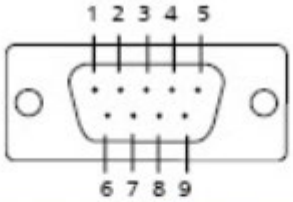
- **UART átvitel időviszonyai**
- **Beállítás: 8N1 (10 bit) Bitidő számítása: 1/bitsebesség**

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 nem célszerű, mert sok időveszteség
  - Megszakítással, ez esetleg túl gyakori, de segít a FIFO
  - Speciális kiegészítés a soros adó/vevőkhöz: **FIFO** Kisméretű (pl. 16 bájt) adatpuffer

UART használata

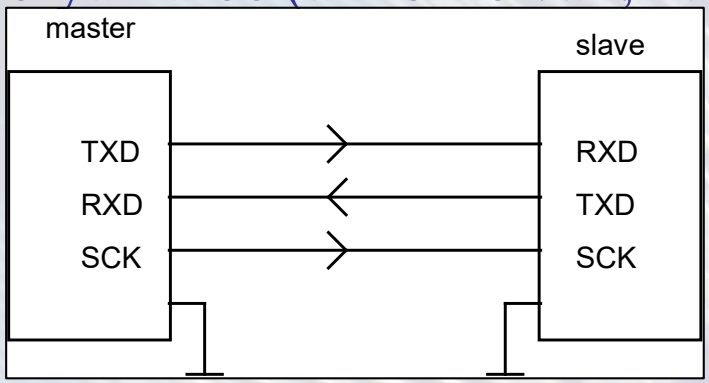
- **UART külső adatkapcsolatokban**
  - Nem használhatók a közvetlen logikai jelek
  - Követelményektől függően speciális meghajtók és vezetékezés.
- **RS232 szabvány:**
  - Aszimmetrikus jelvezeték,
  - ± 5 …±15 V jelszintek („1” = negatív, „0”=pozitív)
  - Távolság 10..100 m, 19,6kb/s
- **RS485/RS422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

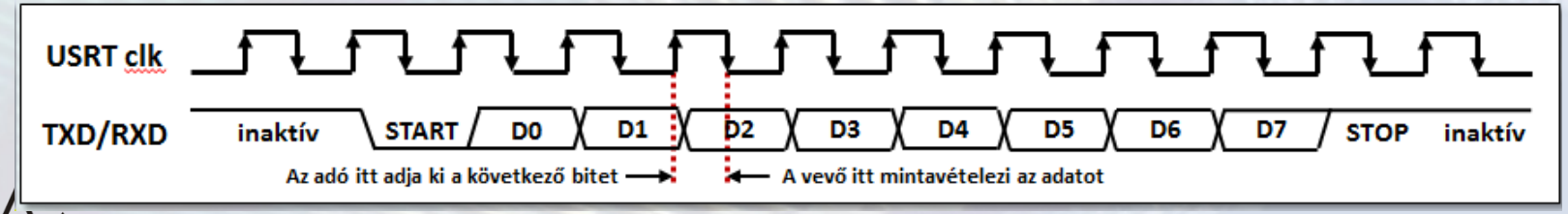


PIN	RS-232	RS-485 (4W)	RS-485 (2W)	RS-422
1	DCD	TxD-(A)	---	TxD-(A)
2	RXD	TxD+(B)	---	TxD+(B)
3	TXD	RxD+(B)	Data+(B)	RxD+(B)
4	DTR	RxD-(A)	Data-(A)	RxD-(A)
5	GND	GND	GND	GND
6	DSR	---	---	---
7	RTS	---	---	---
8	CTS	---	---	---
9	---	---	---	---

USRT Soros kommunikáció

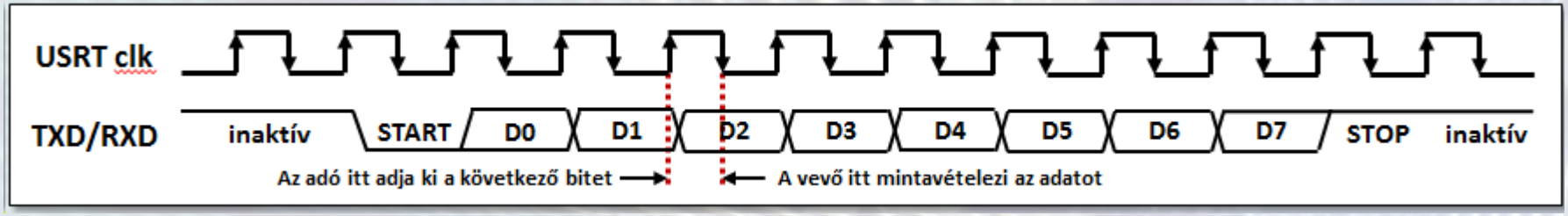
- **Univerzális szinkron adó vevő egység**
  - Szinkron esetben külön órajel adja a bitsebességet, ezért lehetne tetszőleges a sebesség → megszokás alapján tipikus a szabvány sebességek választása
    - +1 vezeték, viszont jelentősen egyszerűbb hardver felépítés
    - Az órajel folyamatos (akkor is van, ha nincs adatátvitel)





USRT Soros kommunikáció

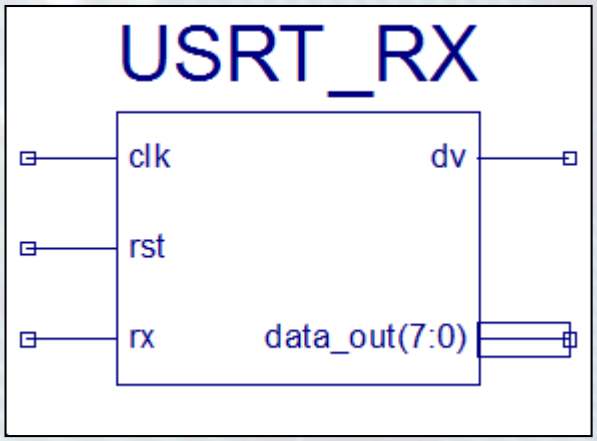
- **USRT paraméterek ugyanazok, mint UART esetén**
  - A MiniRISC rendszerben ezt használjuk a PC terminál felé (8 adat bit, nincs paritás bit, 1 stop bit: 8N1, sebeség: 9600b/s)
  - Átviteli jelek és vezérlésük:
    Adás oldalon felfutó élre új bit a a TX-en
    Vételi oldalon lefutó élre mintavétel az RX-en



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

USRT Soros kommunikáció

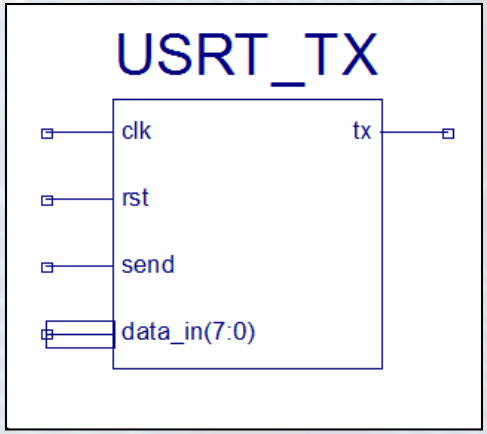
- **Egyszerű USRT vevő Verilog HDL kódja**
  - A 8 adatbittel együtt a START és STOP biteket is beléptetjük (ebben a példában és az adónál is egyszerűsítés miatt a bit sebesség a rendszer órajel frekvenciája)
  - Adat érvényes feltétel: DV = ~START & STOP
  - DV felhasználható az adatnak egy kimeneti regiszterbe írására és egy az adat érkezését jelző regiszter bit (RX_RDY) 1-be állítására




```
`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**
  - Az 1 órajel hosszú **send** indító jelre a 8 adatbittel együtt a START és STOP biteket is betöltjük (TX-en rögtön megjelenik a START-bit)
  - Ezután jobbra léptetve kiléptetjük a teljes karaktert, újra csupa „1” értékű bittel feltöltve, ezért az adás végén a STOP bit jelenik meg és TX 1 marad a következő **dv**-ig.



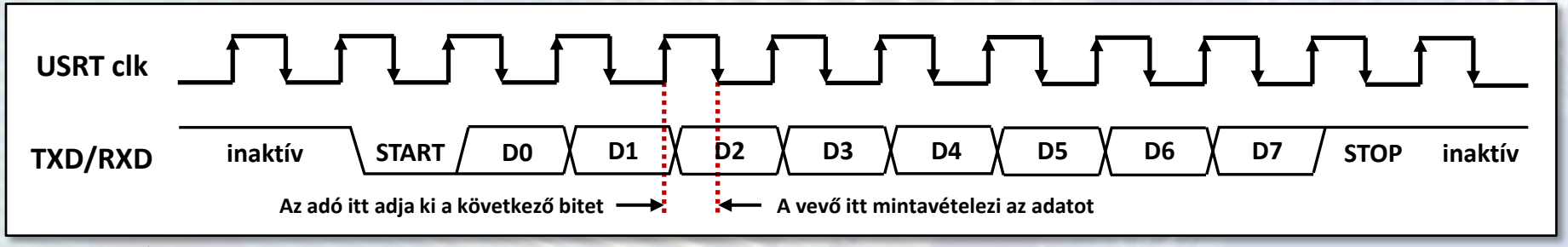

```
`timescale 1ns / 1ps
module USRT_TX(
    input clk,
    input rst,
    input send,
    input [7:0] data_in,
    output tx
    );

   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
```

MiniRISC USRT periféria

slave_usrt: **USRT soros kommunikációt biztosító periféria**

USRT adatátvitel

- Keretezett formátum:
  - 1 START bit, melynek értéke mindig 0
  - 8 adatbit (D0 – D7)
  - Nincs paritás bit
  - 1 STOP bit, melynek értéke 1 (és ha nincs adás TX utána is 1 marad)
- 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

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

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

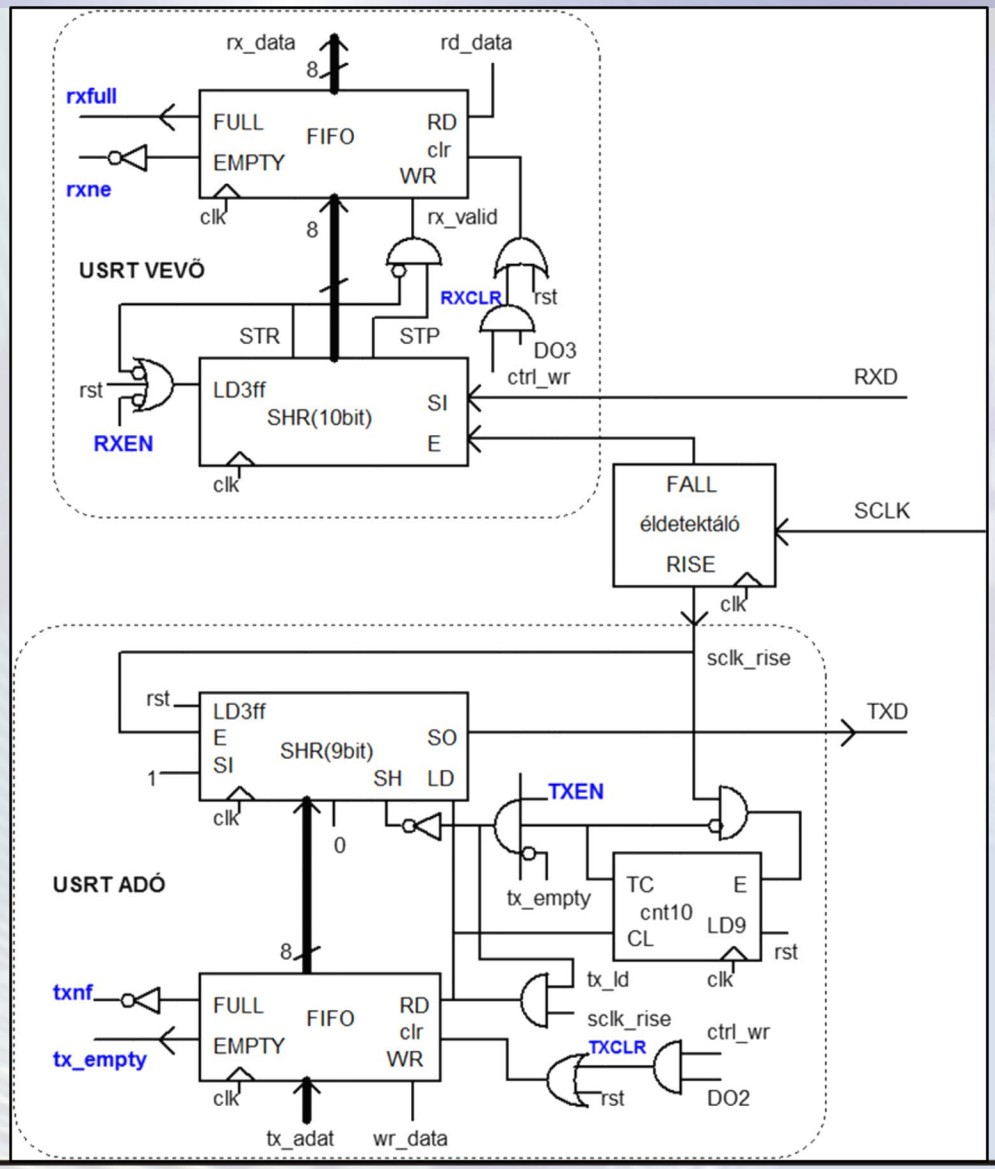
Bit	Jelentés	
TXEMPTY	0: az adási FIFO-ban van adat	1: az adási FIFO üres
TXNF	0: az adási FIFO tele van	1: az adási FIFO nincs tele
RXNE	0: a vételi FIFO üres	1: a vételi FIFO-ban van adat
RXFULL	0: a vételi FIFO nincs tele	1: a vételi FIFO tele van

MiniRISC USRT periféria

MiniRISC USRT blokkvázlata

USRT adó

USRT vevő



MiniRISC USRT periféria

A következőkben bemutatunk egy USRT minta programot, mely interruptosan kezeli a vételt és az adást.

A program az USRT szinkron soros adó/vevo perifériára kiküld egy stringet a BT0 menyomásakor. Az adás interruptot közvetlenül az adás elkezdése előtt engedélyezi (UIE TXNF bitje 1) és az a adás befejezésekor az adási IT program letiltja (UIE TXNF bitje 0).

A terminálról beérkezo adatot a vételi IT program kiírja a LED-ekre.

Működés:

A főprogram az inicializálás utáni végtelen ciklusban megnézi, hogy IT-s USRT adás történik-e éppen. (Ha az adási IT nincs letiltva, akkor adás van.)

Ha nincs adás, BT0 megnyomásakor a string címét átadja az IT rutinnak és engedélyezi az adás IT.

Az USRT IT a US státus register alapján eldönti, hogy az IT-t az adat érkezése (RXNE), vagy az adás regiszter nincs tele esemény (TXNF) okozta.

Beérkező adat esetén azt kiírja a LED-ekre.

Küldendő string esetén, beolvassa a következő karaktert s ha az nem 0x00, kiküldi az USRT adat kimeneti regiszterébe majd növeli a string címet és visszatér.

Ha az adat 0x00 (string vége), akkor letiltja az adás IT-t és visszatér.

MiniRISC USRT periféria

```
;USRT minta program IT-s vétel és adás

DEF BT0  0x01
;UC          7 6 5 4      3         2         1       0
;            0 0 0 0 RXCLR TXCLR RXEN TXEN
;UC_INI    0 0 0 0      1         1         1       1
DEF UC_INI  0x0f
;US/UI 0 0 0 0 RXFULL RXNE TXNF TXEMPTY
;UIE_RXNE 0000   0        1        0          0
;UIE TXNF   0000  0        0        1          0
DEF RXNE   0x04
DEF TXNF    0b00000010
DEF NTXNF 0b11111101 ; /TXNF
       CODE
start:
   jmp main          ;RESET belépési pont
   jmp irq           ;IT belépési pont
main:                ;main()
   mov r0, #UC_INI  ;{
   mov UC, r0        ; UC = UC_INI
   mov r0, #RXNE     ; UIE = RXNE
   mov UIE, r0       ; IT, ha vételi FIFO nem üres
   sti               ; sti()   globális IT eng.
```

```
main_loop:          ;   do{
   mov r0, UIE      ;
   tst r0, #TXNF    ;       if(!(UIE & TXNF))  // ha nincs USRT adás
   jnz  next        ;       {
   jsr BT0_STR      ;           if(BT01_STR()!=NULL)  // ha lenyomták a BT0-at
   cmp r11, #0      ;           {
   jz  next         ;
   mov r10, r11     ;               IT_str = str    //átadja az IT-nek a string címét
   mov r0, UIE      ;
   or r0, #TXNF     ;
   mov UIE, r0      ;               UIE= UIE | TXNF //engedélyezi az adás IT-t
                    ;           }
next:               ;       }
  ;  itt lehetne
  ;  bármit végezni
   jmp main_loop    ;   }
                    ;}
                    ; //a szubrutin BT0 lenyomását figyeli és visszaadja a string címét
BT0_STR:            ;BT01_STR(){   //ha nem volt lenyomás, akkor 0-át ad vissza
   mov r11, #0      ; // result = NULL
   mov r0, BTIF
   mov r1,BT        ;
   and r1,r0        ;  BTcpy = BTIF & BT     //BTcpy-ban
   mov BTIF, r1     ;  BTIF = BTcpy         //BTIF 1-es bitek törlése
   tst r1,#BT0      ;  if(!(BTcpy & BT0))
   jz bt_end        ;  {
   mov r11, #string0 ;      result = string0     ;  }                ;  }
bt_end:             ;  }
   rts              ;   return(result)
                    ;}
                DATA
                 org 1
string0:    DB "BT0 pressed ",0x00
```

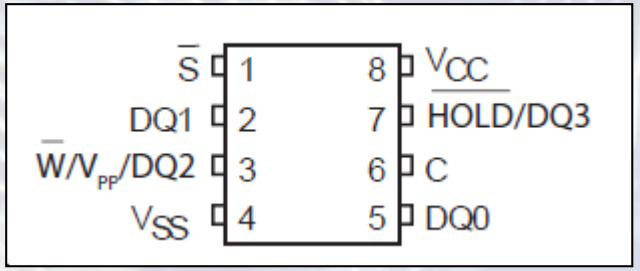
MiniRISC USRT periféria

```
 CODE
;USRT interrupt
;használt regiszterek:
;R12 általános célra
;R10 string következo karakterének címe
irq:                    ;ISR(){
   mov r12, US          ;  if(US & RXNE)      // ha kerekter jött
   tst r12,#RXNE        ;  {
   jz  tx_IT            ;     LD = UD        // beérkezett adat a LED-ekre
   mov r13,UD
   mov LD, r13          ;  }
tx_IT:
   tst r12, #TXNF       ;  if(US & TXNF)      // ha az adás pufferben még van hely
   jz  irq_end          ;  {
   mov r12,(r10)        ;     if(*str != 0x00)   // ha a sring következo karaktere nem 0x00 akkor kiküldi USRT-re
   cmp r12, #00         ;     {
   jz  end_of_string    ;        UD = *str
   mov UD, r12          ;        str++
   add r10, #1          ;     }
   jmp irq_end          ;     else
end_of_string:          ;     {
   mov r12, UIE
   and r12, #NTXNF      ;        UIE = UIE&NTXNF // ha nincs több adat, letiltja a további adás IT-t
   mov UIE, r12         ;     }
irq_end:             ;  }
   rti              ;}
```

Digitális technika
13. EA vége
Az alábbiak nem részei a vizsga anyagnak

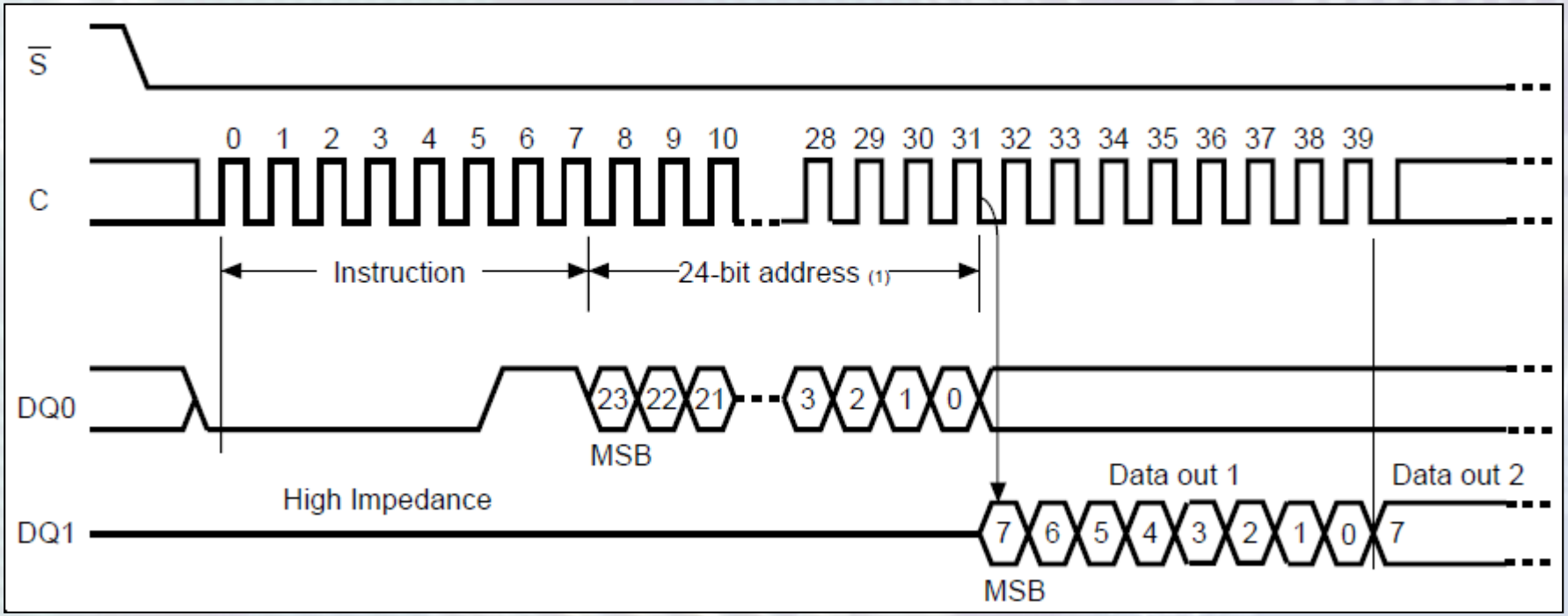
SPI soros protokoll

- **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**
  - Maximum 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 !



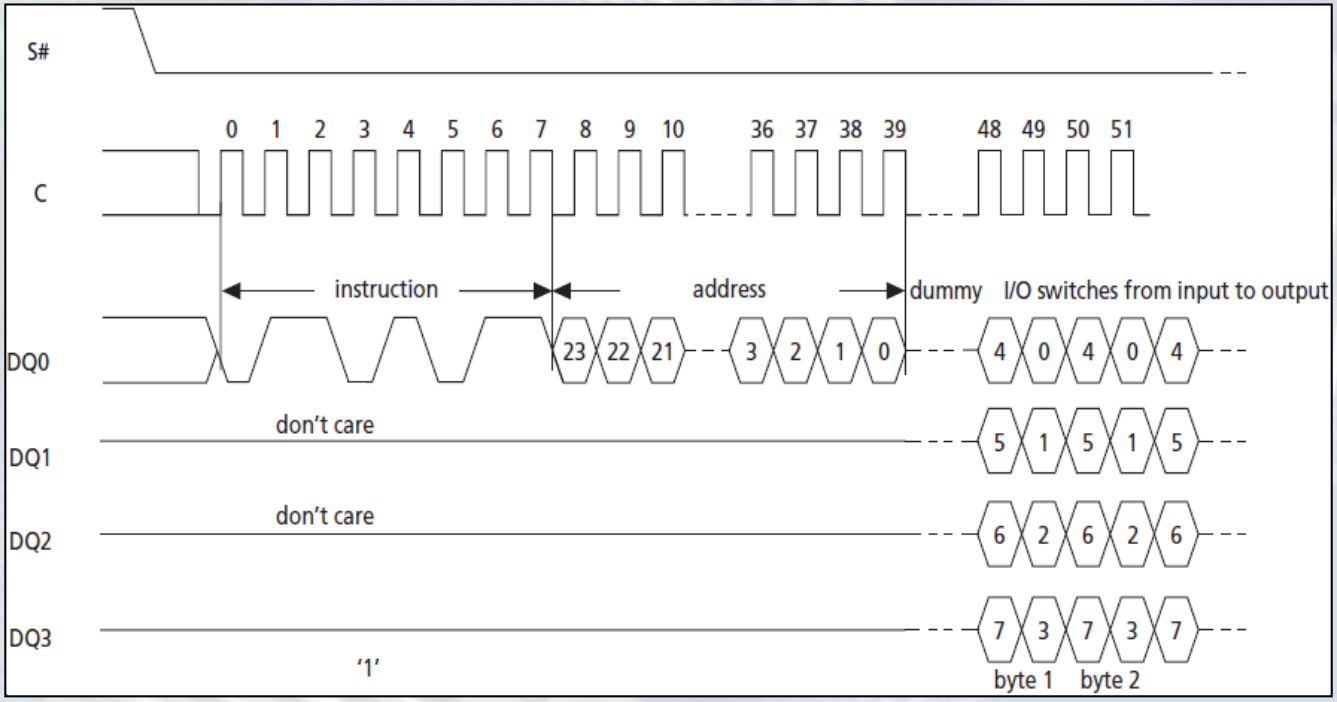
SPI soros protokoll

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



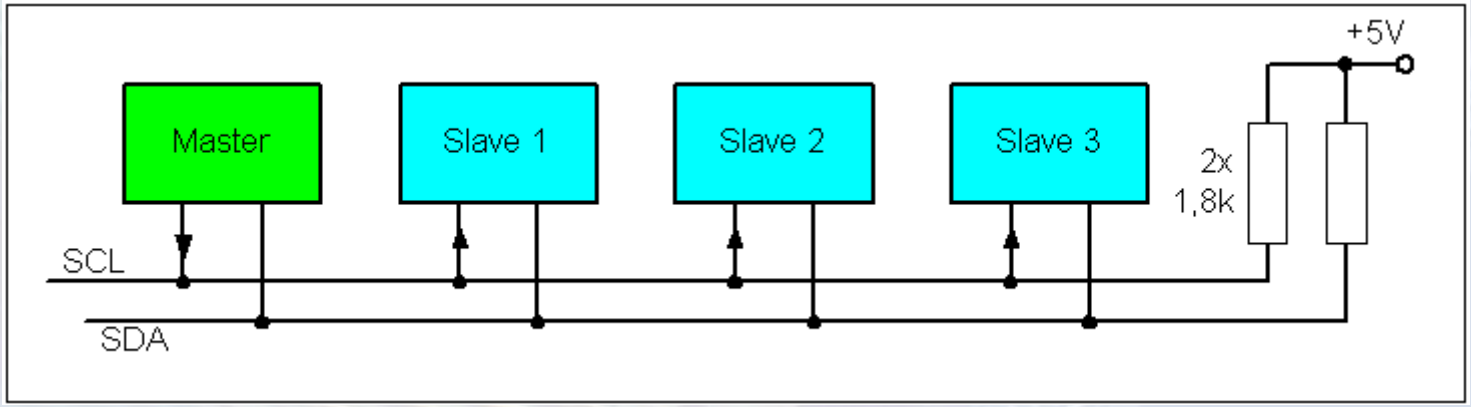
SPI soros protokoll

- **Kiterjesztett SPI protokoll soros memóriáknál**
  - Parancs, cím kivitel DQ0 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**
- **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, IPMB, 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**