

Sisteme cu MicroProcesoare

Cursul 8

Sisteme de comunicații

Conf. Tiberiu COCIAS

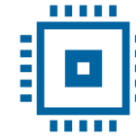


Universitatea
Transilvania
din Brașov

FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR



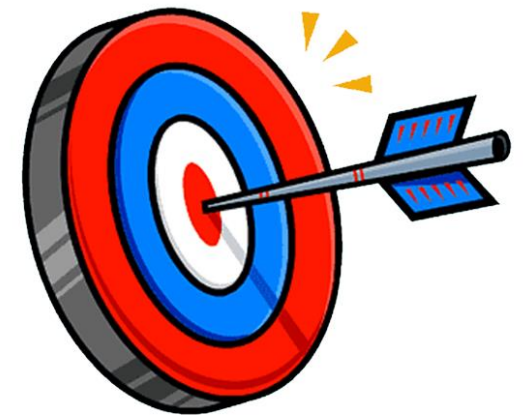
Cuprins



- Introducere în sistemele de comunicații
- Comunicația UART
- Comunicația USART
- Standardele RS-232 și RS-485
- Comunicația SPI

Obiectivul cursului

- Prezentarea principalelor caracteristici ale unui sistem de comunicaţii şi aprofundarea cunoştinţelor pentru două protocoale:
 - Caracteristici ale sistemelor de comunicaţie
 - Clasificarea acestora
 - Prezentarea comunicaţiei serială asincronă şi sincronă: UART si USART
 - Prezentarea caracteristicilor electrice pentru două standarde de comunicaţie
 - Prezentarea protocolului SPI

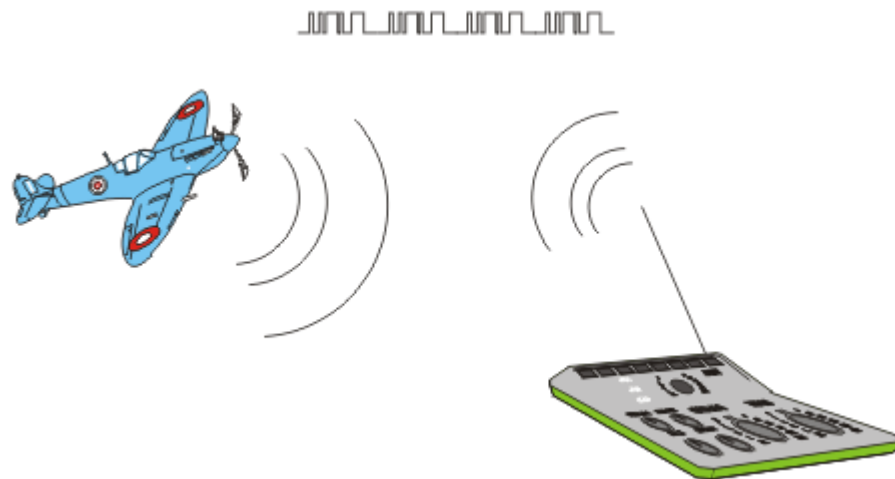
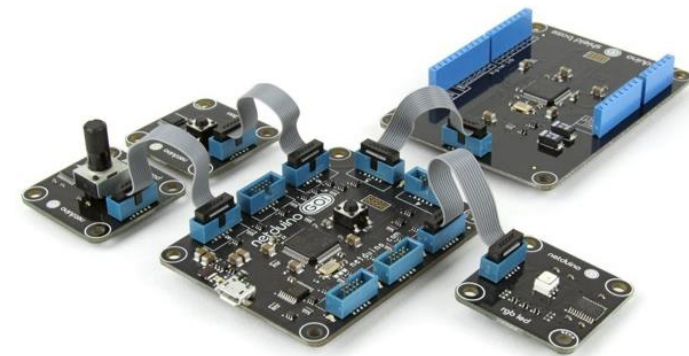


Sisteme de comunicație



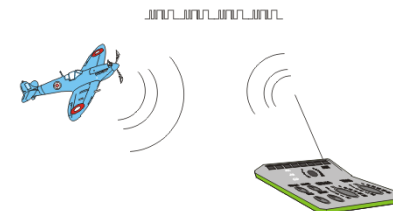
Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

- De ce avem nevoie de sisteme de comunicații?
- Unde se folosesc acestea?

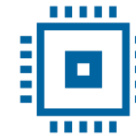


Sisteme de comunicație

- De ce avem nevoie de sisteme de comunicații?
 - Sistemele de comunicații la microcontrolere sunt esențiale pentru interconectarea și coordonarea dispozitivelor într-un sistem integrat (embedded)
- Unde se folosesc acestea?
 - Interconectarea cu periferice externe: senzori, actuatoare, module de stocare sau afișaje
 - Comunicarea între mai multe microcontrolere: permite colaborarea pentru a îndeplini sarcini distribuite
 - Conexiunea cu dispozitivele specifice pentru utilizator: calculatoare, smartphone sau tablete
 - Integrarea în rețele și IoT (Internet of Things): comunica cu servere, aplicații mobile sau alte dispozitive prin protocoale de rețea.

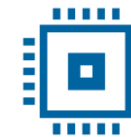


Tipuri de comunicație

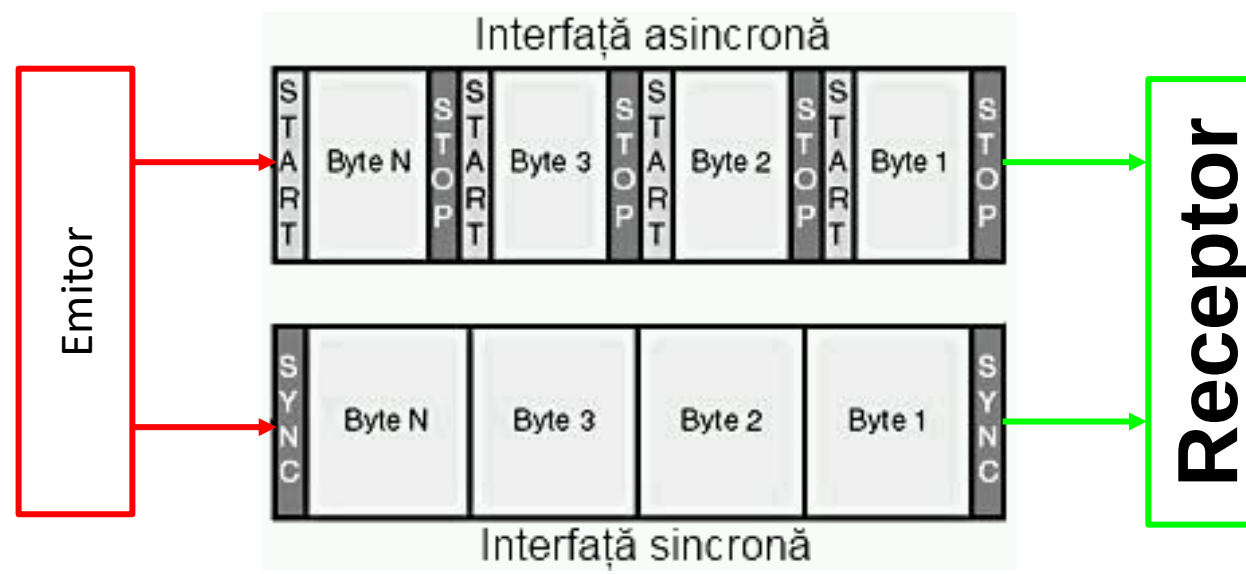


- Pe baza arhitecturii hardware
 - **Comunicații seriale:**
 - Transferul datelor bit cu bit pe un singur fir
 - Exemple: UART, SPI, I²C, RS-232.
 - **Comunicații paralele:**
 - Transferul datelor pe mai multe fire simultan
 - Mai rapide, dar necesită mai multe pini (mai rar utilizate pe microcontrolere moderne)
- Pe baza mediului de transmisie:
 - **Comunicare prin fir (wired):**
 - Protocoale seriale clasice (SPI, I²C)
 - Ethernet pentru aplicații de rețea
 - **Comunicare wireless:**
 - Bluetooth, Wi-Fi, ZigBee
- Sincronizare:
 - **Sincrone:** Comunicarea este sincronizată printr-un semnal de ceas (clock), ex. SPI.
 - **Asincrone:** Nu necesită semnal de ceas, ex. UART.

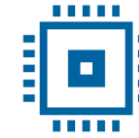
Comunicații seriale sincrone și asincrone



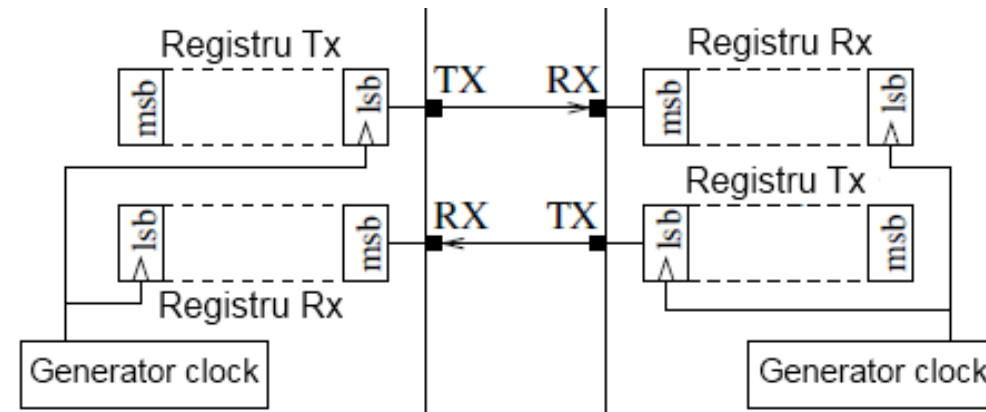
- *Comunicațiile seriale reprezintă transferul datelor bit cu bit, folosind un singur fir pentru transmisie (sau mai multe, în funcție de protocol)*
- Acestea pot fi clasificate în:
 - **Asincrone:** Nu necesită un semnal de ceas comun; sincronizarea se face folosind semnale de start/stop: UART
 - **Sincrone:** Utilizează un semnal de ceas pentru sincronizarea expeditorului și receptorului: USART, SPI, I2C



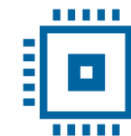
UART (Universal Asynchronous Receiver Transmitter)



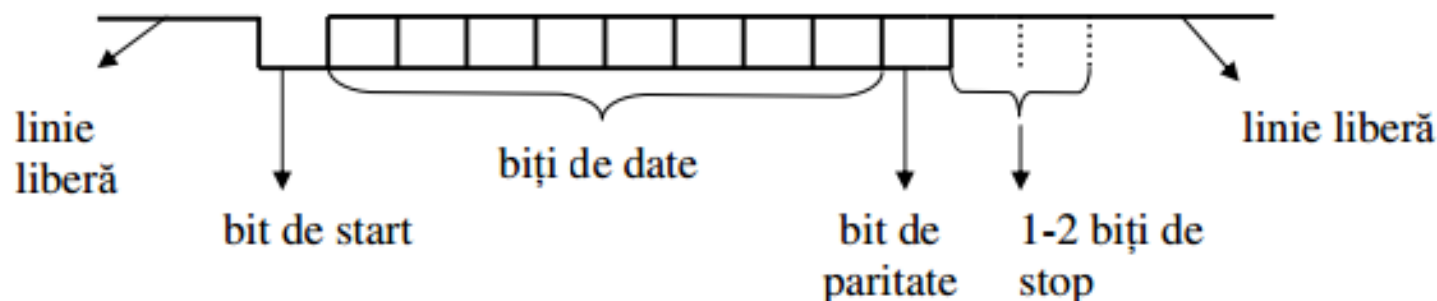
- UART este un protocol de comunicație serială asincronă utilizat pe scară largă în sistemele embedded pentru transferul de date între microcontrolere, periferice sau alte dispozitive.
- Nu necesită un semnal de ceas comun între dispozitivele care comunică
- Utilizează două linii: receptor (RxD) și emitor (TxD)
- Are mai mulți parametri configurabili:
 - Număr biți date: Dependent de producător, în general între 5 și 9 biți de date. Lsb se transmite primul
 - Bit paritate: bit care să marcheze paritatea: pară sau impară
 - Ex. paritate pară: bit paritate = 1 dacă nr. biți din mesaj este par
 - Rată transfer: Viteza de transmisie (biți/sec sau bps)
 - Valori cuprinse între 9600-115200 (depinde de clock)
 - Bit stop
 - Permite utilizarea unui singur bit de stop, sau doi biți de stop



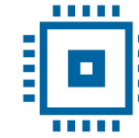
UART – Formatul mesajului



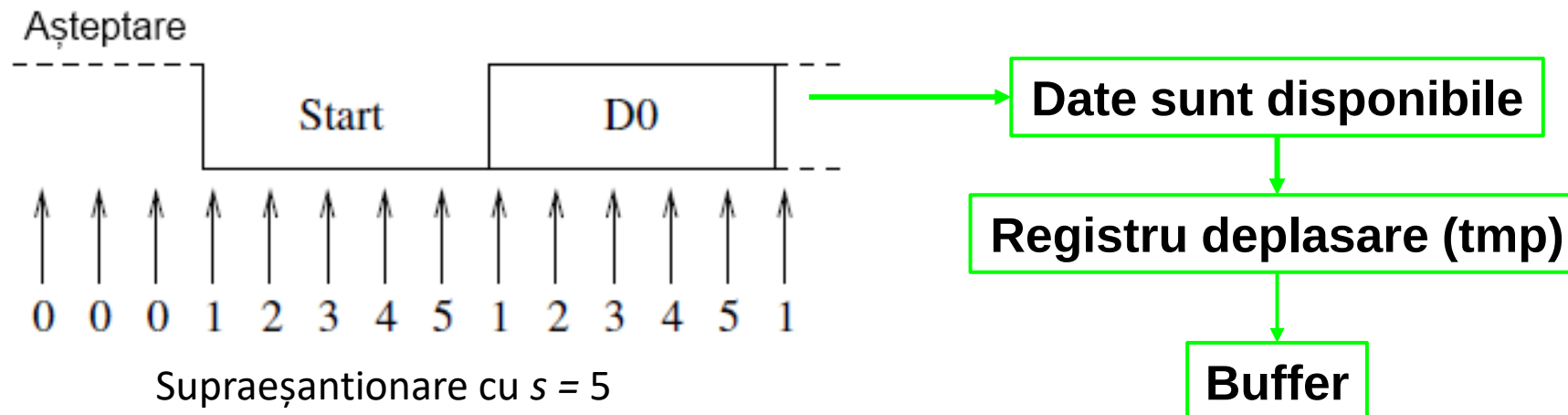
- Utilizează NRZ (Non Return to Zero)
- Următoarea structură de date este transmisă:
 - un bit de start (0 logic)
 - 5-8 biți de date
 - 0-1 bit de paritate (pentru detecția erorilor)
 - 1-2 biți de stop (1 logic)
- În starea de așteptare (liberă) linia este High
- Cel mai puțin semnificativ bit este transmis primul



UART – Sincronizare E/R

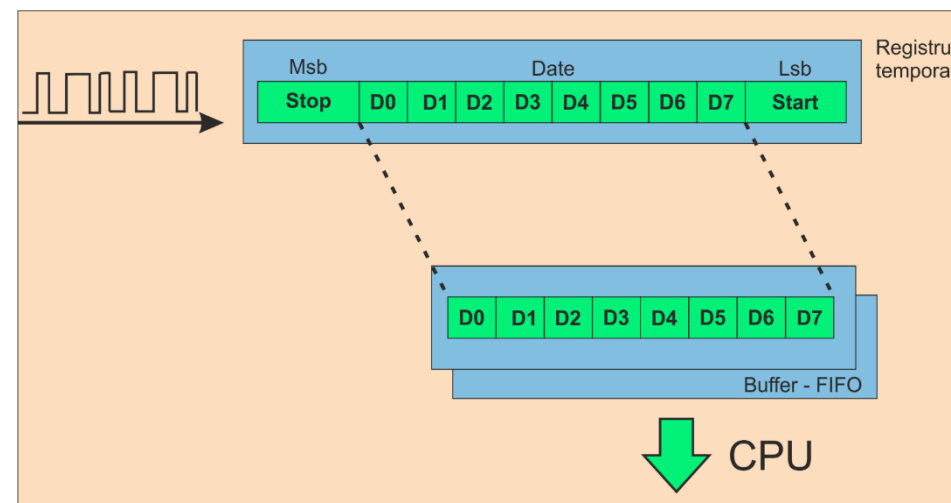


- La comunicația asincronă E și R au semnal de clock independent
- R realizează supraeșantionare pentru sincronizarea cu E
 - **RxD este eșantionată de s ori pentru fiecare bit**
 - În general, s are valoarea 16
 - Pentru Atmega16 eșantioanele 8, 9, 10 decid starea liniei
 - **Un registru cu deplasare și un buffer mai sunt necesari**

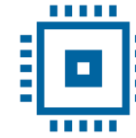


UART – Registre importante

- Registre de configurare şi control:
 - Baud Rate Register - Configurează viteza de transmisie a datelor (baud rate).
 - Control Register: Permite activarea transmisiei şi recepţiei, configurarea biţilor de stop, a parităţii şi a dimensiunii cadrului de date.
 - Status Register: Indică starea curentă a UART (ex. transmisie completă, date disponibile pentru recepţie, erori).
- Registre de date:
 - Un registru de deplasare
 - Un registru buffer (parte din FIFO)

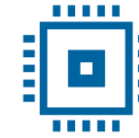


UART – Calcul rată transfer date (bps)

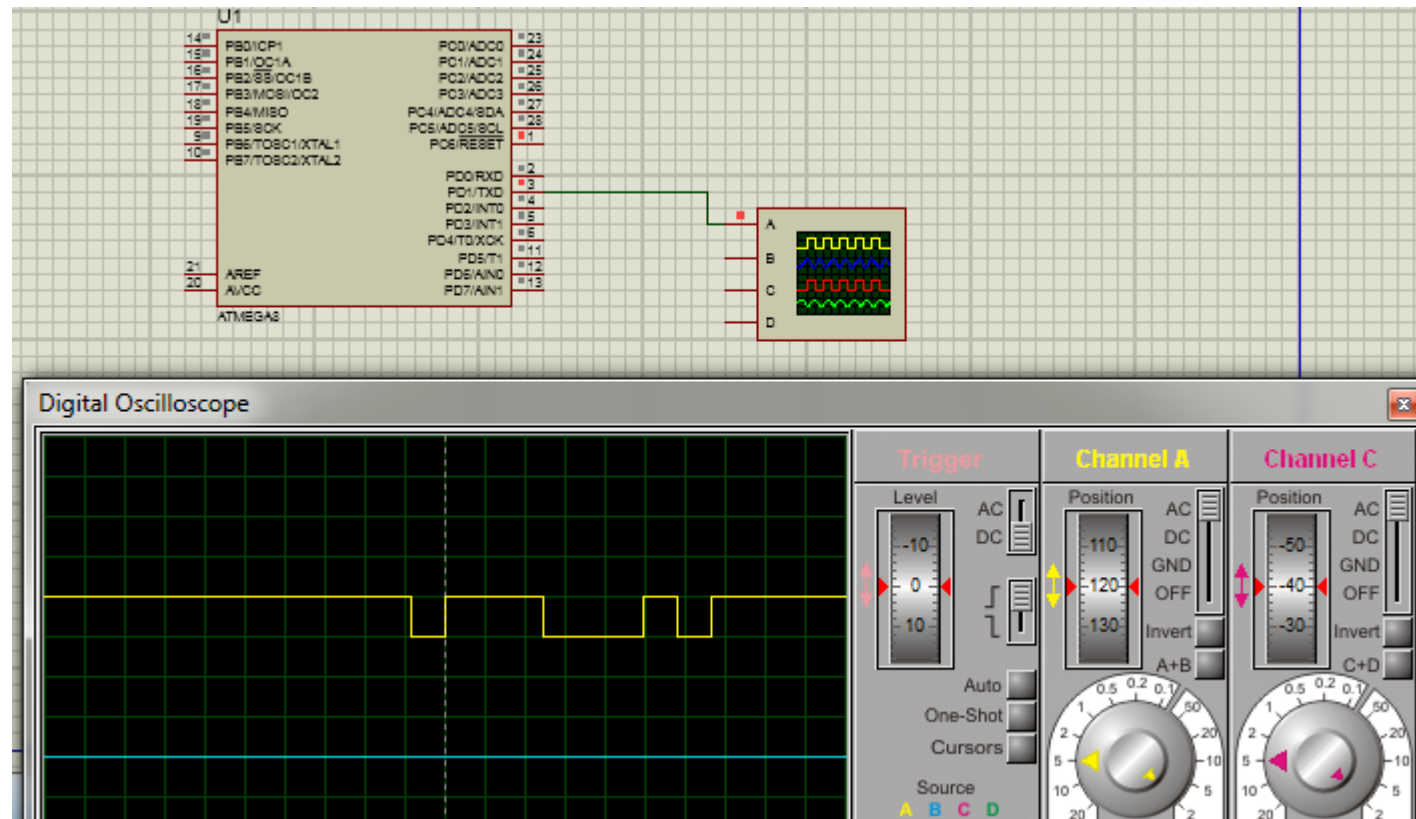


- Rata de transfer pornește de la clock-ul sistemului (registrul ratei de transfer)
- Clock-ul este scalat cu s de un registru prescalar doar la receptor
- Nu poate fi generată orice rată de transfer:
 - Ex: 8MHz, $s = 8$ și $\text{rată_transfer} = 115.2 \text{ kbps}$
 - Prescalar este: $(8\text{Mhz}/s)/115.2 = 8.68$ (!!nu e număr întreg) ≈ 9
 - $\text{Rată_transfer } rt = f/(s \cdot \text{contor}) = 8\text{Mhz}/(8 \cdot 9) = 111.1$
 - Eroarea $e = ((rt_{\text{obținută}})/(rt_{\text{dorită}}) - 1) \cdot 100\% = -3.5\%$
 - În general eroarea trebuie să fie $\pm 2.5\%$

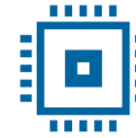
Aplicație - UART



- Exemplu de utilizare a modului UART
 - Cerință: să se implementeze un program care transmite un caracter pe serială (exemplu "G")



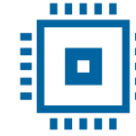
Aplicație - UART



■ Inițializare UART: 8N1

```
// Parametri comunicație 8N1: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
UCSRA=(0<<RXC) | (0<<TXC) | (0<<UDRE) | (0<<FE) | (0<<DOR) |
(0<<UPE) | (0<<U2X) | (0<<MPCM);
UCSRB=(0<<RXCIE) | (0<<TXCIE) | (0<<UDRIE) | (1<<RXEN) | (1<<TXEN)
| (0<<UCSZ2) | (0<<RXB8) | (0<<TXB8);
UCSRC=(1<<URSEL) | (0<<UMSEL) | (0<<UPM1) | (0<<UPM0) | (0<<USBS)
| (1<<UCSZ1) | (1<<UCSZ0) | (0<<UCPOL);
UBRRH=0x00; // registrul contor MSB
UBRRL=0x33; //registrul contor LSB
```

Aplicație - UART



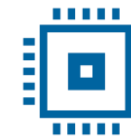
- Transfer date – funcții default

```
//transfer date, la fiecare 500 ms
while (1)
{
    putchar('G'); //transmit caracterul 'G'

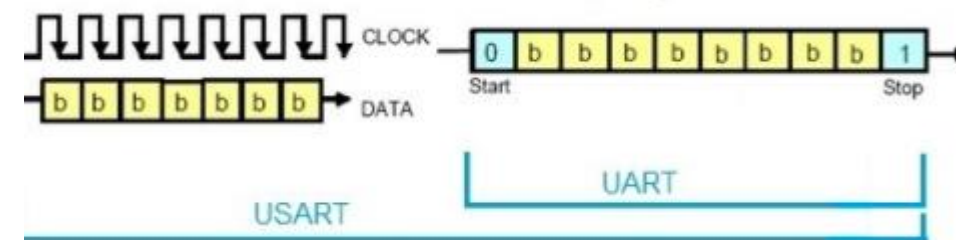
    delay_ms(500); //aștept 500 ms
}
```

- Pentru citire se poate utiliza funcția getchar()

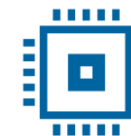
USART (Universal Synchronous and Asynchronous Receiver-Transmitter)



- Este o versiune mai avansată a modulelor UART, deoarece suportă atât comunicația asincronă, cât și comunicația sincronă
- Interfețele sincrone folosesc pentru receptor un clock sincronizat cu cel al emitorului.
 - Avantajul este un risc mic de erori de sincronizare între E și R
 - Semnalul este generat de unul dintre partenerii de comunicație
- Se pot trimite mai multe date în varianta sincronă, datorată semnalului de clk folosit la sincronizare
- USART păstrează logica ambelor moduri de funcționare, sincron și asincron
 - Dacă modul asincron este utilizat linia de clock nu este utilizată și poate fi folosită ca linie de I/O generală



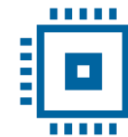
Standardele RS-232 și RS-485



- UART/USART descriu modul de comunicație serială asincronă/sincronă fără specificații tehnice legate de nivelurile de tensiune
- RS – 232/485: standarde utilizate în general pentru comunicații unu-la-unu
 - Are definite echipamentele care pot fi utilizate
 - Sunt definite specificațiile electrice
 - Sunt definite liniile de semnal
- Caracteristici specifice:

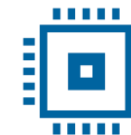
Caracteristică	RS-232	RS-485
Topologie	Punct-la-punct	Multi-punct
Distanță maximă	15m	1200m
Număr de dispozitive	2	Max 32
Imunitate zgomote	Mică	mare
Tensiune de utilizare	$\pm 12V$	$\pm 5V$

Comunicația SPI

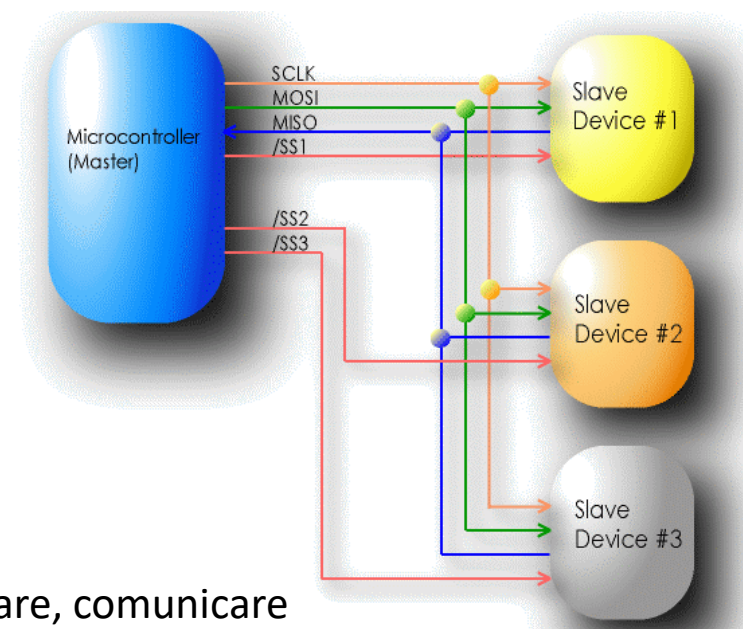


- SPI: *Serial Peripheral Interface*
- Comunicație serială:
 - Sincronă, de tip punct la punct
 - Utilizează principiul master-slave (un master poate controla unul sau mai multe dispozitive slave)
 - Full-duplex, master (MC), slave (dispozitive periferice)
 - Sincronizarea se poate face pe ambele fronturi ale clock-ului
 - Utilizează 4 linii de comunicație:
 - MOSI (Master Out, Slave In): utilizată de master pentru a transmite date către slave
 - MISO (Master In, Slave Out): utilizată de slave pentru a transmite date către master

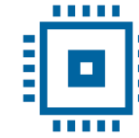
Comunicația SPI



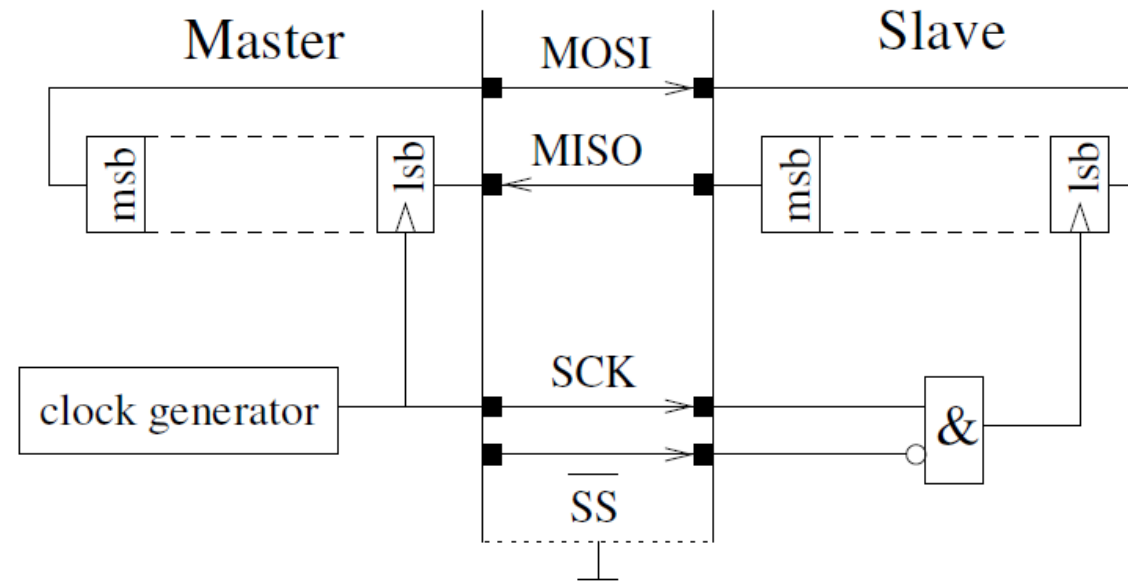
- **Utilizează 4 linii de comunicație:**
 - SCK (System Clock): utilizată de master pentru a transmite semnalul de clock
 - (/SS) (Slave Select): utilizat de master pentru a selecta dispozitivul slave
 - Masterul generează semnalul de ceas și controlează procesul de transfer de date
 - Datele sunt transmise simultan pe liniile MOSI și MISO. Fiecare bit trimis de master este sincronizat cu un bit primit de la slave
 - Transferul de date se face pe baza unor registre de deplasare din dispozitivele conectate
- Avantaje: viteză mare (pana la 10Mbps), ușor de implementat hardware și software, comunicare full-duplex, conectarea mai multor periferice la un singur master
 - Dezavantaje: număr mare de fire de comunicație (4), mecanismul de adresare cu SS, nu e eficient pe distante mari



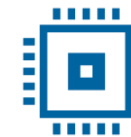
Comunicația SPI – principiu de funcționare



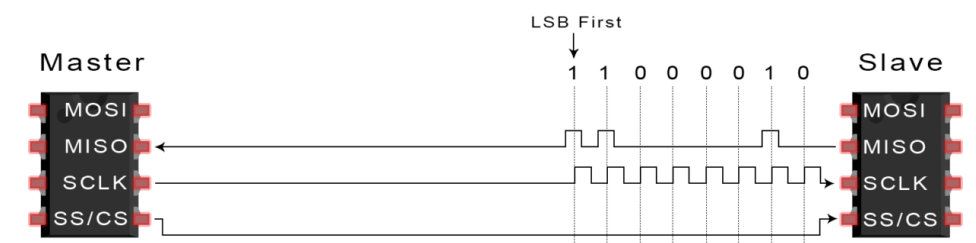
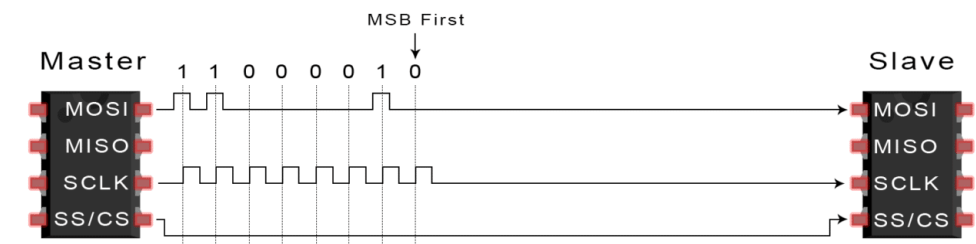
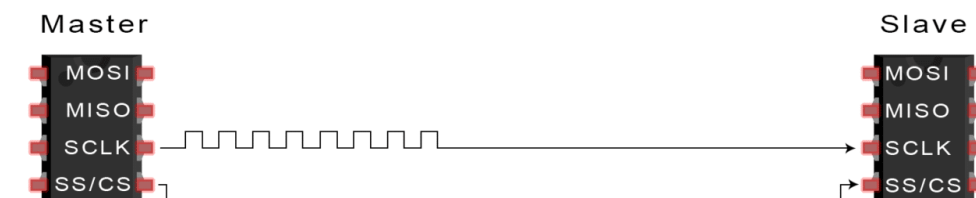
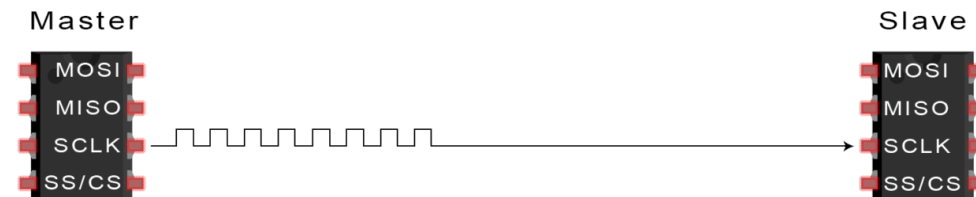
- Master/Slave au un registru cu deplasare (operat de SCK)
- La fiecare clock, master msb (sau lsb) (MOSI) → slave lsb
- În același timp, slave msb (MISO) → master lsb
- După 8 cicluri de clock, master-ul și slave-ul au schimbat între ele datele (8 biți)



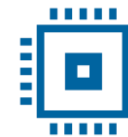
Comunicația SPI – transferul datelor



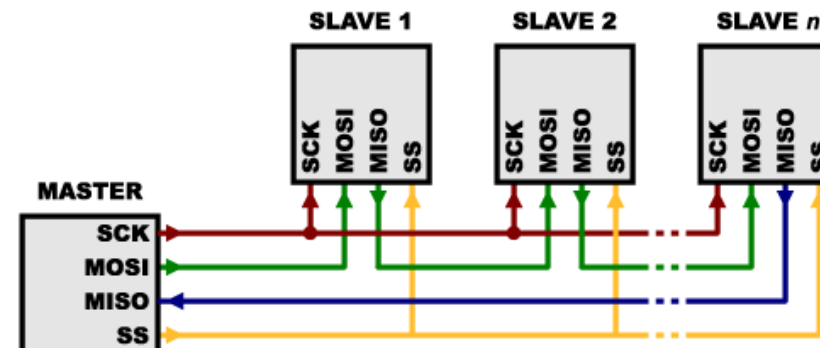
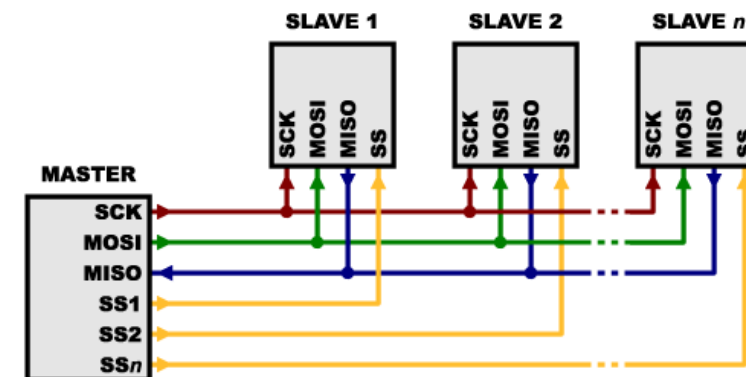
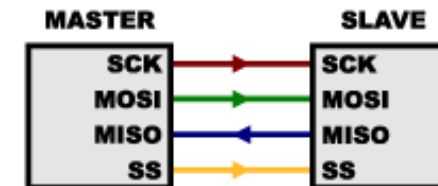
- Masterul generează semnalul de clk
- Masterul trece /SS pe low (activează slave)
- Masterul trimite date, bit după bit, pe MOSI.
- Slave-ul citește datele pe măsură ce le primește
- Dacă un răspuns e necesar , slave-ul trimite date pe MISO
- Masterul citește datele pe măsură ce sunt recepționate



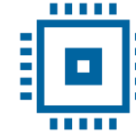
Comunicația SPI – topologii



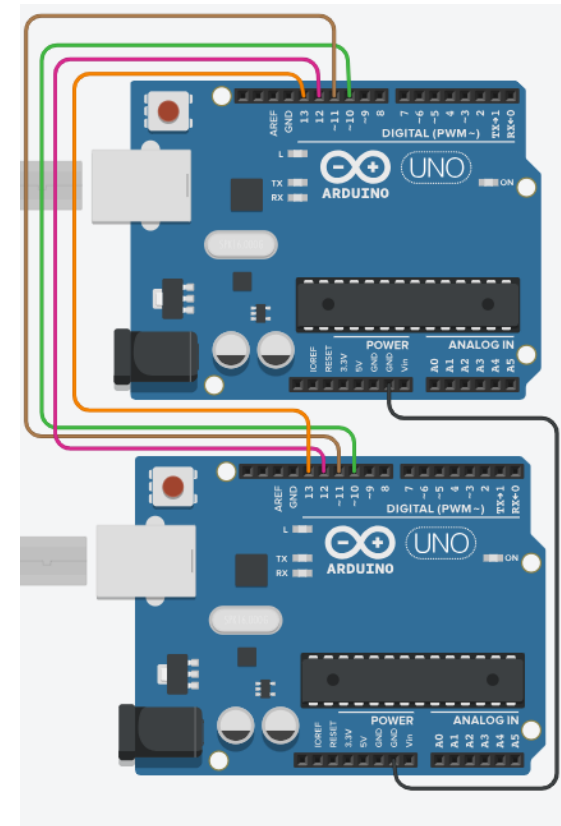
- Master cu un singur slave:
 - Conexiune directă între master și un slave
 - Fiecare linie este partajată exclusiv între cele două dispozitive
- Master cu mai mulți slave (în paralel):
 - Toți slave-ii împart liniile SCLK, MOSI și MISO.
 - Fiecare slave are o linie SS dedicată.
- Lanț:
 - Dispozitivele slave sunt conectate în serie
 - ieșirea unui dispozitiv este conectată la intrarea următorului
 - ex. driver adresă LED



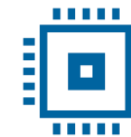
Aplicație SPI



- Se va implementa protocolul SPI pentru transmiterea și recepționarea unui byte de date
- Se va utiliza MC-ul ATmega328P
- Implementare master
 - Inițializare SPI Master
 - Implementare SPI_MasterTransmit
- Implementare Slave:
 - Implementare SPI Slave
 - Implementare SPI_SlaveReceive



Aplicație SPI



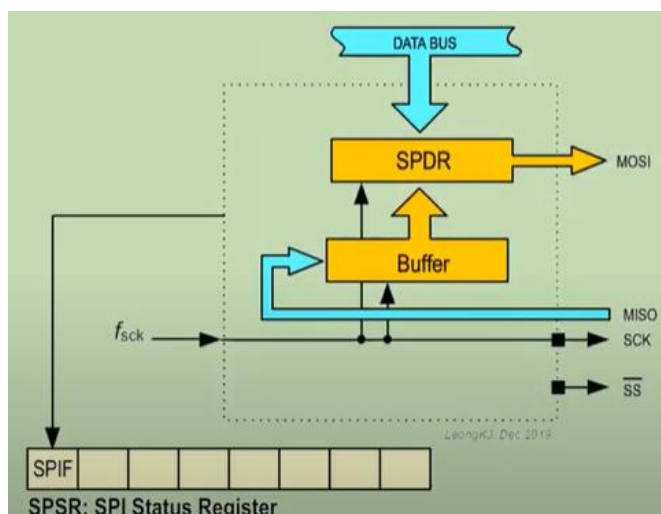
- Implementare master:

- Inițializare SPI

```
//MOSI, SCK și SS definiți ca ieșire
DDRB |= (1<<PB3) | (1<<PB5) | (1<<PB2);
//MISO definit ca pin de intrare
DDRB &= ~(1<<PB4);
//Activare SPI, rata de transfer fsck/16
SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR0);
//Trecere SS pe High
PORTB |= (1<<PB2);
```

- SPI_MasterTransmit

```
//Trecere SS pe Low
PORTB &= ~(1<<PB2);
//Pornire transmitere, cData e un byte
SPDR = cData;
//Așteptare transmitere date
while(!(SPSR & (1<<SPIF))){}
//Trecere SS pe High
PORTB |= (1<<PB2);
```



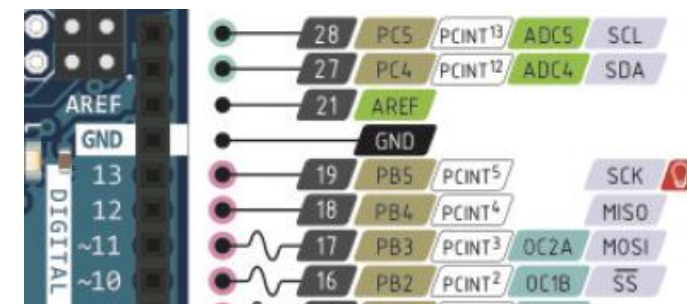
SPCR – SPI Control Register

- Bit 6 – SPE: SPI Enable
- Bit 4 – MSTR: Master/Slave Select (1 Master)
- Bits 1, 0 – SPR1, SPR0: SPI Clock Rate

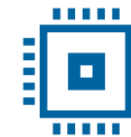
SPSR – SPI Status Register

- Bit 7 – SPIF: SPI Interrupt Flag (When a serial transfer is complete, the SPIF Flag is set)

SPDR – SPI Data Register



Aplicație SPI



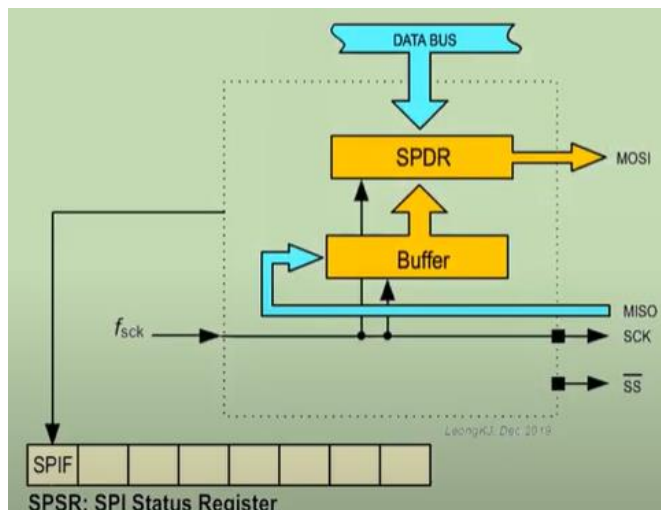
- Implementare master:

- Inițializare SPI

```
//MOSI, SCK și SS definiți ca intrare
DDRB &= ~(1 << PB3) | (1 << PB5) | (1 << PB2));
//MISO definit ca pin de ieșire
DDRB |= (1 << PB4);
//Activare SPI
SPCR = (1 << SPE) | (0 << MSTR)
```

- SPI_SlaveReceive

```
//Așteptare recepție date
while(!(SPSR & (1 << SPIF))) {}
//returnare date primite
return SPDR;
```



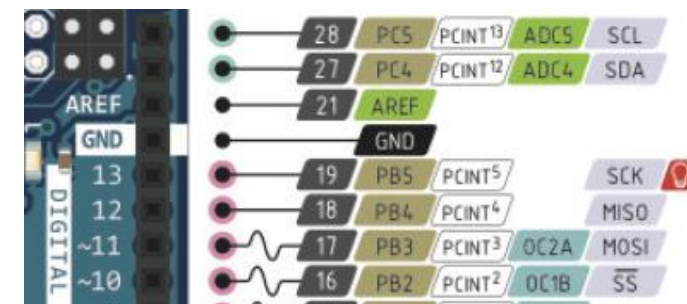
SPCR - SPI Control Register

- Bit 6 - SPE: SPI Enable
- Bit 4 - MSTR: Master/Slave Select (1 Master)
- Bits 1, 0 - SPR1, SPR0: SPI Clock Rate

SPSR - SPI Status Register

- Bit 7 - SPIF: SPI Interrupt Flag (When a serial transfer is complete, the SPIF Flag is set)

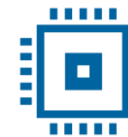
SPDR - SPI Data Register



Concluzii

- Am prezentat mai multe caracteristici ale sistemelor de comunicație și am introdus o varietate de protocoale de comunicație utilizate în interfațarea MC-urilor cu alte dispozitive
- Fiecare protocol are caracteristici specifice, avantaje și limitări, fiind potrivit pentru diferite aplicații:
 - UART (Universal Asynchronous Receiver-Transmitter) este un protocol simplu, utilizat pe scară largă pentru comunicațiile seriale asincrone. Este ușor de implementat și necesită doar două linii: TX (transmit) și RX (receive)
 - USART (Universal Synchronous/Asynchronous Receiver-Transmitter) adaugă suport pentru comunicații sincrone, extinzând astfel flexibilitatea în aplicații care necesită sincronizare precisă
 - SPI este un protocol de comunicație serială sincronă, ideal pentru transferuri rapide de date între microcontrolere și periferice. Folosește un model master-slave și necesită linii dedicate pentru semnalul de ceas (SCK), date (MOSI/MISO) și selecția dispozitivelor (SS).





Contact:

Email: tiberiu.cocias@unitbv.ro

elearning.unitbv.ro – Microcontrollere si Microprocesoare