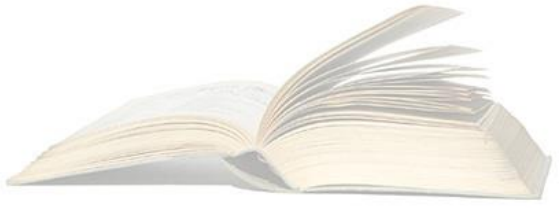
Óbudai Egyetem
Neumann János Informatikai Kar

UNIVERSITAS BUDENSIS

Beágyazott és érzékelőalapú rendszerek

Lovas István
lovas.istvan@nik.uni-obuda.hu



UART – soros kommunikáció

- A legtöbb mikrovezérlőnél alap perifériaként elérhető az U(S)ART, mely lehetővé teszi a soros kommunikációt
- U(S)ART:
  - **U**niversal (**S**ynchronous) **A**synchronous **R**eceiver **T**ransmitter (Univerzális Szinkron és Aszinkron Küldő és Fogadó)
- Felhasználható:
  - Hardverek közötti kommunikáció
    - mikrovezérlő ⇔ mikrovezérlő
    - Mikrovezérlő ⇔ PC stb.
  - Hibakeresésre stb.

UART – soros kommunikáció

- Az AVR mikrovezérlők esetén külön céláramkör látja el ezt a feladatot (szoftveresen is megvalósítható két digitális IO kivezetéssel, korlátozott átviteli sebesség mellett)
- Egy kommunikáció kiépítéséhez összesen UART esetén 3db (RxD/TxD/Gnd), USART esetén 4db (RxD/TxD/Gnd/Clock) vezetékre van szükség.
- RxD: Receive
- TxD: Transmit
- Az Atmega2560 (Arduino mega) összesen 4db USART-al rendelkezik:
  - TX/RX0 (0/1 pin)
  - TX1/RX1 (18/19 pin)
  - TX2/RX2 (16/17 pin)
  - TX3/RX3 (14/15 pin)

UART - Soros aszinkron kommunikáció működése

- UART kommunikáció maximum két eszköz között valósítható meg
- Az UART esetén nincs közös órajel (clock) vezeték, azaz nincs a két eszköz (adás/vétel) szinkronizálva.
- A szinkronizáció hiánya miatt, a küldő eszköznek ugyanakkora frekvencián kell küldenie az adatot, mint amilyen frekvencián a vevő mintavételezi azt, és ez fordítva is igaz.

UART - Soros aszinkron kommunikáció működése

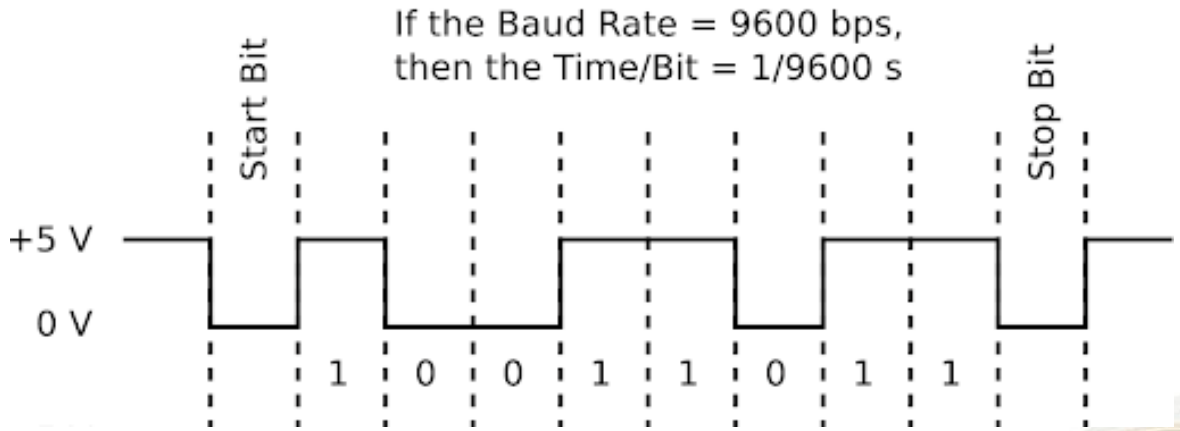
- 5 rétegre bontható a kommunikáció:
  - Fizikai:
    - csatlakozók fizikai paramétere, kialakítása
  - Elektromos:
    - feszültségszintek meghatározása
  - Logikai:
    - Logikai 1 és Logikai 0 –nak megfelelő feszültségszintek meghatározása
  - Adat:
    - Kommunikáció sebességének (baud rate) meghatározása
    - Küldendő adatcsomag mérete (8, 9, 10 bitből álló)
    - Speciális jelzőbitek meghatározása (1/2 stop bit, paritás bit)
  - Alkalmazási:
    - Adatcsomagok küldési sorrendjének meghatározása
    - Adatcsomagok visszaalakításának módja üzenetté

UART - Soros aszinkron kommunikáció működése

- AVR mikrovezérlő esetén az UART-nál 3db vezeték kell:
  - RxD, TxD, GnD
- Elektromos/Logikai réteg:
  - TTL jelszint
  - 0V = Logikai 0
  - 3.3V vagy 5V = Logikai 1
- Adatréteg (pl.):
  - 9600 bps
  - 1 START bit, 8DATA bit, 1 STOP bit
- Alkalmazási réteg:
  - Küldés/fogadás 10bites csomagokban
  - Csomag tartalma az adatrétegben definiált jelzőbitek, és byte méretű adat

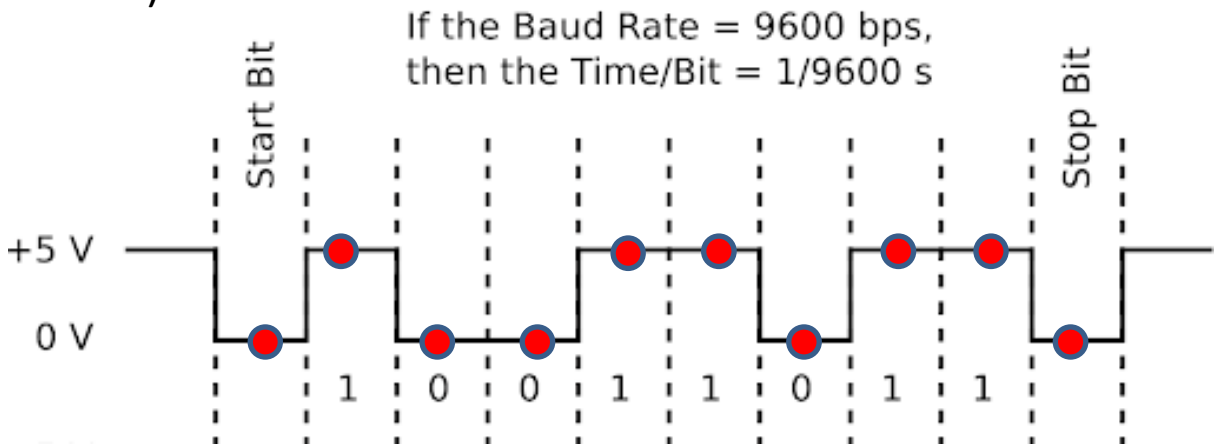
UART - Soros aszinkron kommunikáció működése

- Küldés:
  - Adásszünetben a TxD állapota Logikai 1 értéket vesz fel
  - Küldés esetén 1 Stop bit kiküldésre kerül (9600 bps esetén 104uS ideig logikai 0 szintet vesz fel a TxD vonal)
  - Ezután kiküldésre kerülnek az adatbitek (egymás után sorba)
  - Az adatbit után ha be van állítva, akkor egy paritásbit áll
  - Az átvitel végét pedig 1 vagy 2db STOP bit jelzi (Logikai 1 érték)
  - AVR esetén a STOP bit mindig Logikai 1 értéket vesz fel! (az ábrán logikai 0 van)



UART - Soros aszinkron kommunikáció működése

- Fogadás:
  - Várakozás, amíg az RxD vonal állapota meg nem változik (Logikai 0 értéket vesz fel)
  - 1/9600/2=52us (9600 bps esetén) várakozás után mintavételezi és ellenőrzi, hogy valóban START bit érkezett
  - Ezután 104us időközönként mintavételezi az RxD vonalat, amíg meg nem érkezik a teljes adatcsomag
  - AVR esetén a STOP bit mindig Logikai 1 értéket vesz fel! (az ábrán logikai 0 van)



Rendszerórajel meghatározása

- Ideális esetben a rendszer órajel a soros kommunikáció egész számú többszöröse!
- Ellenkező esetben fáziskülönbség lép fel, ami adatvesztéssel járhat, mivel a két kommunikáló eszköz nem lesz szinkronban egymással!!!
- Minden adatlap tartalmazza az egyes órajelekhez tartozó általánosan használt baud rate értékeket, illetve a hozzá tartozó hibaszázalékokat is.
- Az ajánlott maximális hibaszázalék ±2-3%

Rendszerórajel meghatározása

- Datasheet: 22.11 Examples of Baud Rate Setting

Table 22-11. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4K	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2K	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8K	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4K	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6K	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8K	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2K	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4K	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250K	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max.(1)	0.5Mbps		1Mbps		691.2Kbps		1.3824Mbps		921.6Kbps		1.8432Mbps	

Rendszerórajel meghatározása

- Az AVR mikrovezérlő használhat belső illetve külső RC oszcillátort, ami az óra jelet állítja elő.
- Mivel a mikrovezérlő az USART órajelét a rendszer órajeléből állítja elő, ezért a belső oszcillátor helyett külső oszcillátort szokás használni, mert a belső oszcillátor nem túl stabil!!!
- A megfelelő adatküldéshez és fogadó oldali mintavételezéshez elengedhetetlen a pontos időzítés!
- Ehhez be kell állítani az AVR fuse bitjeit! (programozó használata esetén)

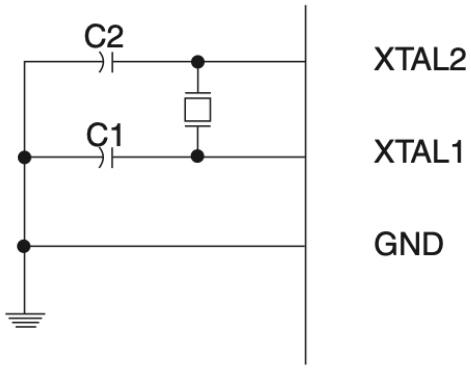
Rendszerórajel meghatározása

Table 10-1. Device Clocking Options Select[1]

Device Clocking Option	CKSEL3:0
Low Power Crystal Oscillator	1111 - 1000
Full Swing Crystal Oscillator	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

Note: 1. For all fuses “1” means unprogrammed while “0” means programmed.

Figure 10-2. Crystal Oscillator Connections



Rendszerórajel meghatározása

Table 10-3. Low Power Crystal Oscillator Operating Modes[3]

Frequency Range [MHz]	CKSEL3:1[1]	Recommended Range for Capacitors C1 and C2 [pF]
0.4 - 0.9	100[2]	–
0.9 - 3.0	101	12 - 22
3.0 - 8.0	110	12 - 22
8.0 - 16.0[4]	111	12 - 22

Table 10-4. Start-up Times for the Low Power Crystal Oscillator Clock Selection

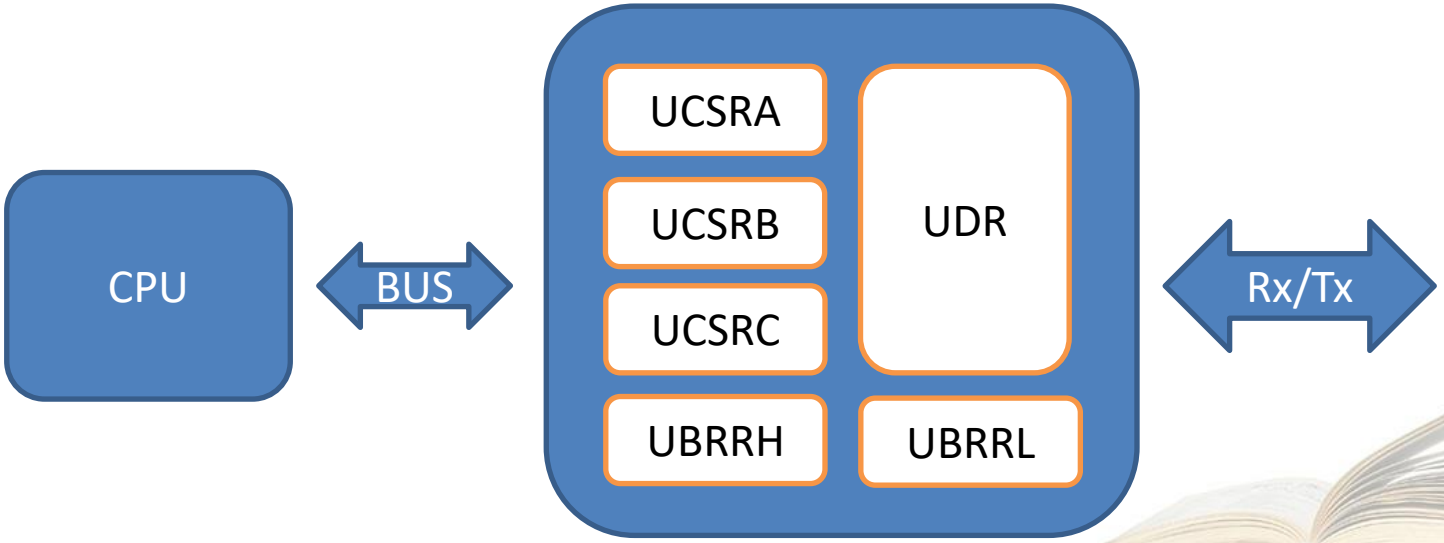
Oscillator Source / Power Conditions	Start-up Time from Power-down and Power-save	Additional Delay from Reset (V_{CC} = 5.0V)	CKSEL0	SUT1:0
Ceramic resonator, fast rising power	258CK	14CK + 4.1ms[1]	0	00
Ceramic resonator, slowly rising power	258CK	14CK + 65ms[1]	0	01
Ceramic resonator, BOD enabled	1KCK	14CK[2]	0	10
Ceramic resonator, fast rising power	1KCK	14CK + 4.1ms[2]	0	11
Ceramic resonator, slowly rising power	1KCK	14CK + 65ms[2]	1	00
Crystal Oscillator, BOD enabled	16KCK	14CK	1	01
Crystal Oscillator, fast rising power	16KCK	14CK + 4.1ms	1	10
Crystal Oscillator, slowly rising power	16KCK	14CK + 65ms	1	11

UART regiszterek

- A CPU-tól ”Független” perifériaként érhető el. Az UART beállítása után, a működtetése nagyon egyszerű, a felhasználónak csak el kell helyeznie az adatküldő regiszterbe a küldendő byte-ot, és a többit már a periféria magától elvégzi.
- Az adat fogadása szintén automatikusan történik. Detektálja az RX vonalon a beérkező adatot, beolvassa a teljes adatcsomagot, majd a mikrovezérlőnek jelzi, hogy olvassa be a megfelelő regiszterből.

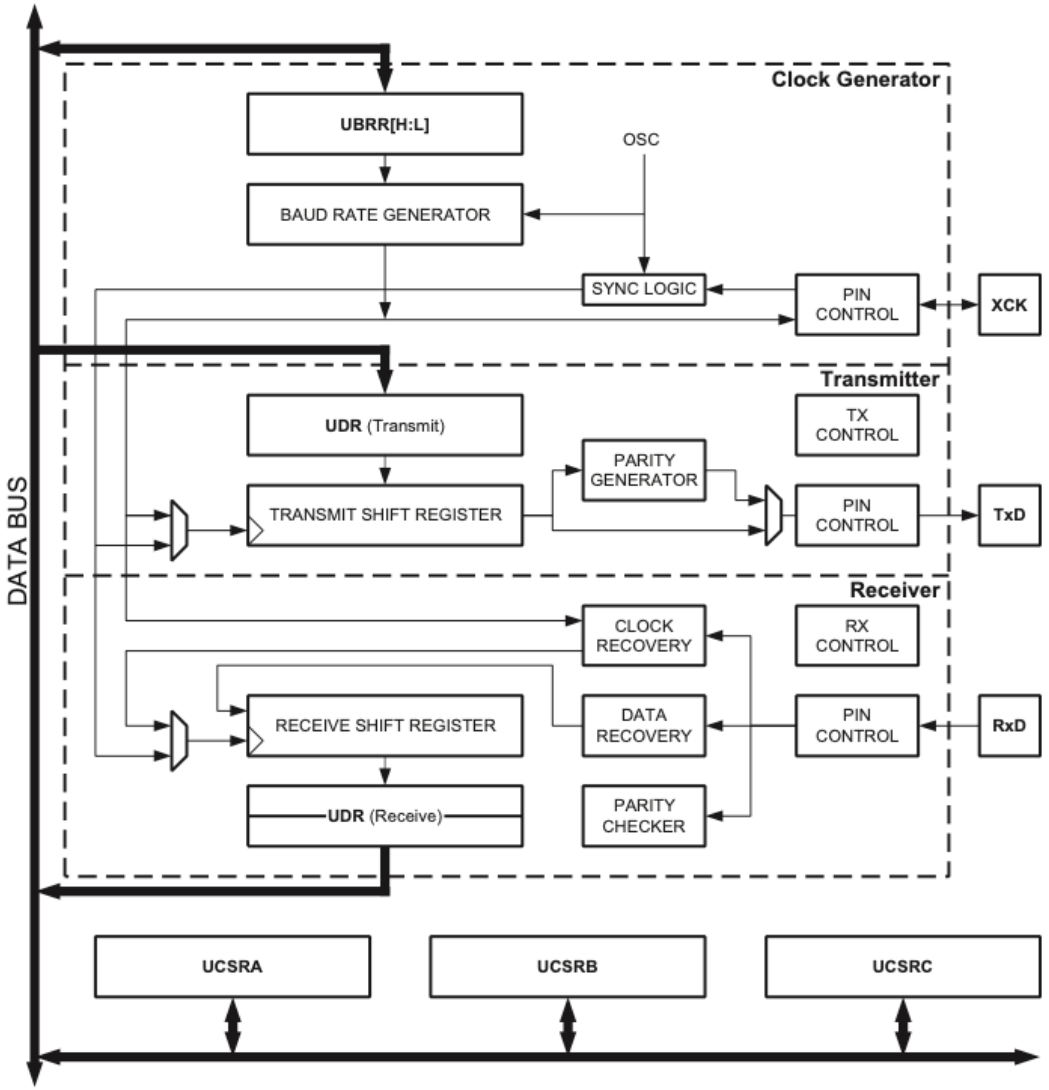
UART regiszterek

- UDR (UART Data Register):
  - UART adat regiszter
- UCSRA, UCSRB, UCSRC (UART Control and Status Regiszter)
  - UART kontrol és státusz regiszter, mely az UART beállításához tartozik
- UBRRH, UBRRL (UART Baudrate Register):
  - Az UART baudrate beállítására szolgáló 16 bites regiszter



UART regiszterek

Figure 22-1. USART Block Diagram[(1)]



UART - UDR regiszter

- UDR (UART Data Register)
  - UART adat regiszter
  - Valójában két két 8bites regiszterből (pufferből) áll (küldő/fogadó), melyhez ugyan az a cím tartozik.
  - Írásnál a küldő pufferbe kerül az adat:

    UDR = data;
  - fogadásnál pedig bejövő a pufferből kerül kiolvasásra:

    Data = UDR;

22.10.1 UDRn – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDRn (Read)
	TXB[7:0]								UDRn (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

UART - UCSRA regiszter

- RXCn (Receive Complete):
  - Bejövő adat beolvasása befejeződött, ki kell ovlasni az adatot
- TXCn (Transmit Complete):
  - Küldés befejeződött, újra lehet adatot írni az UDR regiszterbe

22.10.2 UCSRnA – USART Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

UART - UCSRB regiszter

- RXCIEn (RX Complete Interrupt Enable n):
  - Adatfogadást befejezéséhez rendelhető megszakítás bekapcsolása
- TXCIEn (TX Complete Interrupt Enable n):
  - Küldés befejezéséhez rendelhető megszakítás bekapcsolása
- RXENn (Receiver Enable n):
  - Vevő áramkör bekapcsolása (fogadás engedélyezése)
- TXEn (Transmitter Enable n):
  - Küldő áramkör engedélyezése (küldés engedélyezés)
- UCSZ2n (Character Size n):
  - Adatméret beállításához tartozó bit (UCSRnC-ben van másik 2 bit)

22.10.3 UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

UART - UCSRC regiszter

- UMSELn (UMSELn1:0 USART Mode Select):
  - Működési mód kiválasztása

Table 22-4. UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)[1]

- USBSn (Stop Bit Select):
  - Stop bit beállítása

Table 22-6. USBS Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

22.10.4 UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

UART - UCSRC regiszter

- UPMn1:0 (Parity Mode):
  - Paritás beállítása

Table 22-5. UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

- UCSZ (UCSZn1:0: Character Size):

Table 22-7. UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

22.10.4 UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

UART – UBRR regiszter

- UBRR11:0 (USART Baud Rate Register):
  - A soros kommunikáció sebessgét beállító regiszter
  - Az órajel a rendszer órajel leosztásával állítható be
  - 16 bites regiszter (felső és alsó 8 bit, a felsőből csak négy használható)
  - Az UBRR értékét az alábbi formulával lehet meghatározni:

Table 22-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$

  - Csak egész szám lehet! Adatlap alapján olyan sebesség legyen kiválasztva, aminek a hibaszázaléka maximum +-3%

22.10.5 UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRR[11:8]				UBRRHn
	UBRR[7:0]								UBRRLn
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

UART – UBRR regiszter

- Arduino Mega (ATmega2560): 16Mhz oscillator

Table 22-12. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4K	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2K	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8K	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4K	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6K	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8K	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2K	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4K	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250K	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max.(1)	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

```
#define F_CPU 16000000UL // 16 MHz
#define UART_BAUDRATE 9600 // 9600 bps
#define UBRR_ERTEK ((F_CPU / (USRT_BAUDRATE * 16UL)) - 1)
```

UART adatküldés

- Az `UART_Transmit(unsigned char data);` függvény egy adott karaktert tud elküldeni.
- Az UDR regiszterbe töltött adat automatikusan kiküldésre kerül.
- Minden UDR-be való írás előtt ellenőrizni kell, hogy a kimenő puffer üres!!!!
- Az UDR-ből a küldő shiftregiszterbe kerül az adat, melyet továbbít a vevő számára.
- Lehet, hogy már az UDR regiszter üres, de ez még nem azt jelenti, hogy készen áll az újabb küldésre!
- Ha kiürült a puffer, akkor azt az UCSRnA regiszterben található UDRE bit jelzi! (Logikai 1 értéket vesz fel)

UART fogadás

- Az `unsigned char UART_Receive(void)` függvény, az adat beolvasását valósítja meg.
- Fogadás során, a bejövő adat a bejövő pufferbe kerül. Ha beérkezett az összes bit (teljes adatcsomag), akkor az UDR regiszterbe áttöltésre kerül, majd az UCSRnA regiszter RXC bitje bebillen (logikai 1), ezzel jelezve, hogy az adat kiolvasható.
- Figyelem! A bejövő adat folyamatos kiolvasását biztosítani kell! Ha nem kerül kiolvasásra, akkor a következő adat felülírja az UDR regiszterben lévő adatot!

UART példa (polling)

```
1  #define CPU 16000000UL //16MHz
2  #define UART_BAUDRATE 9600 //9600bps
3  #define UBRR_VALUE ((CPU/(UART_BAUDRATE*16UL))-1)
4
5  void UART_Init(int ubrr);
6  void UART_Transmit(char data);
7  char UART_Receive();
8
9  char data;
10
11 void setup() {
12   UART_Init(UBRR_VALUE);
13 }
```

UART példa (polling)

```
22 void UART_Init(int ubrr)  // UART beallitasa
23 {
24   // 9600 bps sebesség beállítasa
25   // UBRR alsó 8 bitjének betöltése az UBRRL regiszterbe
26   UBRR0L = ubrr;
27   // UBRR felső 8 bitjének betöltése az UBRRH regiszterbe
28   UBRR0H = (ubrr >> 8);
29   // Aszinkron mód,
30   UCSR0C &= ~((1 << UMSEL01) | (1 << UMSEL00));
31   //8 adat bit
32   UCSR0B &= ~(1 << UCSZ02);
33   UCSR0C |= (1 << UCSZ00) | (1 << UCSZ01);
34   //nincs paritás
35   UCSR0C &= ~((1 << UPM01) | (1 << UPM00));
36   //1 stop bit
37   UCSR0C &= ~(1 << USBS0);
38   //Adó és vevő bekapcsolása
39   UCSR0B|=(1 << RXEN0)|(1 << TXEN0);   //
40 }
```

UART példa (polling)

```
43 void UART_Transmit(char data ) {
44   /* Várakozás amíg a TX buffer üres nem lesz */
45   while (!(UCSR0A & (1 << UDRE0)));
46   /* Put data into buffer, sends the data */
47   UDR0 = data;
48 }
49
50 char UART_Receive() {
51   /* Várakozás amíg adat nem érkezik */
52   while (!(UCSR0A & (1 << RXC0)))
53     ;
54   /* Adat kiolvasása az UDR regiszterből*/
55   return UDR0;
56 }
```

UART példa (polling)

```
15 void loop() {
16   data = UART_Receive();
17   UART_Transmit('[');
18   UART_Transmit(data);
19   UART_Transmit(']');
20 }
```

UART (megszakítással)

- A pollingolási módszernek a következő a hátránya:
  - 9600bps esetén 1/9600=104us időnként kerül elküldésre egy bit, egy adatcsomag pedig (beállítástól függően) 10bit-ből áll.
  - A fentiek alapján kb. 1ms-ként érkezik egy új adatcsomag. A folyamatos kiolvasást biztosítani kell! Ami ebben az esetben igen nagy kihívást jelent, ugyanis a fő programból kell biztosítani azt, hogy egy újabb bejövő adat ne írja felül a ki nem olvasott adatot az UDR regiszterben!
- A megszakításoknak köszönhetően ez könnyen megoldható!
- Egy megszakítás esetén a mikrovezérlő félbehagyja a főprogram futtatását, és a megszakításhoz tartozó ISR kiszolgáló rutinhoz ugrik.
- UART esetén az ”adatcsomag érkezett” megszakítás kerül kiváltásra.
- A megszakítás engedélyezéséhez az UCSRnB regiszter RXCIEn bitjét kell beállítani (logikai 1-re)

14.1 Interrupt Vectors in ATmega640/1280/1281/2560/2561

```
0x0032          jmp     USART0_RXC          ; USART0 RX Complete Handler
0x0034          jmp     USART0_UDRE         ; USART0,UDR Empty Handler
0x0036          jmp     USART0_TXC          ; USART0 TX Complete Handler
```

UART (megszakítással)

- Egészítsük ki a korábban megírt kódot!:

```
38    //Adó és vevő bekapcsolása
39    UCSR0B|=(1 << RXEN0)|(1 << TXEN0);    //
40    //megszakítás engedélyezése
41    UCSR0B |= (1 << RXCIE0);
42  }
```

22.10.3 UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

UNIVERSITAS BUDENSIS

UART (megszakítással)

```
57 //Interrupt Service Rutin
58 ISR(USART0_RX_vect) {
59   char data;
60   data = UART_Receive();
61   UART_Transmit('[');
62   UART_Transmit(data);
63   UART_Transmit(']');
64 }

15 void loop() {
16 }
```

22.10.3 UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	