Universitatea Tehnică din Cluj-Napoca
Departamentul Calculatoare

Proiectarea cu Micro-Procesoare

Lector: Mihai Negru

An 3 – Calculatoare și Tehnologia Informației

Seria B

Curs 6: Comunicare Serială

http://users.utcluj.ro/~negrum/

Dispozitive de Comunicare Serială la µC AVR

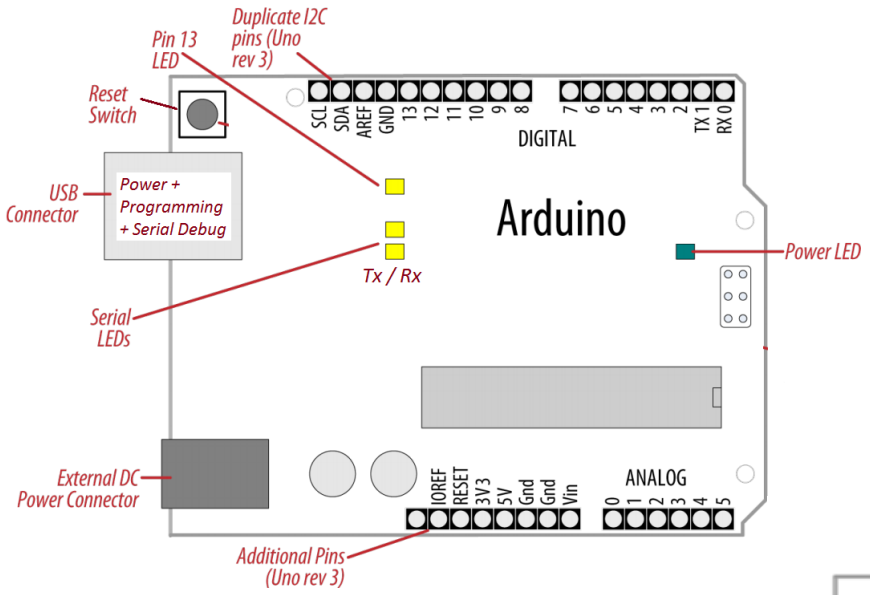
- **Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART)**
  - Comunicare serială sincronă sau asincronă
  - Frecvența (baud rate) variabilă
  - Suportă pachete de date de 5-9 biți, cu sau fară paritate
  - Suportă întreruperi pentru controlul transmisiei
  - Detecția erorilor de transmisie
  - Comunicare între placa de dezvoltare și PC

- **Serial Peripheral Interface (SPI)**
  - Comunicare serială sincronă
  - Mod de funcționare full duplex
  - Configurare Master sau Slave
  - Frecvență de comunicare variabilă
  - Se poate folosi pentru conexiune între plăci de dezvoltare, sau între o placă și diferite module PMOD (ex. DAC extern, Serial Flash)

Dispozitive de Comunicare Serială la µC AVR

- **Two Wire Serial Interface (TWI)**
  - Protocol de comunicare complex, folosind doar două fire (clock și data)
  - Implementare Atmel a protocolului I2C (Inter Integrated Circuit)
  - Controllerul TWI integrat suportă moduri master și slave
  - Adresare pe 7 biți
  - Suport pentru arbitrare pentru mai multe dispozitive master
  - Adresa slave programabilă
- **PS2**
- **CAN**
- **...**

Interfețe seriale la Arduino

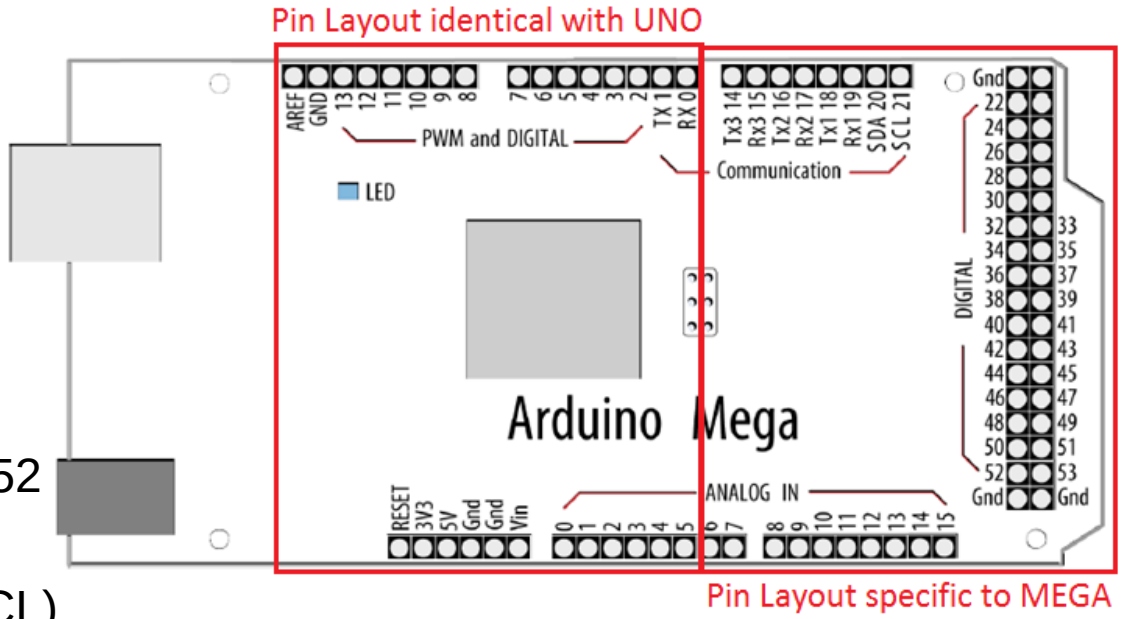


Arduino UNO (rev. 3)

- **UART**: pin 0 (RX) , pin 1 (TX) – Utilizaţi pentru programare!
- **SPI**: pin 10 (SS), pin 11 (MOSI), pin 12 (MISO), pin 13 (SCK).
- **TWI(I2C)**: A4 sau pin SDA, A5 sau pin SCL

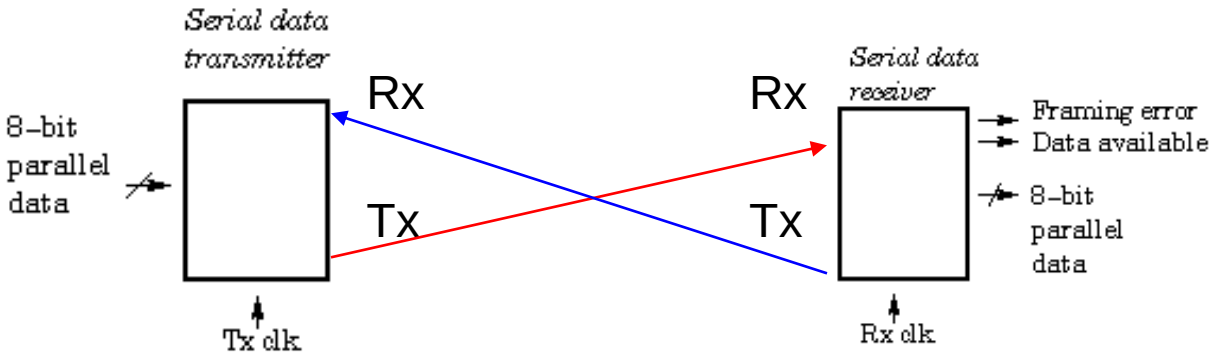
Arduino MEGA (rev. 3)

- **UART0**: pin 0 (RX), pin 1 (TX)
- **UART1**: pin 19 (RX), pin 18 (TX)
- **UART2**: pin 17 (RX) si 16 (TX)
- **UART3**: pin 15 (RX) si 14 (TX)
- **SPI:** pin 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS)
- **TWI(I2C)**: pin 20 (SDA) si 21 (SCL)



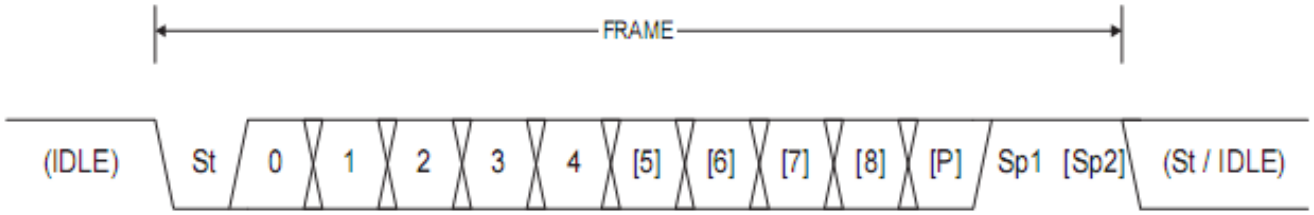
USART

- **USART – UART cu posibilitate de sincronizare prin semnal de ceas**
- UART – Interfață pentru comunicare serială asincronă
  - **Asincron**: intervalul dintre pachetele de date poate fi nedefinit. Destinatarul transmisiei detectează când începe și când se termină un pachet
  - **Baud rate** (rata de transfer): intervalul de timp dintre biți este fix, și trebuie cunoscut de ambele parți
  - Transmisia și recepția se pot efectua simultan (full duplex).
  - O interfață UART are două semnale:
    Rx – intrare, recepție
    Tx – ieșire, transmisie
  - USART are un semnal în plus, xck (external clock) care poate fi intrare sau iesire, și va sincroniza transmisia și recepția



USART

- **Transmisia datelor** → pachet (frame)
  - **St**: 1 bit de start, cu valoare ‘0’
  - **D**: 5, 6, 7, 8, 9 biți de date
  - **P**: 1 bit de paritate. Paritatea poate fi:
    - Absentă: bitul P nu există
    - Pară (*Even*)
    - Impară (*Odd*)
  - **Sp**: 1, 1.5 sau 2 biți de stop, cu valoare ‘1’

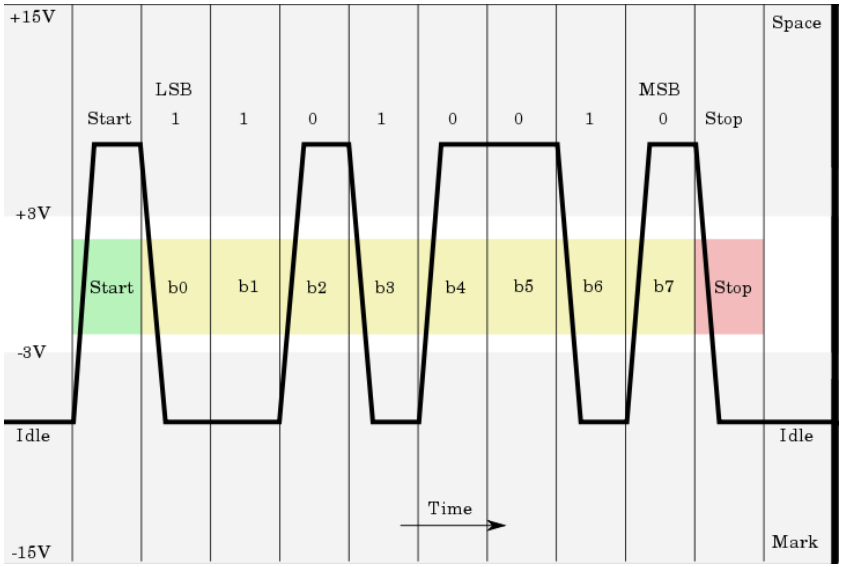


- **Recepția datelor**

1. Se detectează o tranziție din ‘1’ in ‘0’ pe linia Rx (recepție)
2. Se verifică mijlocul intervalului pentru bitul de start. Dacă este ‘0’, se inițiază secvența de recepție, altfel tranziția se consideră zgomot.
3. Se verifică mijlocul intervalului pentru biții următori (date, paritate, stop), și se reconstruiește pachetul de date
4. Dacă în poziția unde trebuie să fie biții de stop se detectează valoarea zero, se generează eroare de împachetare (**framing error**)
5. Dacă paritatea calculată la destinație nu corespunde cu bitul P, se generează eroare de paritate (**parity error**)

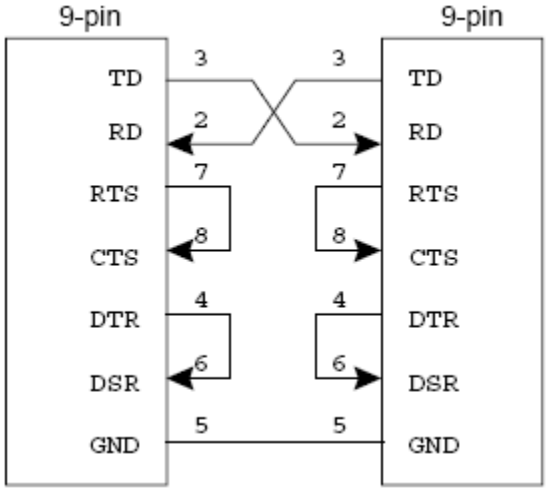
- Pentru robustețe, receptorul eșantionează semnalul de intrare la o frecvență de 8-16 ori mai mare decât *baud rate* – *oversampling*

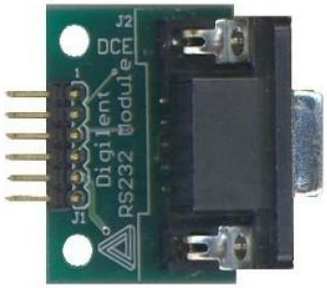
- **UART și RS232**
- Adaptarea nivelelor de tensiune
  - RS232 logic ‘1’ -5… -15 V
  - RS232 logic ‘0’ +5…+15 V

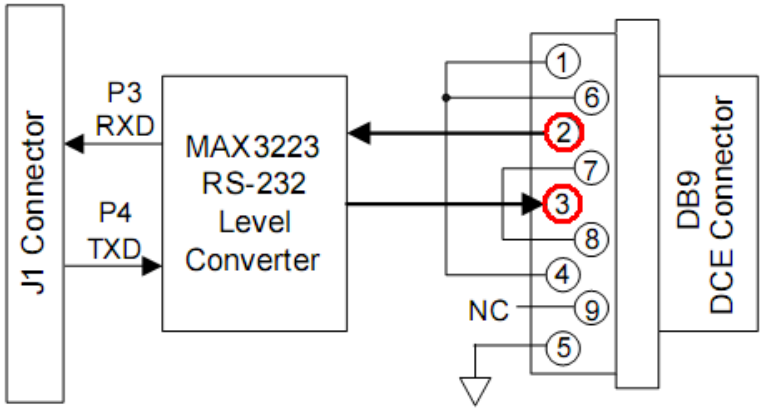


Este nevoie de conversie de la nivelele Logice UART la RS232

Corespondenta pinilor



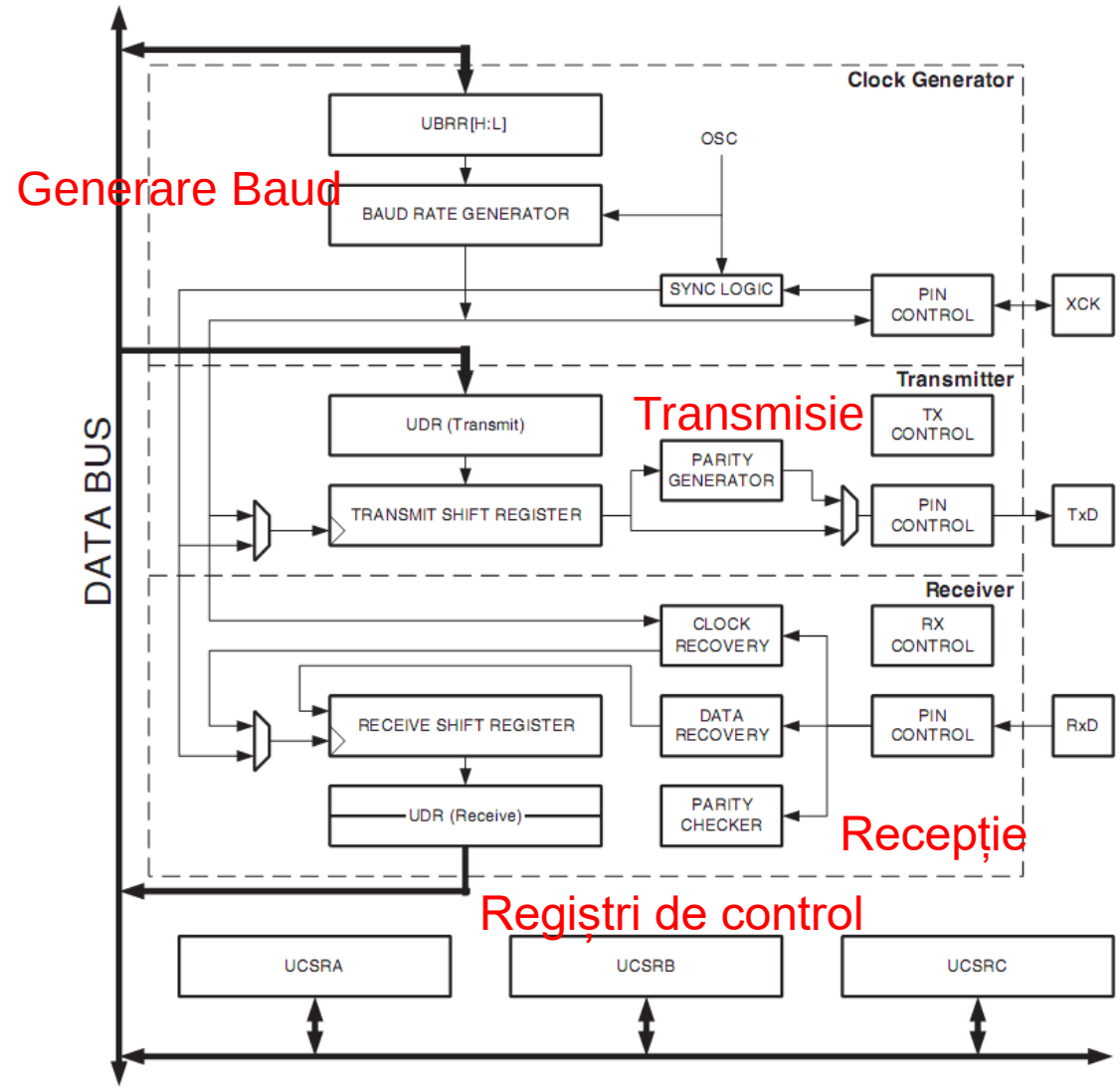




USART la AVR

- **Două unități USART:**
  - USART0 si USART1

- Arhitectura generală



USART la AVR – Configurare

- Registrul de control și stare **UCSRnA**

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	

- **RXCn** – Este ‘1’ cand recepția e completă. Poate genera întrerupere
- **TXCn** – Este ‘1’ cand transmisia e completă. Poate genera întrerupere
- UDREn – Data Register Empty, semnalează că registrul poate fi scris
- FEn – Semnalează eroare de impachetare (Frame Error)
- DORn – Data overrun – când se detectează un început de recepție înainte ca datele deja recepționate să fie citite din registrul de date
- UPEn – Eroare de paritate (Parity Error)
- U2Xn – Valoare ‘1’ → Dublare viteză de transmisie USART
- MPCMn – Activare mod de comunicare multiprocesor

USART la AVR – Configurare

- Registrul de control și stare **UCSRnB**

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	

- RXCIEn – Dacă e setat ‘1’, se generează întrerupere la terminarea receptiei
- TXCIEn – Dacă e setat ‘1’, se generează întrerupere la terminarea transmisiei
- UDRIEn – Dacă e setat ‘1’, se generează întrerupere când registrul de date e gol
- **RXEn** – activare receptie
- **TXEn** – activare transmisie
- UCSZn2 – combinat cu UCSZn1 si UCSZn0 din **USCRnC** stabilește mărimea pachetului
- RXB8n – al 9-lea bit receptionat, cand pachetul are 9 biti
- TXB8n – al 9-lea bit de transmis, cand pachetul are 9 biti

USART la AVR – Configurare

- Registrul de control și stare **UCSRnC**

Bit	7	6	5	4	3	2	1	0	
	–	UMSELn	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	

- **UMSELn** – Mod asincron ‘0’ sau sincron ‘1’
- UPMn1 si UPMn0 – Selectia modului de paritate
- **USBSn** – configurare biti de stop: ‘0’ – 1 bit, ‘1’ – 2 biți
- **UCSZn1:UCSZn0** – combinat cu UCSZn2 din UCSRnB → dimensiunea pachetului

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

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

USART la AVR – Configurare

- Registrii de control al frecvenței: **UBRRnH** si **UBRRnL**
  - Formează împreună valoarea **UBRRn**, de 12 biți

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	UBRRn[11:8]				UBRRnH
	UBRRn[7:0]								UBRRnL
	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	

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR + 1n)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn + 1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$

- **Citire date recepționate / scriere date pentru transmis**
  - Ambele acțiuni se fac prin registrul **UDRn**

USART la AVR

- **Pini comuni cu porturile I/O –** Direcția este configurată automat prin activarea recepției și/sau a transmisiei!

Port Pin	Alternate Function
PD7	T2 (Timer/Counter2 Clock Input)
PD6	T1 (Timer/Counter1 Clock Input)
PD5	XCK1[(1)] (USART1 External Clock Input/Output)
PD4	ICP1 (Timer/Counter1 Input Capture Pin)
PD3	INT3/TXD1[(1)] (External Interrupt3 Input or UART1 Transmit Pin)
PD2	INT2/RXD1[(1)] (External Interrupt2 Input or UART1 Receive Pin)
PD1	INT1/SDA[(1)] (External Interrupt1 Input c
PD0	INT0/SCL[(1)] (External Interrupt0 Input c

USART1, portul D

Port Pin	Alternate Function
PE7	INT7/ICP3[(1)] (External Interrupt 7 Input or Timer/Counter3 Input Capture Pin)
PE6	INT6/ T3[(1)] (External Interrupt 6 Input or Timer/Counter3 Clock Input)
PE5	INT5/OC3C[(1)] (External Interrupt 5 Input or Output Compare and PWM Output C for Timer/Counter3)
PE4	INT4/OC3B[(1)] (External Interrupt 4 Input or Output Compare and PWM Output B for Timer/Counter3)
PE3	AIN1/OC3A [(1)] (Analog Comparator Negative Input or Output Compare and PWM Output A for Timer/Counter3)
PE2	AIN0/XCK0[(1)] (Analog Comparator Positive Input or USART0 external clock input/output)
PE1	PDO/TXD0 (Programming Data Output or UART0 Transmit Pin)
PE0	PDI/RXD0 (Programming Data Input or UART0 Receive Pin)

USART0, portul E

USART la AVR

- **Exemplu:** comunicare între ATmega2560 si PC – ECHO simplu
- **Necesar:** cablu serial, modul **PMOD RS232**

1. **Configurare**
   Baud: 9600
   Marime pachet: 8 biti
   Biți de stop: 2
   Paritate: fără paritate

$$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$$

f_{osc} = 16 MHz → 16.000.000

UBRRn = 103

2. **Așteptare recepție caracter**
   - Verificare **RXCn** din **UCSRnA**, așteptare până devine 1
3. **Citire caracter recepționat, din UDRn**
4. **Scriere caracter de transmis, în UDRn**
5. **Așteaptă transmisie caracter**
   - Verificare **TXCn** din **UCSRnA**, așteptare până devine 1
6. **Salt la 2**

USART la AVR

- **Exemplu:** comunicare între ATmega64 si PC – ECHO simplu

```
ldi r16, 0b00011000              ; activare Rx si Tx
sts UCSR1B,r16
ldi r16, 0b00001110              ; dimensiune frame 8 biti, fara paritate, 2 biti de stop
sts UCSR1C,r16
ldi r16, 103                     ; Baud rate calculat, incape in primii 8 biti
ldi r17, 0                       ; Bitii superiori la UBRR sunt zero
sts UBRR1H, r17
sts UBRR1L, r16

mainloop:
    recloop:
        lds r20, UCSR1A
        sbrs r20, 7              ; bitul 7 din UCSR1A – receptie completa
    rjmp recloop

    lds r16, UDR1                ; citire date receptionate
    sts UDR1,r16                 ; scriere date spre transmisie

    txloop:
        lds r20, UCSR1A          ; asteptare terminare transmisie
        sbrs r20, 5
        rjmp txloop
    rjmp mainloop
```

UART la Arduino

- **Toate placile Arduino** au cel putin un port serial (port UART sau USART), accesibil prin obiectul C++ **Serial**
- Comunicare µC ↔ PC prin conexiunea USB, folosind adaptorul USB-UART integrat pe placă – comunicare folosită şi pentru programarea plăcii!!
- Comunicarea între plăci folosind pinii digitali 0 (RX) și 1 (TX) – nerecomandată. Pentru a putea folosi aceste tipuri de comunicație, nu folosiți pinii 0 și 1 pentru operații generale de I/O digital !!!

- **Placa Arduino MEGA** are trei porturi seriale suplimentare: **UART1** folosește pinii 19 (RX) și 18 (TX), **UART2** pinii 17 (RX) și 16 (TX), **UART3** pinii 15 (RX) și 14 (TX)
- Pentru a comunica cu un PC, este nevoie de un adaptor USB-UART extern, sau un adaptor UART-RS232, deoarece aceste interfețe nu folosesc adaptorul integrat pe placa
- Pentru a comunica cu un dispozitiv care folosește interfața serială cu nivele logice TTL, se conectează pinul TX al plăcii la pinul RX al dispozitivului, pinul RX al plăcii la pinul TX al dispozitivului, și pinii de masă (GND) împreuna

UART la Arduino

- Biblioteca **Serial** integrată în mediul de dezvoltare Arduino (http://arduino.cc/en/Reference/Serial) – folosită pentru facilitarea comunicației prin interfețele seriale disponibile.

- Metodele clasei **Serial** (selectie):
- **Serial.begin(speed)** – configurează viteza de transmisie (speed) și formatul implicit de date (8 data bits, no parity, one stop bit)
- **Serial.begin(speed, config)** – configurează viteza (speed) + selectează un alt format al datelor (config): SERIAL_8N1 (implicit), SERIAL_7E2, SERIAL_5O1
- **Serial.print(val)** – trimite valoarea val ca un șir de caractere citibil de catre utilizator (ex: Serial.print(20) va trimite caracterele ‘2’ și ‘0’)
- **Serial.print(val, format)** – format specifică baza de numerație folosită (BIN, OCT, DEC, HEX. Pentru numere în virgulă mobilă, format specifică numărul de zecimale folosit.
- **Serial.println** – Trimite datele + \r\n (ASCII 13 + ASCII 10)
- **Example:**

Serial.print(78) transmite "78“ Serial.print(78, BIN) transmite "1001110"

Serial.print(1.23456) transmite "1.23“ Serial.println(1.23456, 4) transmite "1.2346"

Serial.print("Hello") transmite "Hello"

UART la Arduino

- byte IncomingByte = **Serial.read()** – citește un byte prin interfața serială
- int NoOfBytesSent = **Serial.write(data)** – scrie date în format binar prin interfața serială. Datele se pot scrie ca un byte (val) sau ca un șir de octeți specificat ca un string (str) sau ca un șir buf, de lungime specificată len (buf, len)
- **Serial.flush**() – așteaptă până când transmisia datelor pe interfața serială este completă.
- int NoOfBytes = **Serial.available()** – Returnează numărul de octeți disponibili pentru a fi citiți prin interfața serială. Datele sunt deja primite și stocate într-o zonă de memorie buffer (capacitate maximă 64 octeți)
- **serialEvent()** – funcție definită de utilizator, care este apelată automat când există date disponibile in zona buffer. Folosiți **Serial.read()** în această funcție, pentru a citi aceste date.
- serialEvent1(), serialEvent2(), serialEvent3() – Pentru Arduino Mega, funcții care se apelează automat pentru celelalte interfețe seriale.

UART la Arduino

Exemplul 1:

```
void setup() {
    Serial.begin(9600);            // deschide portul serial, configureaza viteza la 9600 bps
    Serial.println(“Hello”);
}
void loop() {}
```

Exemplul 2 (doar pentru Arduino Mega):

```
// Arduino Mega foloseste patru porturi seriale  (Serial, Serial1, Serial2, Serial3),
// se pot configura cu viteze diferite:
void setup(){
  Serial.begin(9600);
  Serial1.begin(38400);
  Serial2.begin(19200);
  Serial3.begin(4800);

  Serial.println("Hello Computer");
  Serial1.println("Hello Serial 1");
  Serial2.println("Hello Serial 2");
  Serial3.println("Hello Serial 3");
}
void loop() {}
```

UART la Arduino

Comunicație µC ↔ PC, Exemplul 3 – recepționează o cifră (caracter de la ‘0’ la ‘9’) și modifică starea unui LED proporțional cu cifra citită

```
const int ledPin = 13;              // pin LED
int blinkRate=0;                    // rata de modificare a starii
void setup()
{
  Serial.begin(9600); // initializare port serial
  pinMode(ledPin, OUTPUT); // configurare pin LED ca iesire
}
void loop() {
  if ( Serial.available())          // Verifica dacă avem date de citit
  {
        char ch = Serial.read(); // citeste caracterul receptionat
        If( isDigit(ch) )           // Este cifră ?
        {
            blinkRate = (ch - '0');             // Se converteste in valoare numerica
            blinkRate = blinkRate * 100;    // Rata = 100ms * cifra citita
        }
  }
  blink();
}
```

Exemplul 3 – cont.

```
// modifica stare LED pe baza ratei calculate
void blink()
{
  digitalWrite(ledPin,HIGH);
  delay(blinkRate); // intarzierea calculata
  digitalWrite(ledPin,LOW);
  delay(blinkRate);
}
```

- **Pentru a utiliza acest exemplu:**
- Folositi Serial Monitor, inclus in mediul Arduino, activându-l din meniul Tools sau apăsând <CTRL+SHIFT+M>)
- Selectați aceeași viteză cu cea pe care ați configurat-o apelând Serial.begin()
- Tastați cifre, și apăsati butonul “Send”.

Exemplul 4 – exemplul 3 modificat să folosească **serialEvent()**

```
void loop()
{
  blink();
}

void serialEvent() // functia se apeleaza automat, cand exista date de citit
{
  while(Serial.available()) // cat timp exista date disponibile
  {
     char ch = Serial.read(); // acestea se citesc
     Serial.write(ch); // echo – trimitem datele inapoi pentru verificare
     if( isDigit(ch) ) // e cifra ?
     {
        blinkRate = (ch - '0'); // convertire la valoare numerica
        blinkRate = blinkRate * 100; // calcul timp de intarziere
     }
   }
}
```

Software UART la Arduino

- În afara interfețelor seriale hardware, Arduino permite comunicarea serială și pe alți pini digitali, realizând procesul de serializare / de-serializare a datelor prin program
- Se folosește biblioteca SoftwareSerial, inclusă în pachetul software Arduino (necesită includerea header-ului softwareserial.h)
- Pinul de receptie **(RX) trebuie conectat la un pin digital care suporta întreruperi externe** declanșate prin schimbarea stării acestuia:
  - La Arduino Mega, acesti pini sunt: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 (69)
- Se pot crea mai multe obiecte C++ de tip SoftwareSerial, dar numai unul poate fi activ la un moment dat
- Crearea unui obiect – trebuie specificați pinii de recepție și transmisie:

SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin)

- Sunt implementate functiile **begin**, **read**, **write**, **print**, **println**, care se folosesc în același mod ca în cazul interfețelor seriale hardware

Software UART la Arduino

- **Exemplu:** comunicarea folosind două interfețe seriale, una hardware, conectată la PC, și una software, conectată la un alt dispozitiv compatibil UART
  - Arduino joaca rolul de releu de comunicație: ce primește pe o interfată, transmite pe cealaltă.

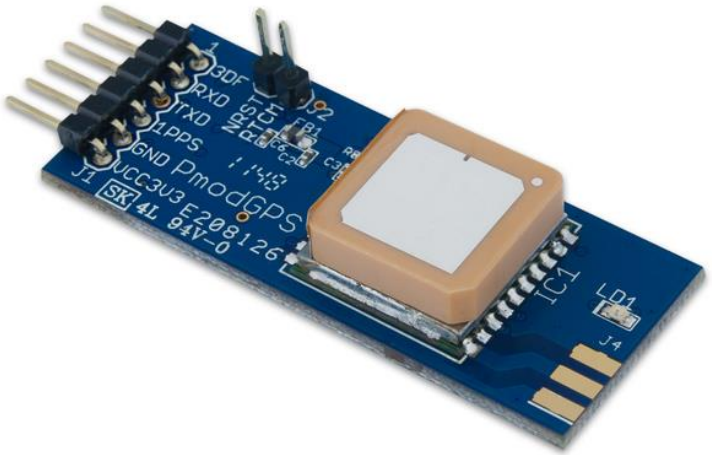
```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11); // Interfata software foloseste pin 10 pt RX, pin 11 pt TX

void setup()
{
 Serial.begin(9600);            // Configurare interfata hardware
 mySerial.begin(9600);          // Configurare interfata software
}

void loop()
{
  if (mySerial.available())
    Serial.write(mySerial.read()); // citire de pe interfata software, transmisie prin cea hardware
  if (Serial.available())
    mySerial.write(Serial.read()); // citire de pe interfata hardware, transmisie prin cea software
}
```

Software UART la Arduino

- **Exemplu:** comunicarea folosind două interfețe seriale, una hardware, conectată la PC, și una software, conectată la un alt dispozitiv compatibil UART
- Posibilă utilizare a programului anterior: vizualizarea datelor transmise de către un dispozitiv compatibil UART pe terminalul Serial Monitor
- Exemplu: receptor GPS Digilent PMod GPS:




```
$GPGGA,064951.000,2307.1256,N,12016.4438,E,1,8,0.
95,39.9,M,17.8,M,,*65
$GPGSA,A,3,29,21,26,15,18,09,06,10,,,,,2.32,0.95,
2.11*00
$GPGSV,3,1,09,29,36,029,42,21,46,314,43,26,44,020
,43,15,21,321,39*7D
$GPRMC,064951.000,A,2307.1256,N,12016.4438,E,0.03
,165.48,260406,3.05,W,A*55
$GPVTG,165.48,T,,M,0.03,N,0.06,K,A*37
```

Codurile ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Codurile ASCII Extinse

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	⊕	249	·
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	▉	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Source: www.LookupTables.com

Referințe

1. Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V datasheet
2. Atmel Atmega64 datasheet
3. Atmel Atmega2560 datasheet