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 mikrokontrollerek 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 kommunikációs technológiák

- A digitális technika tárgy keretében a legegyszerűbb, leggyakoribb ***soros adatátviteli interfészeket*** tekintjük át és csak ***vezetékes kapcsolatokkal*** foglalkozunk.
- A ***soros adatátviteli interfészek*** között az ***adatbitek átvitele*** egy vezetéken, ***időben egymás után*** történik.
- **Megkülönböztetünk szinkron és aszinkron kommunikációt:**
  - **Szinkron** (órajel jelvezeték is van): SPI, (I2C nem tanuljuk), USRT
    Egy bit az órajel periódusidejével egyező időnként (bitidő) változik.
  - **Aszinkron**: nincs külön órajel jelvezeték: UART
    Egy bit megegyezés szerinti időnként (bitidő) változik.

A kommunikációs technológiák

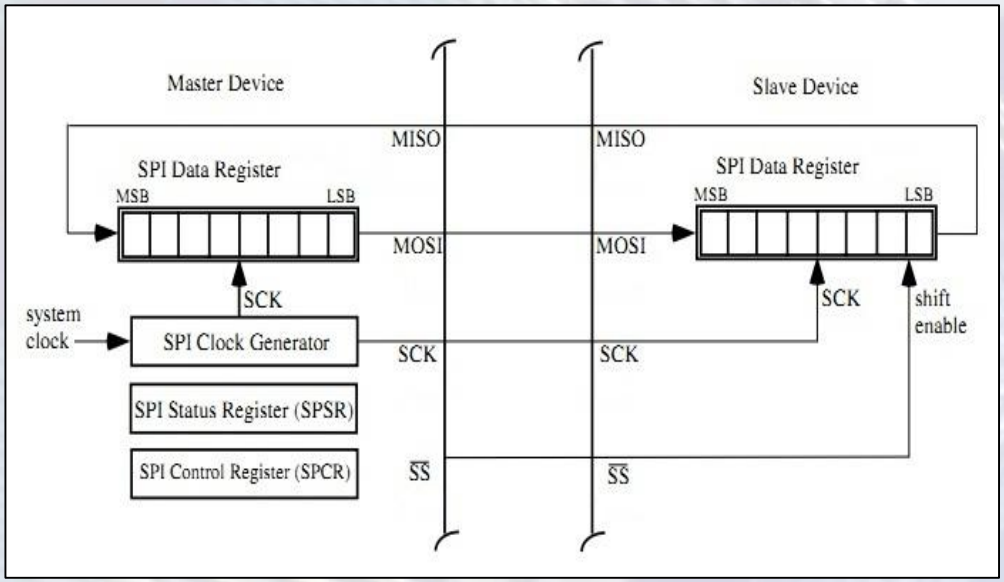
- **Megkülönböztetjük** a **belső** és **külső** kommunikációt
- **Belső kommunikáció:**
  - ***Készüléken***, nyomtatott áramkörön (NYÁK) ***belüli, kis távolságú adatátvitel*** (pl. mikrokontroller és szenzor között)
  - Általában azonosak a logikai szintek (a low és hi szintekhez tartozó feszültség) ezért többnyire nem kell szintillesztő áramkör.

A kommunikációs technológiák

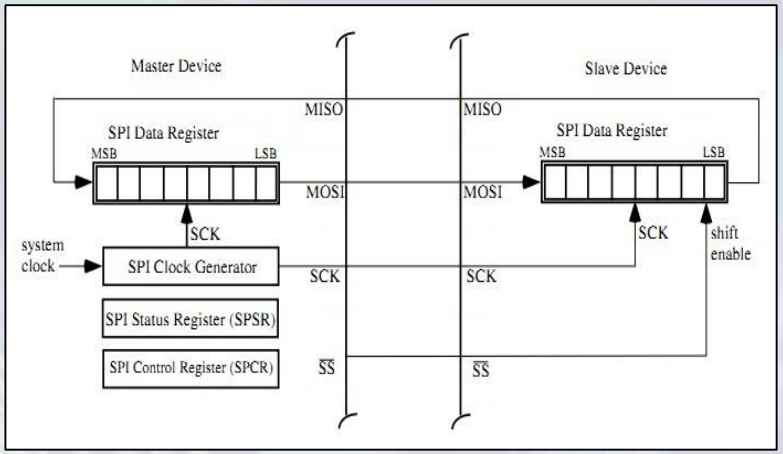
- **Külső kommunikáció**
  - **Speciális jelszintek** (speciális meghajtó és vevő áramkörök kellenek, nem részletezzük)
  - **Nagy távolság esetén** különösen fontos a **zavarvédelem**
  - A **megbízható működési távolság** a zavarvédelemtől is függ (pl. a speciális meghajtó és vevő áramkörök tulajdonságaitól), 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*, aktív low szintű), ***SCK*** (Serial clock), ***MOSI*** (Master Output Slave Input), ***MISO*** (Master Input Slave Output)
  - ***Egyszerre 2 irányú*** (*full-duplex*, adás és vétel) kommunikáció.



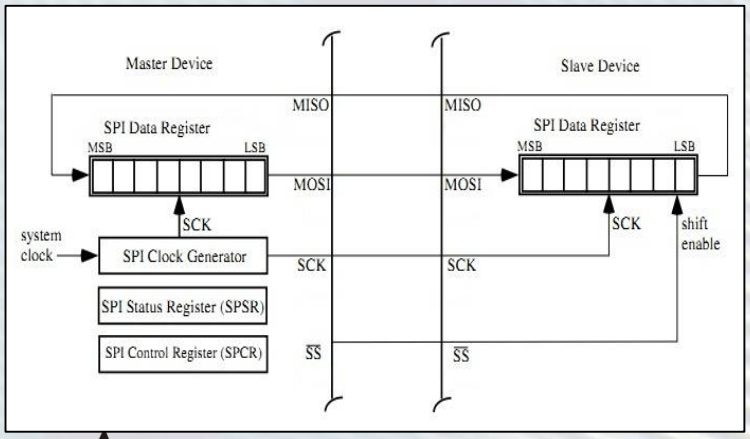
SPI soros protokoll

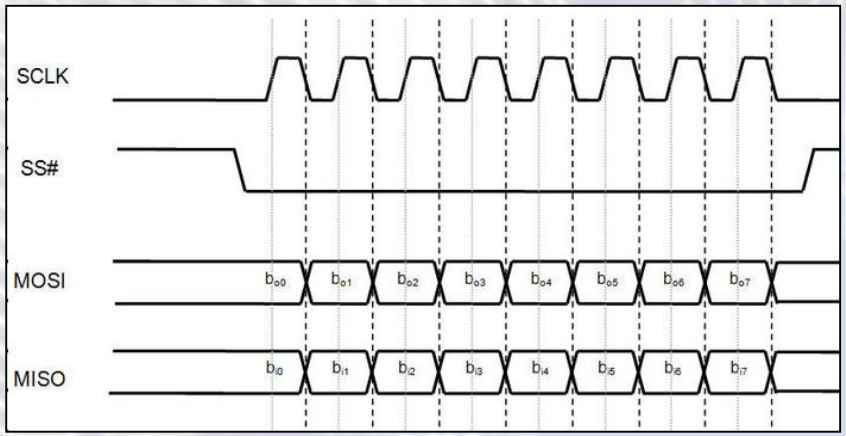


- Az ***/SS=0*** beállítása ***választja ki a slave-et***. A kiválasztott ***slave 3 állapotú MISO kimenete*** ennek hatására ***engedélyeződik***.
- ***A kommunikációt a master adatregiszterébe írás kezdeményezi.*** (Akkor szabad beleírni, ha a státus regiszterében jelzi, hogy van benne hely.)
- ***Egyszerű modell szerint magyarázva, a master shiftregisztere és a slave shiftregisztere*** sorba kapcsolódnak és együtt ***egy hurkot alkotnak***. *A* ***standard SPI-nál*** *a shitregiszterek* ***8 bitesek.***
- A ***master*** ezután ***8 órajel impulzust ad ki***, ennek hatására ***a master és a slave shiftregiszterében levő adat helyet cserél.***
- A ***master*** a státus regiszterében ***jelzi az adatcsere végét***. A jelzés után olvasható ki az adat.

SPI soros protokoll

- Az egyszerű modell szerint az SPI interfész „sorosan csatolt” shiftregiszterekből álló hurok, azonban a megvalósítás ennél kicsit bonyolultabb.
  - 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 órajel él melyre a kimeneti adat változik:** lelfutó, felfutó
      - **A bitek sorrendje:** MSB-vel vagy LSB-vel kezdődik

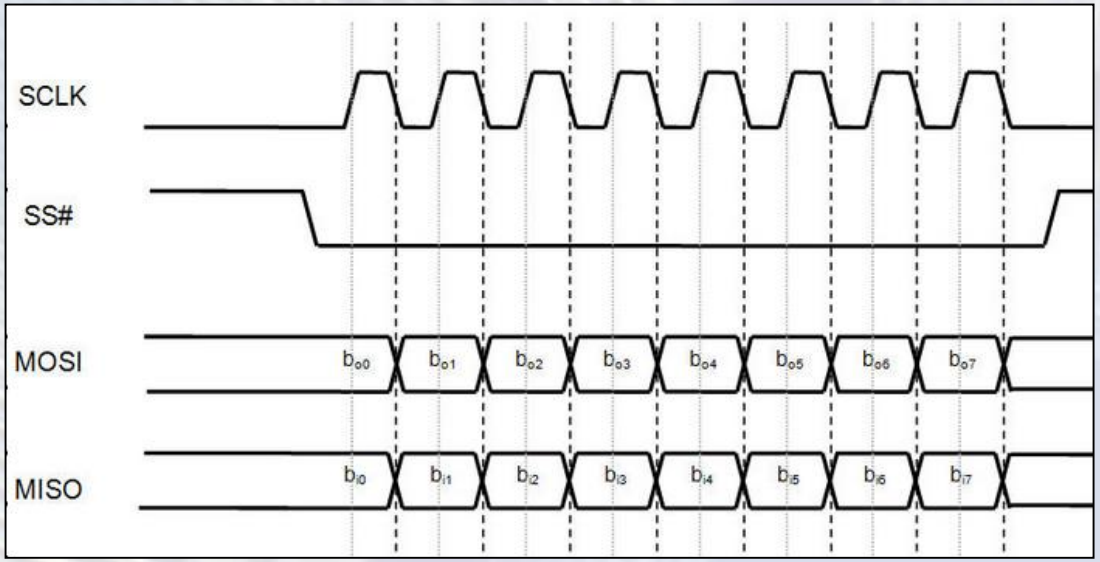




SPI soros protokoll

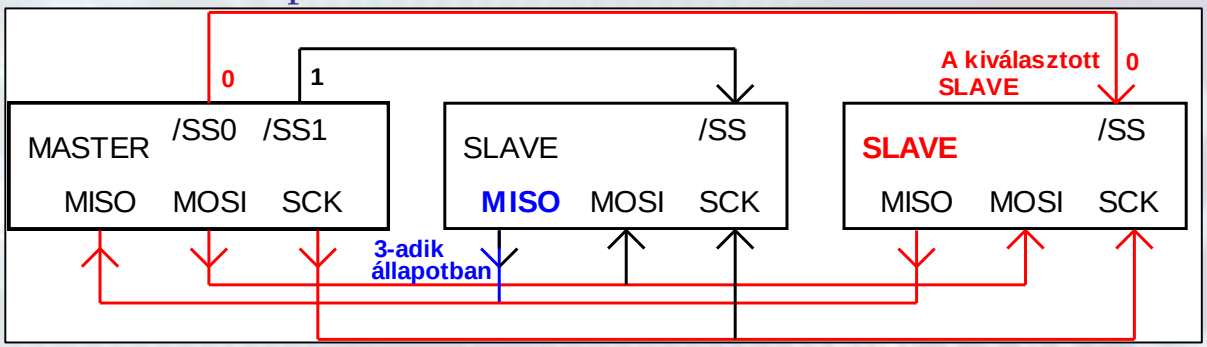
Az alábbi példában:

- Az **órajel kezdeti szintje low.**
- **Az adatátvitelt /SS = 0 beállításval kezi a master.**
- Az órajel **lelfutó élére változik az adat** és **felfutó élnél** íródik be (mintavételeződik) a shiftregiszterekbe.
- A bitek sorrendje LSB-vel kezdődik
- **Az átvitel után (8 órajel) a master** /SS=1-gyel **megszünteti a slave kiválasztását.**



SPI soros protokoll

- Az ***SPI*** interfész ***buszjellegű kommunikáció***, a ***master több slave-el*** képes kommunikálni**.** A tipikus rendszer ***1 MASTER – több SLAVE***



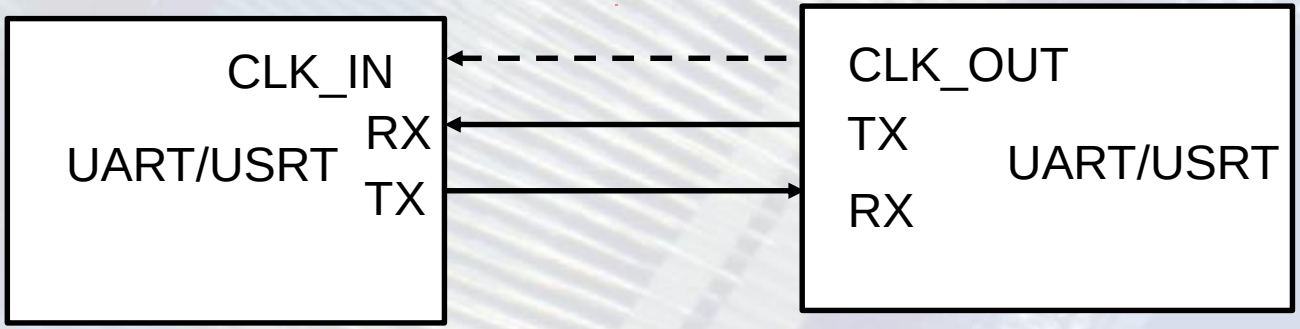
- ***/SSi 0-ba állításaval választja ki a master azt a slave-et***, amellyel kommunikálni akar (/SSi alacsony aktív).
- A ***kiválasztott slave MISO kimenete aktivizálódik, a többi slave MISO-ja 3-adik állapotban*** marad.
- ***Az SPI adatregiszterébe írásra indul a kommunikáció***.
- Ennek hatására ***8 (vagy több) órajel alatt a master és a kiválasztott 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***, 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***
  - ***Csak bemenet*** (MISO-MISO): Pl. A/D átalakító, szenzorok.
    A masterba írt adat indítja a kommunikációt, de a bele írt adat nem kerül sehova, csak az adatátvitelhez szükséges órajel generálását kezdeményezi. A beérkező adatot ki kell olvasni.
  - ***Csak kimenet*** (MOSI-MOSI): Pl. D/A átalakító, kijelzők.
    A master adatregiszterébe írjuk a kiküldendő adatot és az bekerül a slave-be. A MISO-n levő logikai érték ugyan bekerül a master adatregiszterébe, de az nem valós adat, ki kell olvasni, de „eldobjuk”.

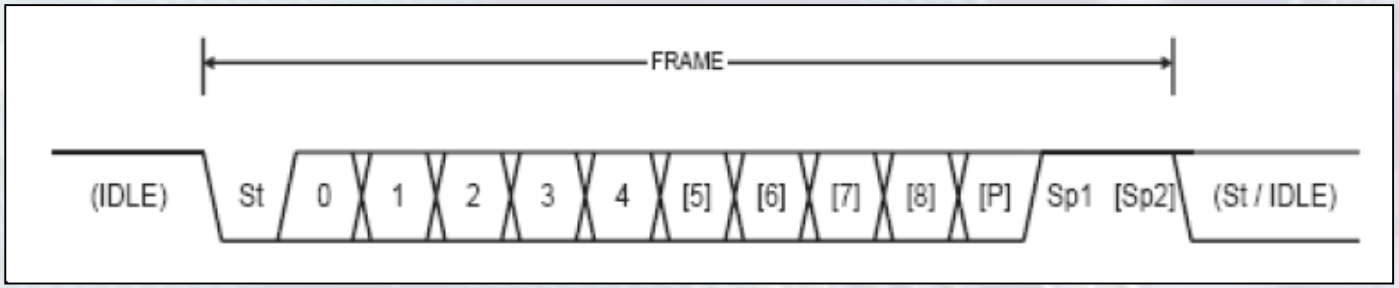
UART/USRT Soros kommunikáció

- **Univerzális aszinkron/szinkron adó vevő egység**
  - ***Kis sebességű adatátvitel***, bit rate < 1 Mb/s (Mega bit/sec).
  - Nincs megkülönböztetett szerep (master, slave), egyenrangú kommunikáció két egység között
  - ***Külön TX (adat adás) és RX (adat vétel) adatvezetékek***, egy időben akár két irányú (full duplex) kommunikáció.
    - Órajel csak USRT esetében van.



UART/USRT Soros kommunikáció

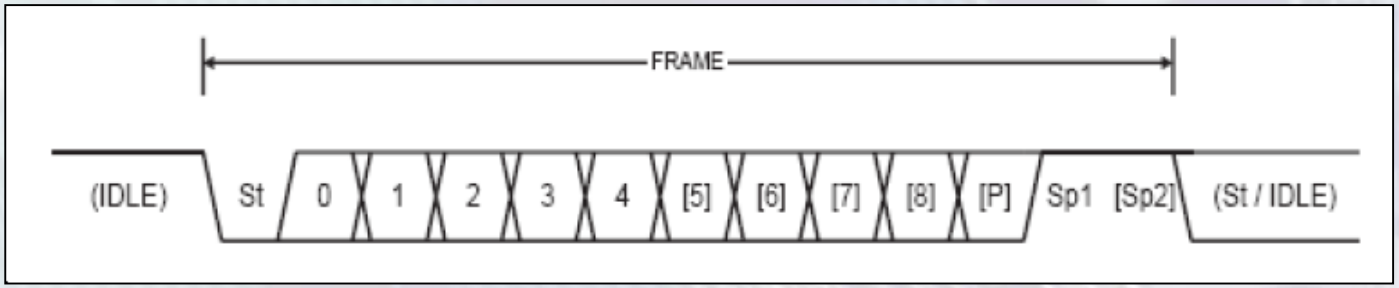
- **Univerzális aszinkron/szinkron adó vevő egység**
  - ***Aszinkron*** esetben ***órajel nélkül***, de ***előre egyeztetett, szabványos bitsebességgel*** (2400, 4800, ..115200 b/s)



    - 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 és az adat bit ennek a megfelelő élére mintavételeződik a vételi oldalon***. Kell +1 vezeték, viszont jelentősen egyszerűbb a hardver felépítése.

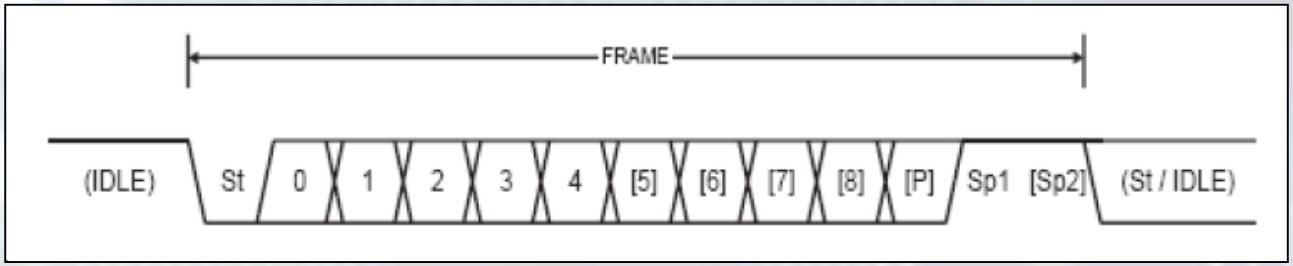
UART/USRT Soros kommunikáció

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



  - 1 **START** bit + [5-6-7-8] **ADAT** bitek+ [1 **Paritás** bit] + [1-1,5, 2] **STOP** bit
  - Az ***adat bitek száma*** beállítható [5-6-7-8], de tipikusan 8-ra állítjuk.
  - ***Adatbitek sorrendje***: LSB → MSB (D[0], D[1],..D[7])
  - A Paritás bit használata opcionális.

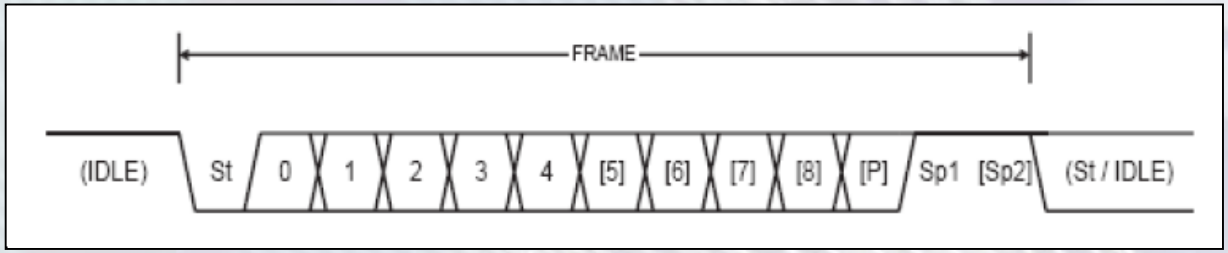
UART/USRT Soros kommunikáció



- A **Paritás** bit az adat bitek 1-eseinek a számát párosra vagy páratlan-ra egészíti ki. Ennek megfelelően beállítható ***párosra*** vagy ***páratlan-ra (vagy nem használtra)***. ***Hiba detektálására*** használható. A vevő ***jelezni tudja, ha az adatbitekben 1 vagy páratlan számú bit ellentétesre változik*** (zaj).
- A **STOP** bit az utolsó adat bit után a beállítottnak megfelelő bitideig 1 értékű. A FARME (keret) végét, vagyis az adat megérkezését jelzi.
- Tipikus beállítás, amit a leggyakrabban használunk:
  1 START bit + **8 ADAT** bit + **1 STOP** bit vagyis
  8 adat bit, *nincs PARITÁS* bit, 1 STOP bit. Ezt **8N1**-el szokás jelölni.
- Ekkor 8 adatbit 10 bitidő alatt vihető át.

UART Soros kommunikáció

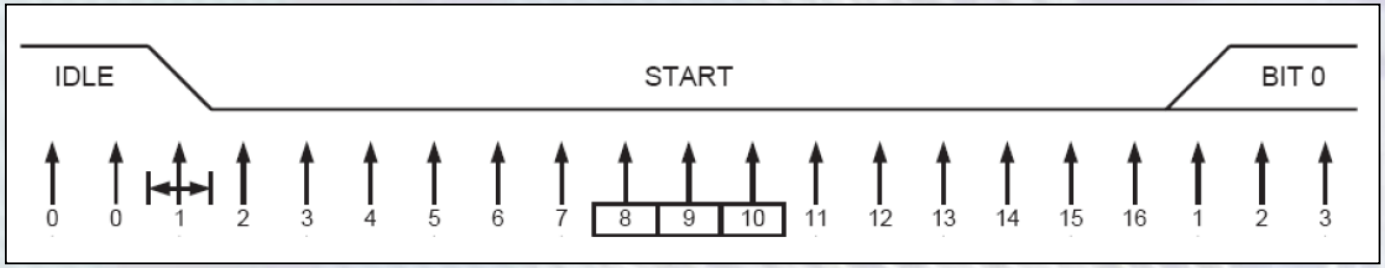
- **<u>Aszinkron</u> adó: Saját (pontos) órajelének ütemezése alapján kiadja a biteket**
  - Tetszőleges időben elkezdhet adni a TX vonalán.
  - A STOP bit vége után azonnal kezdheti a következő bájt START bitjét.



  - ***Nincs órajel vezeték***. A bitidőt jelző órajel (tipikusan) 16x-osát az adó oldal és vevő oldal magának állítja elő. A vevő figyeli az RX vonalát. Ha a vonalon lefutó élet ↓ érzékel, az egy aktív átvitel elindítását jelentheti (vagy egy tüske szerű zavart!)

UART/USRT Soros kommunikáció

- **<u>Aszinkron</u> vevő: Saját (pontos) órajelének ütemezése alapján a bitfrekvencia 16-szorosával mintavételezi a vonalat → Várja START lefutó ↓ élét**
  - Ezután a fél bitidő múlva, kb. a START bit közepén végrehajt egy (vagy 3) mintavételt.



  - Három mintavétel lehetővé teszi a tüske szerű zavarok kiszűrését.
  - Ekkor, ha 3-ból legalább 2 minta 0, akkor elindul az adatbitek mintavételezése is (hasonló módon). Egyébként újra a START bitre vár.

UART/USRT Soros kommunikáció

- A 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 a shiftregiszterből áttölthető egy adat tárolóba és a vevő azonnal újra vételi módra állhat.
- Az adat tároló egyszerűbb esetben egy adat regiszter komplexebb UART esetén néhány byte tárolására képes FIFO.

UART/USRT Soros kommunikáció

- **Felprogramozásnál be kell állítani:**
  - A bitsebességet (Baud rate),
  - Az adatbitek számát (3-8, tipikusan 8),
  - A paritást: *nincs* paritás bit, *páros* paritás, *páratlan* paritás (kis távolságú átvitelnél nem használunk paritás bitet
  - A stop bitek számát (1, 1.5, 2, tipikusan 1),

UART/USRT Soros kommunikáció

- Az UART státus regisztere jelzi, ha ***az adás regiszter vagy FIFO üres. Jelzi, ha az adási FIFO-ban van hely***. Újabb adat csak akkor küldhető, ha van hely vagy üres jelzést ad.
- Jelzi, ha a ***vételi adat regiszterben*** vagy ***vételi FIFO-ban van adat***. Ekkor lehet kiolvasni az adatot.
- Jelzi, ha vételnél a ***STOP bit helyén 0-át érzékelt***. Ez a ***FRAMING ERROR***.
- Paritás bit használata esetén ***jelzi a paritás hibát***.
- Hiba jelzés esetén az adat nem érvényes.

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

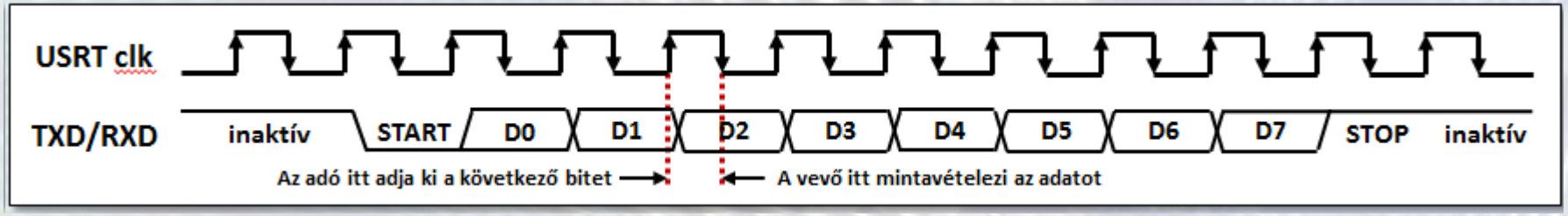
- **Pl. 9600 b/s sebességnél 1 karakter átviteli ideje (1ms) alatt a MiniRISC kb. 5300 utasítást hajt végre**
- **UART/USRT Periféria kezelés?**
  - Lekérdezéssel nem célszerű, mert sok időveszteség
  - Megszakítással, ez esetleg túl gyakori, de segít a FIFO
  - Speciális kiegészítés a soros adó/vevőkhöz: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

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



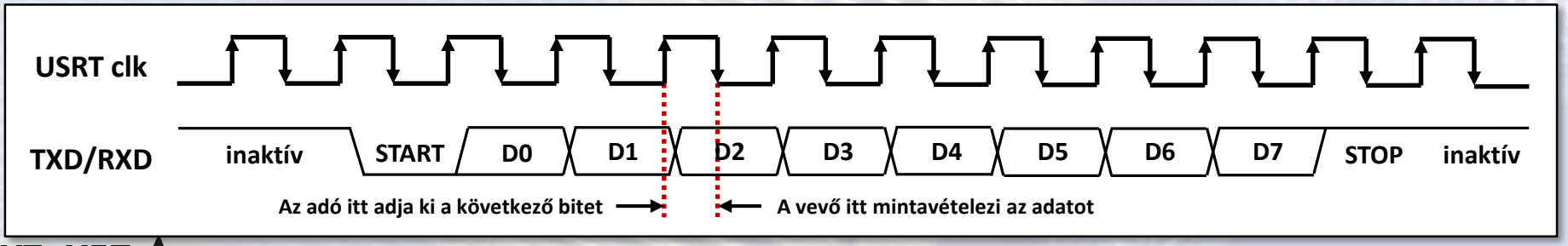
  - Az órajelet bármelyik egység adhatja, valójában csak szinkronizációs célokat szolgál.
  - Az USRT adó vevő hardver realizációja egy-egy tölthető, engedélyezhető shiftregiszter + egyéb logika.

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
- 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

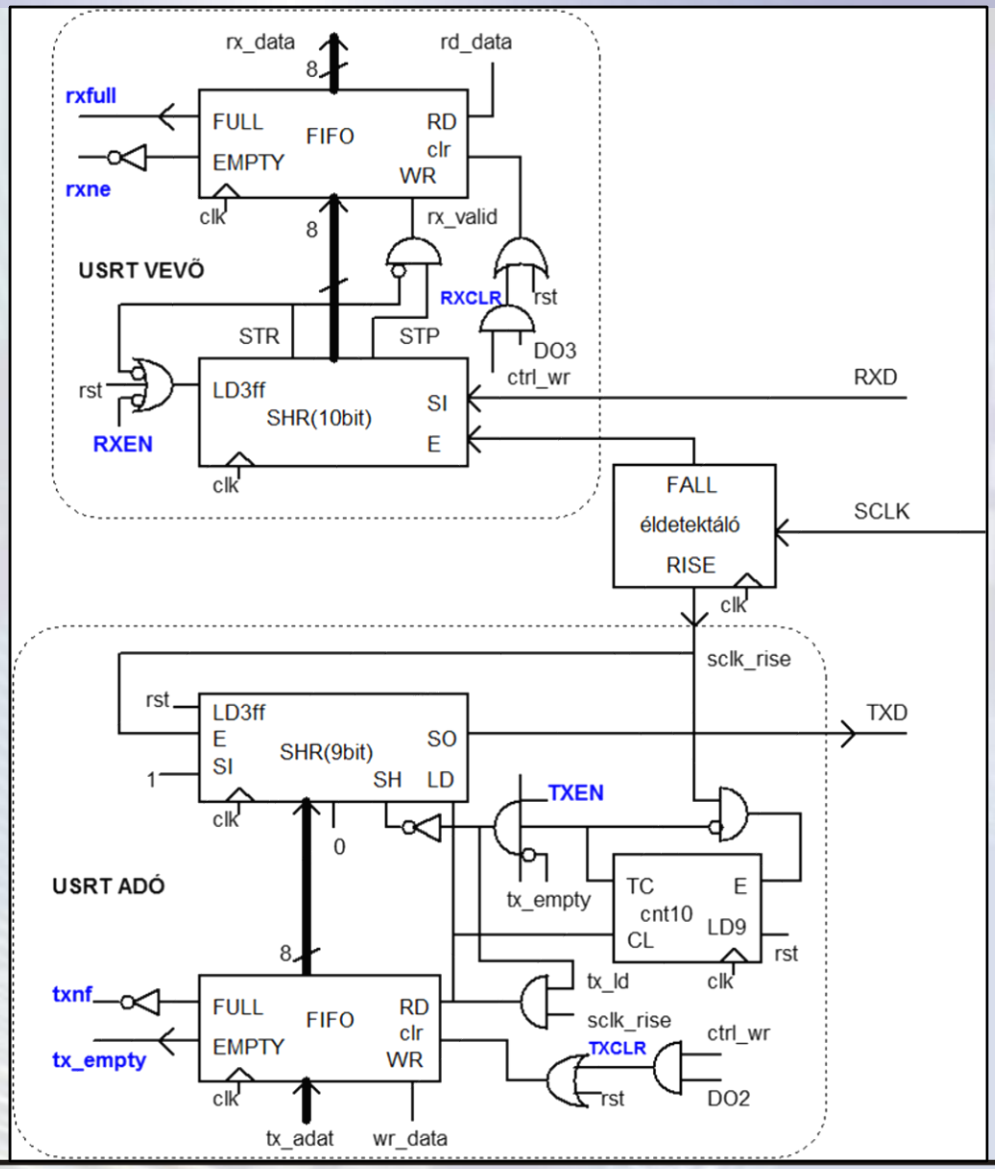
Bit	Jelentés	
TXEMPTY	0: az adási FIFO-ban van adat	1: az adási FIFO üres
TXNF	0: az adási FIFO tele van	1: az adási FIFO nincs tele (lehet bele írni)
RXNE	0: a vételi FIFO üres	1: a vételi FIFO-ban van adat (ki lehet olvasni)
RXFULL	0: a vételi FIFO nincs tele	1: a vételi FIFO tele van

MiniRISC USRT periféria

MiniRISC USRT blokkvázlata

USRT adó

USRT vevő



MiniRISC USRT periféria

A következőkben bemutatunk két USRT minta programot IT-s kezeléssel. Az könnyebb érthetőség miatt külön programban mutatjuk be az IT-s vételt és az IT- adást.

USRT IT-s vétel

Ez a program a terminálról beérkező adatot a vételi IT programban kiírja a LED-ekre.

Működés

Inicilizáláskor törli a FIFO-kat, engedélyezi az USRT vevőjét és az RXNE (vételi FIFO nem üres) interruptot és engedélyezi a globális IT elfogadást.

A főprogram az inicializálás utáni végtelen ciklusban semmit nem végez.

Az USRT IT a US státus regiszter alapján ellenőrzi, hogy az IT-t az adat érkezése (RXNE, vételi FIFO nem üres) okozta.

Beérkező adat esetén azt kiírja a LED-ekre (egyébként nem) és rti-vel visszatér.

MiniRISC USRT periféria

```
;USRT minta program IT-s vétel
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       0
DEF UC_INI  0x0e
;US/UI 0 0 0 0 RXFULL RXNE TXNF TXEMPTY
;UIE_RXNE 0000  0       1       0       0
DEF RXNE   0x04
        CODE
start:
   jmp main          ;RESET belépési pont
   jmp irq           ;IT belépési pont

main:                ;main()
   mov r0, #UC_INI    ;{
   mov UC, r0         ;  UC = UC_INI // inicializálás
   mov r0, #RXNE      ;  UIE = RXNE  // vételi IT
   mov UIE, r0        ;              // engedélyezés
   sti                ;  sti()       // globális IT engedélyezés
main_loop:            ;  do{
                      ;              //itt lehetne bármit végezni
   jmp main_loop      ;  }while(1)
                      ;}
```

```
;USRT interrupt
;használt regiszterek:
;R12 általános célra
irq:                  ;ISR(){
   mov r12, US        ;  if(US & RXNE)  // ha kerekter jött
   tst r12,#RXNE      ;  {
   jz  irq_end        ;     LD = UD     // beérkezett adat a LED-ekre
   mov r12,UD
   mov LD, r12        ;  }
irq_end:              ;
   rti                ;}
```

MiniRISC USRT periféria

USRT IT-s adás

A program az USRT perifériába kiküld két stringet a BT0 menyomásakor.

A stringek küldését a PRINT_STR szubrutin kezdeményezi.

Az interrupt rutin karakterenként küldi ki a stringet. Minden IT-kéréskor egy karaktert ír ki és visszatér. Ha elért a string végére, az adási IT-t letiltja (UIE TXNF bitje 0) és utána tér vissza.

Működés

A főprogram az inicializáláskor törli a FIFO-kat, engedélyezi az USRT adót, de ***az adási IT-t letiltva hagyja***.

Az inicializálás utáni végtelen ciklusban megnézi, hogy megynomták-e a BT0 nyomógombot. Ha nem tovább fut a végtelen ciklus (nem vár benne, tehát itt bármit lehetne végezni).

Ha a nyomógombot megnyomták, meghívja kétszer a PRINT_STR szubrutint, átadva neki az egyik majd másik string címét.

A PRINT_STR szubrutin megvárja, hogy az előző string kiíratása befejeződjön, majd átadja a kiírandó string címét az interrupt rutinnak és az adás interruptot engedélyezi (UIE TXNF bitje 1).

Az USRT IT a US státus register alapján ellenőrzi, hogy az IT-t adási FIFO nincs tele esemény (TXNF) okozta. (Ha nem, visszatér.) Ha igen, beolvassa a string következő karakterét 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 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       0       1
DEF UC_INI  0x0d
;US/UI 0 0 0 0 RXFULL RXNE TXNF TXEMPTY
;UIE TXNF   0000  0       0       1       0
DEF TXNF    0b00000010
DEF NTXNF 0b11111101 ; /TXNF
      CODE
main:                ;main()
   mov r0, #UC_INI    ;{
   mov UC, r0         ;  UC = UC_INI //
   sti                ;  sti()      // globális IT eng.
main_loop:            ;  do{
   mov r0, BTIF       ;
   mov BTIF, r0       ;     BTIFcpy = BTIF
                      ;     BTIF = BTIFcpy // törlés
   mov r1, BT         ;
   and r0, r1         ;
   tst r0, #BT0       ;
   jz next            ;   if(BTIFcpy & BT & BT0) //
                      ;     Ha BT0 lenyomás volt
   mov r1, #str0      ;   {
   jsr PRINT_STR;     PRINT_STR(&str0[0])
   mov r1, #str1
   jsr PRINT_STR;     PRINT_STR(&str1[0])
next:                 ;   //itt lehetne bármi
   jmp main_loop  ;   }while(1)
                      ;}
PRINT_STR:  ;PRINT_STR(char* string){
   mov r0, UIE    ;   while(UIE & TXNF)
   tst r0, #TXNF ; // vár a string küldésre
   jnz PRINT_STR
   mov r10, r1     ; IT_str = string //str cím
   mov r0, UIE    ;   UIE = UIE | TXNF
   or r0, #TXNF
   mov UIE, r0    ;// adás IT engedélyezés
   rts            ;}

   DATA
str0:
   DB "String ONE\r\n",0x00
str1:
   DB "Strint TWO\r\n",0x00
CODE
;USRT interrupt
;használt regiszterek:
;R12 általános célra
;R10 string következo karakterének címe
irq:                  ;ISR(){
   mov r12, US        ;
   tst r12, #TXNF     ;  if(US & TXNF)
   jz  irq_end        ;  {
   mov r12,(r10)      ;     if(*str != 0x00)
   cmp r12, #00       ;     {
   jz  end_of_string ;
   mov UD, r12        ;        UD = *str
   add r10, #1        ;        str++    }
   jmp irq_end        ;     else
end_of_string:        ;     {
   mov r12, UIE
   and r12, #NTXNF;   UIE = UIE &NTXNF
   mov UIE, r12       ;     } //adás IT tilt.
irq_end:              ;  }
   rti                ;}
```

Digitális technika
13. EA vége