UNIVERSITATEA DIN
BUCUREȘTI
VIRTUTE ET SAPIENTIA

Facultatea de
Matematică şi Informatică
Universitatea din Bucureşti

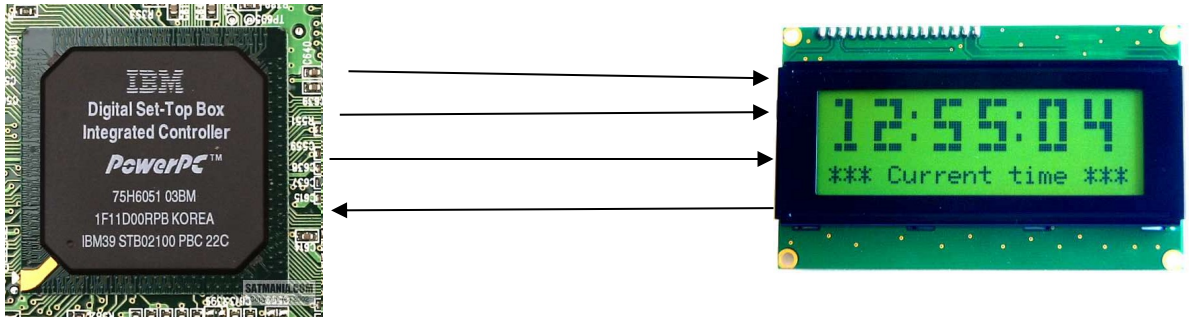
Introducere în Robotică

Cursul 6
SPI & QSPI

Facultatea de Matematică și Informatică
Universitatea București

Ce este SPI?

- Serial Peripheral Interconnect
- Protocol de comunicație serială
- Rapid, simplu, ușor de folosit
- Disponibil pe aproape toate microprocesoarele și microcontrolerele



Serial Peripheral Interconnect (SPI)

- Propus inițial de Motorola
- Four-wire protocol
  - SCLK — Serial Clock
  - MOSI/SIMO — Master Output, Slave Input (PICO – Peripheral In, Controller Out)
  - MISO/SOMI — Master Input, Slave Output (POCI – Peripheral Out, Controller In)
  - SS — Slave Select (CS – Chip Select)
- Single master cu unul sau mai multe dispozitive slave
- Lățime de bandă mai mare ca I2C, poate să facă “stream transfers”
- Nu este nevoie de arbitrare
- Dar
  - Necesită mai mulți pini de date
  - Nu are flow control hardware
  - No slave acknowledgment (master poate să vorbească în gol și să nu își dea seama)

SPI Basics

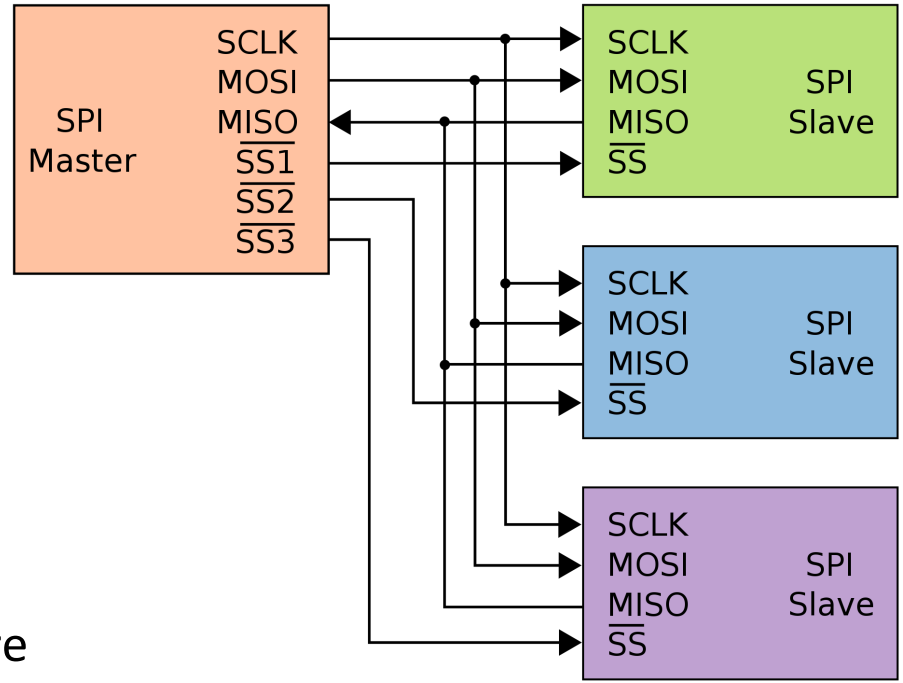
- Folosește patru linii de comunicație
  - De aceea este cunoscut și ca *4-wire bus*
- Folosit pentru comunicația peste distanțe mici (pe același PCB)
- Multiple Slaves, Single Master
- Protocol sincron

Capabilități SPI

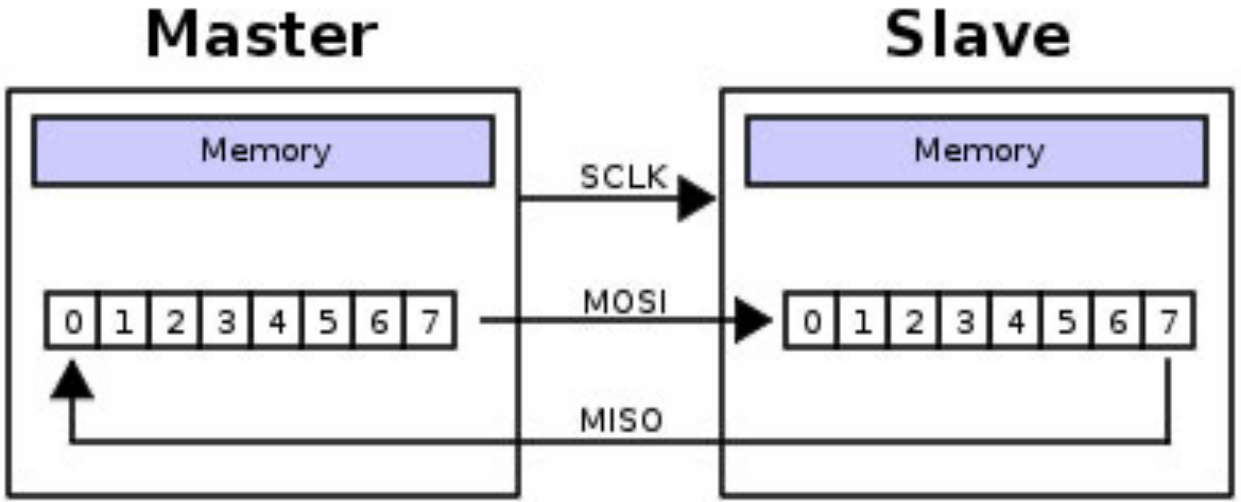
- Întotdeauna full-duplex
  - Comunică în două direcții simultan
  - Transmisia nu trebuie să aibă sens (în ambele sensuri)
- Viteze de transmisie de zeci sau chiar sute Mbps
- Transferuri de date folosind pachete de la 4 la 16 biți
- Mai multe dispozitive slave
  - Posibil să fie în configurație *daisy-chain*

Protocolul SPI

- Linii date:
  - Master Out Slave In (MOSI)
  - Master In Slave Out (MISO)
  - System Clock (SCLK)
  - Slave Select 1…N
- Master setează Slave Select pe zero
- Master generează semnalul de ceas
- MISO – MOSI legate la registre shiftare



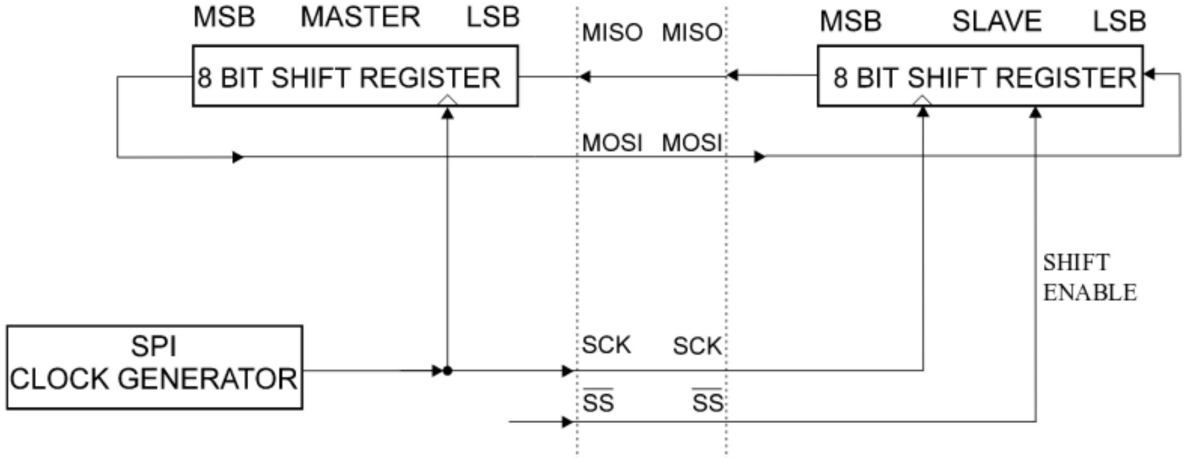
SPI folosește un model Shift Register



Master shifts out data to Slave, and shifts in data from Slave

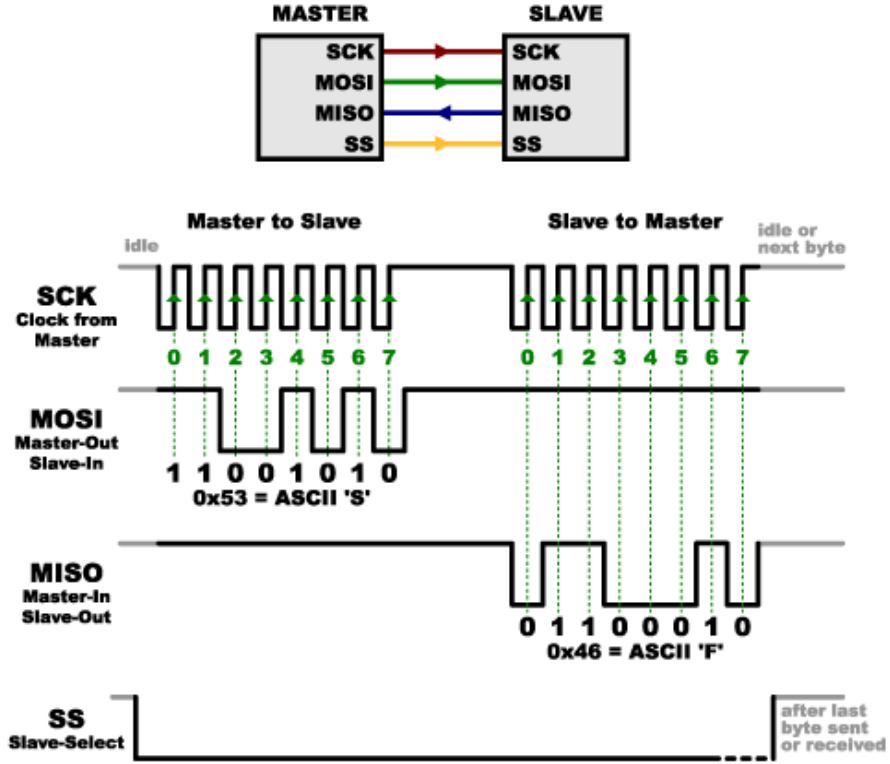
http://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/SPI_8-bit_circular_transfer.svg/400px-SPI_8-bit_circular_transfer.svg.png

Comunicația SPI



- MSB de la Master devine LSB la Slave
- Se formează un barrel shifter pe 16 biți
  - 8 biți la Master, 8 biți la Slave

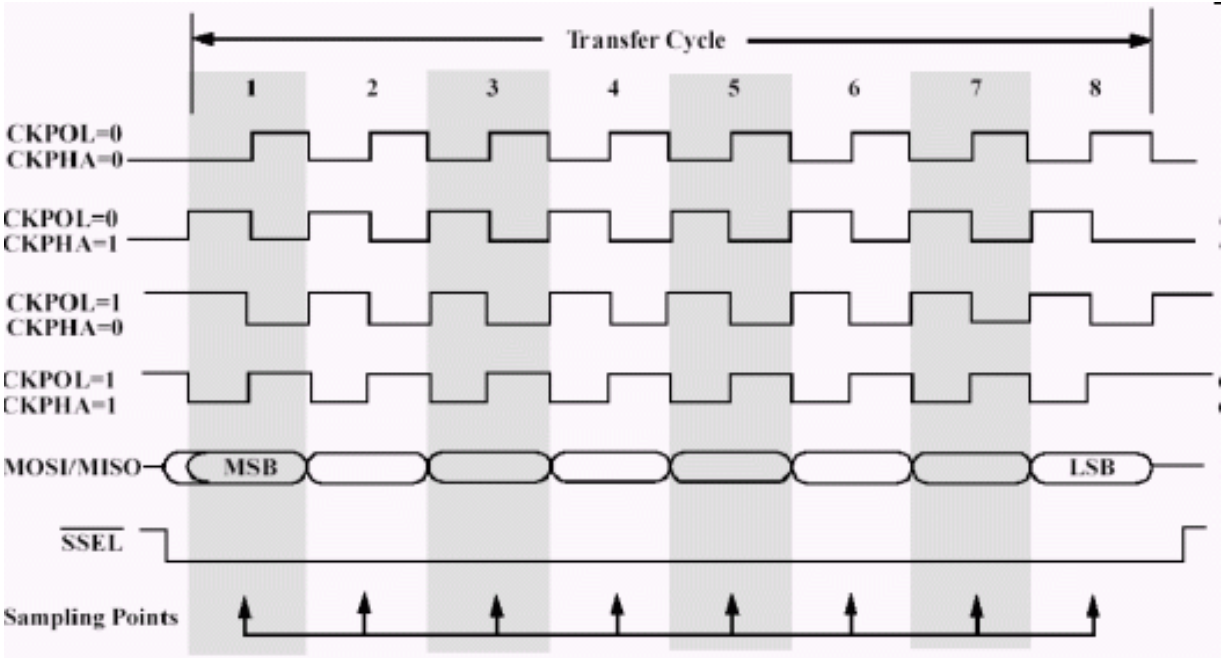
Comunicația SPI



SPI clocking: nu există un “mod standard”

- Patru moduri de a furniza semnal de ceas
  - Două faze
  - Două polarități
- Master și Slave-ul *selectat* trebuie să fie în același mod
- În timpul unui transfer cu Slave A și B ce au moduri diferite de clocking, un Master trebuie să:
  - Configureze ceasul la modul potrivit pentru A
  - Chip Select Slave A
  - Efectuează transferul
  - Deselect Slave A
  - Configureze ceasul la modul potrivit pentru B
  - Chip Select Slave B
  - Efectuează transferul
  - Deselect Slave B
- Master reconfigurează clock mode dinamic!

SPI timing diagram

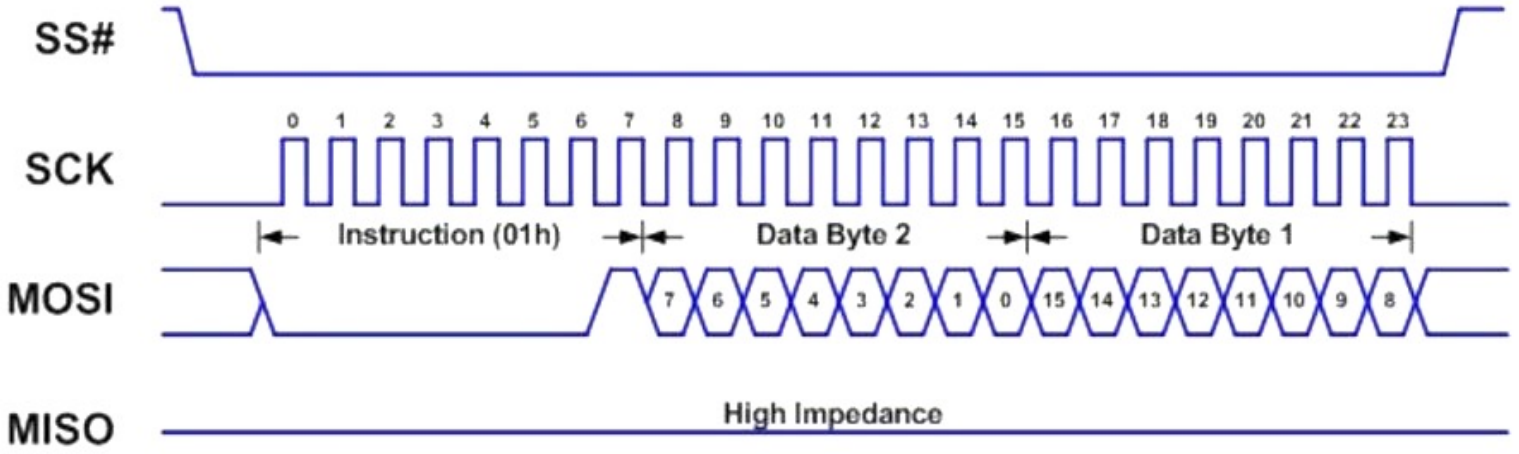


Timing Diagram – Showing Clock polarities and phases

http://www.maxim-ic.com.cn/images/appnotes/3078/3078Fig02.gif

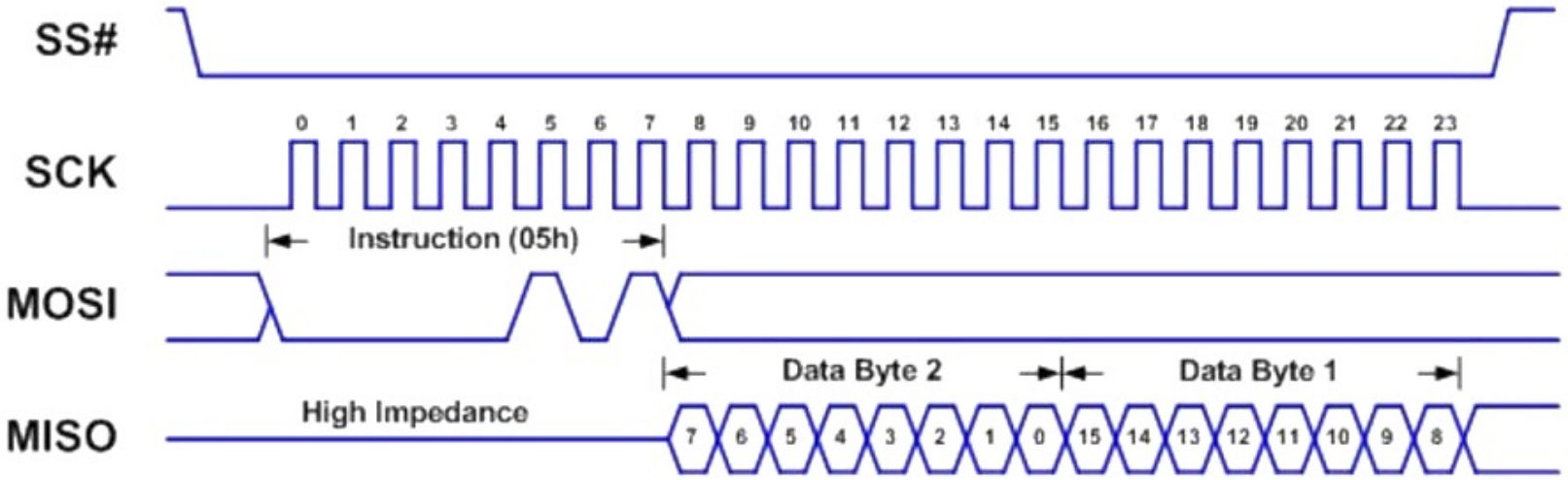
Scrierea SPI

- Majoritatea perifericelor SPI (memorii, senzori) acceptă o secvență de scrieri
  - Structura este de obicei: <comandă>, <date>
- Pentru a face o scriere, SPI Master trece linia de CS/SS pe zero logic, generează semnalul de ceas, apoi trimite informația bit cu bit pe linia MOSI.
- Pentru că este o scriere, nu este nevoie de un răspuns din partea Slave, deci linia MISO este în HiZ



Citirea SPI

- Procedura pentru o citire este similară cu scrierea, cu două diferențe notabile
  - Master trimite doar o comandă, de obicei
  - Slave răspunde pe MISO cu unul sau mai mulți octeți de date, în funcție de comandă

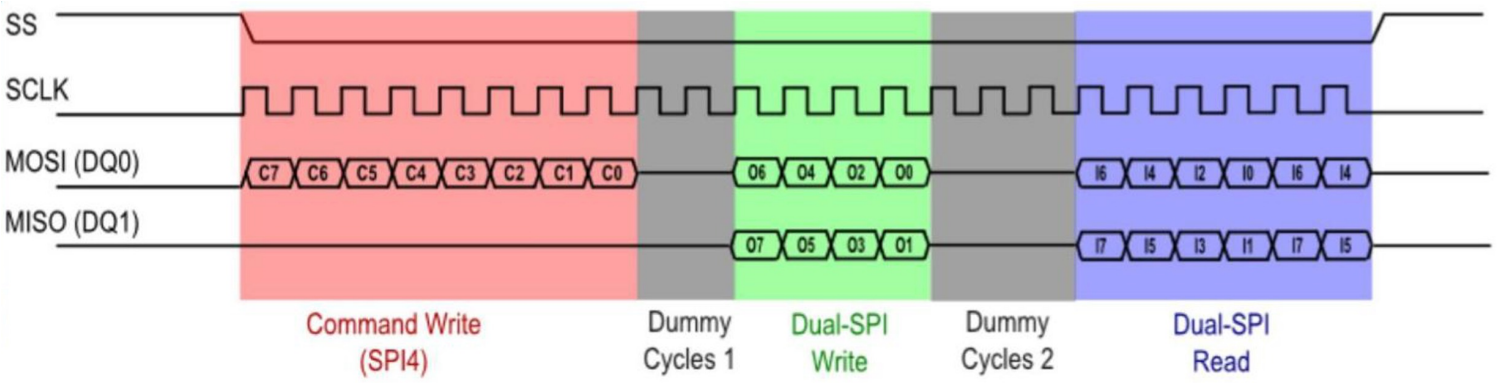


Extensii SPI

- Pentru a obține o lățime și mai mare de bandă, folosim mai multe linii de date
  - Obținem Dual, Quad sau chiar Octo-SPI
  - 2, 4 sau 8 linii de date bidirecționale

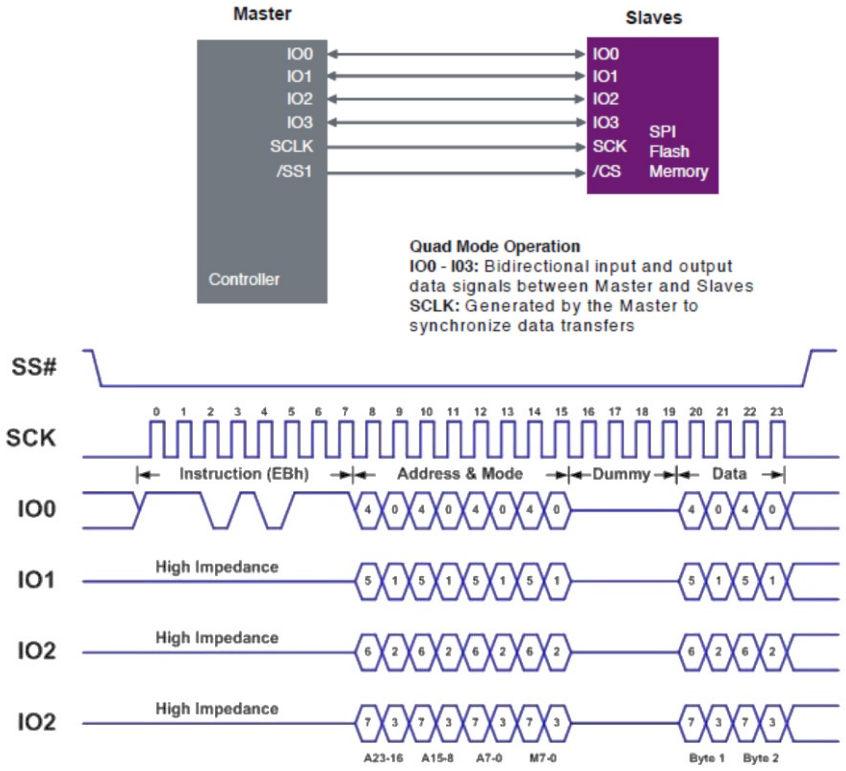
Dual SPI

- Extensie a SPI care permite transferul de date bidirecțional pe două linii
- Posibil să refolosească liniile MISO/MOSI pentru transferul simultan Master-Slave și invers



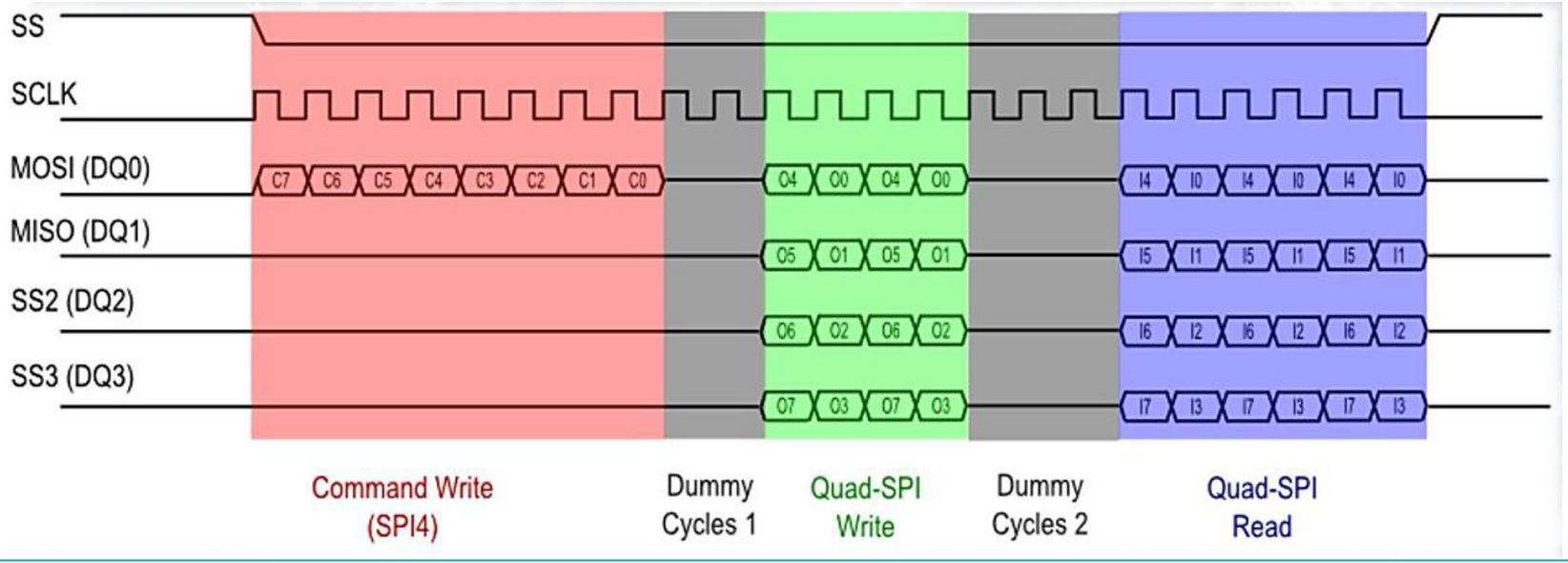
Quad-SPI

- Componentele care folosesc acest mod pot să ofere viteze de transfer serial similare cu un transfer paralel de date. Pentru memorii Flash, QSPI este foarte folosit pentru a accesa și executa programul direct din memorie (execute-in-place)
- Exemplu interfațare memorie Flash QSPI:
  - Un transfer începe prin transmiterea unei instrucțiuni (citire sau scriere) pe linia IO0
  - Adresa (32 de biți) este transmisă în paralel de host pe toate cele 4 linii de date
  - Urmează un timp de așteptare (dummy bits), pentru ca memoria să proceseze intern cererea
  - Datele (pentru scriere sau citite) vor fi transmise/recepționate în paralel pe toate cele 4 linii de date



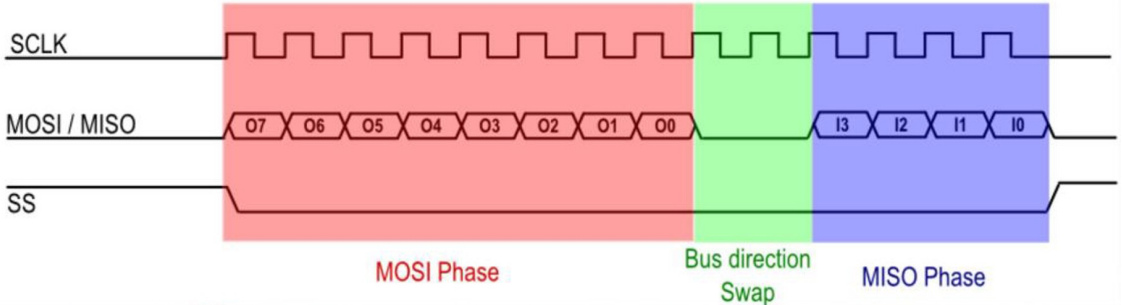
Quad-SPI

Exemplu de citire și scriere QSPI



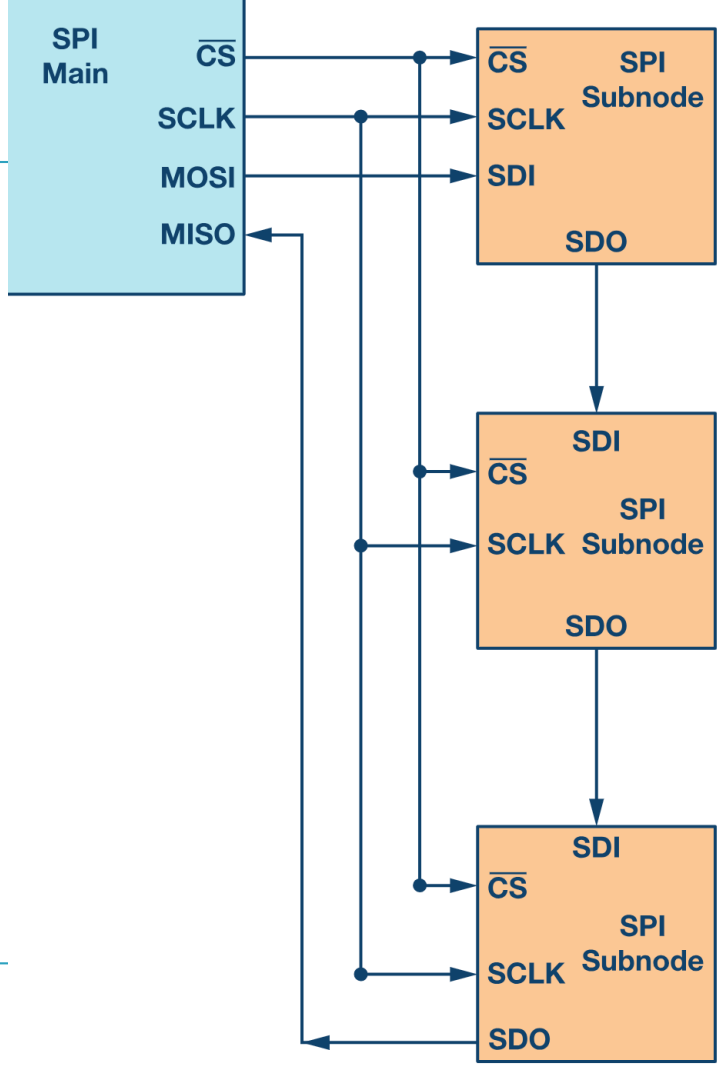
3-Wire SPI

- Există un standard ce folosește doar trei linii de date pentru comunicație SPI: CS, SCK și o singură linie bidirecțională pentru transferul de date (MISO&MOSI contopit)
- Sunt totuși diferențe față de SPI standard, de aceea un dispozitiv trebuie să fie compatibil cu standardul 3-wire
- Exemplu: DS1306 RTC (Real-Time Clock) are suport intern pentru SPI dar și pentru 3-Wire
- Comunicația este half-duplex!



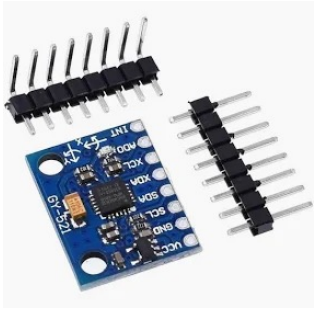
SPI Daisy Chaining

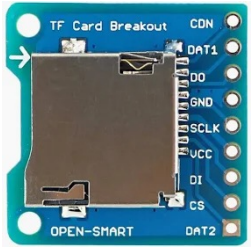
- Tehnică de conectare a mai multor dispozitive Slave prin înșiruire
- MISO de la un Slave este legat la MOSI de la următorul Slave
- Avantaj: toate cipurile Slave au același Chip Select – sunt tratate ca același Slave
- Dezavantaj: un transfer durează mai mult (trebuie ca biții să fie shiftați prin mai mulți Slave până la destinație)

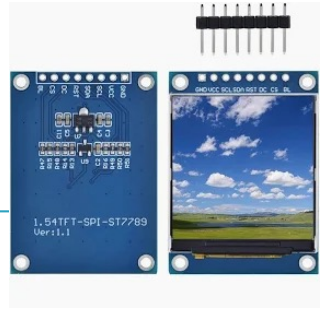


Ce periferice au SPI?

- Senzori de mișcare
  - Accelerometre, magnetometre, giroscoape
- Senzori ambientali
  - Temperatură, presiune, umiditate etc.
- Memorii
  - Carduri SD/MMC, memorii externe Flash
- Display-uri LCD și OLED
- Multe alte periferice!

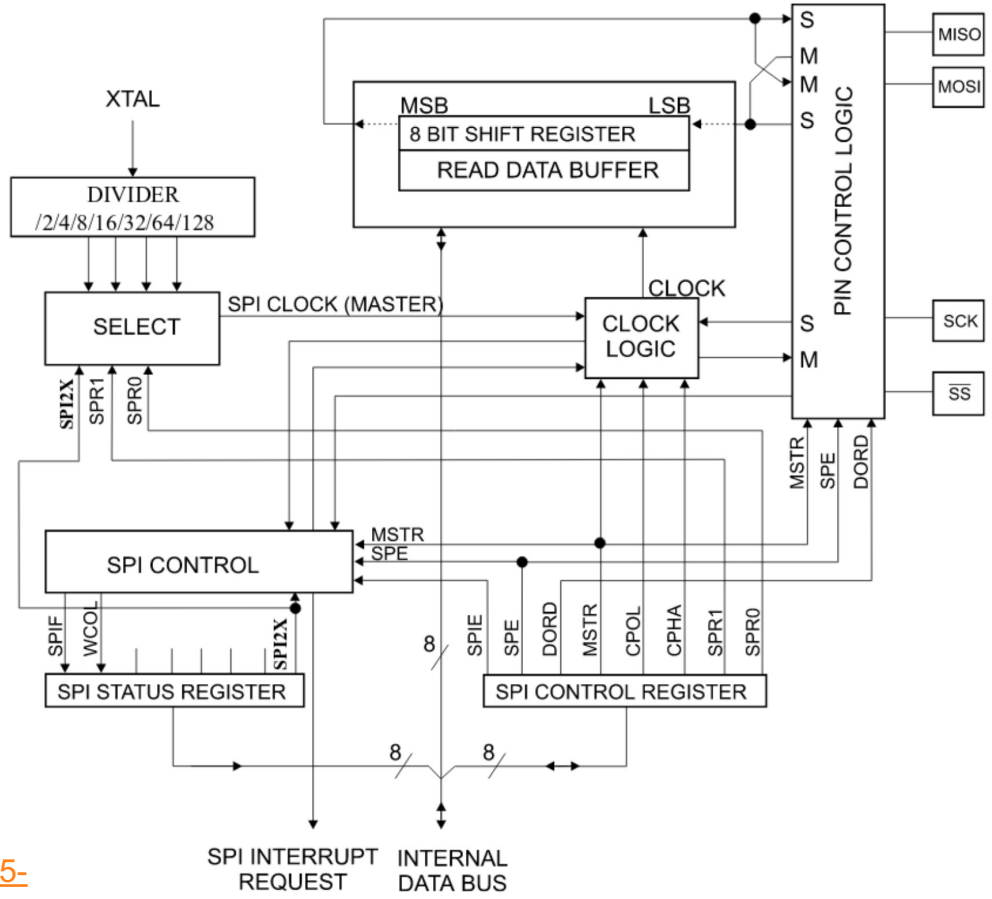






SPI la ATMega324P

- Interfața poate funcționa atât în mod Master cât și Slave
- Prescaler selectabil la transmisie
- Funcționare prin polling sau întreruperi



https://ww1.microchip.com/downloads/en/Appnotes/Atmel-2585-Setup-and-Use-of-the-SPI_ApplicationNote_AVR151.pdf

SPI - Registre

- SPCR
  - SPE – SPI enable
  - SPIE – interrupt enable
  - DORD – data order (LSB sau MSB-first)
  - CPOL, CPHA – clock polarity & phase
  - SPR + SPI2X - setare prescaler
- SPSR
  - SPIF – se pune pe 1 la fiecare transfer de date
- SPDR
  - Data register, buffer intrare-ieșire

SPCR – SPI Control Register

Bit	7	6	5	4	3	2	1	0	
0x2C (0x4C)	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
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	

SPSR – SPI Status Register

Bit	7	6	5	4	3	2	1	0	
0x2D (0x4D)	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

SPDR – SPI Data Register

Bit	7	6	5	4	3	2	1	0	
0x2E (0x4E)	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

Pinul SS la ATMega324p

- Pinul SS este folosit pentru a inițializa sau a termina un transfer de date
- Atunci când microcontrolerul este în modul Master, pinul poate fi declarat la latitudinea celui care scrie cod
  - Puteți folosi orice alt pin GPIO pentru SS în Master mode
- În modul Slave pinul este automat declarat ca input și trebuie activată rezistența de pull-up (sau pus pull-up extern)
  - Dacă un Master extern setează pinul pe 0 logic, bitul SPIF din SPSR va fi automat setat la 1
- Atenție: dacă microcontrolerul este setat în mod Master iar pinul de SS este declarat la input și un periferic extern scrie un 0 logic pe pin, microcontrolerul va trece automat în modul Slave (setează la 0 bitul MSTR din SPCR)!

Pinul SS la ATMega324p

- Este foarte indicat să faceți pinul SS output atunci când lucrați cu SPI în modul Master, ca să nu fiți întrerupți din funcționare
- Atunci când lucrați cu SPI în modul Slave, pinul SS este întotdeauna intrare și nu îl puteți controla din software. Trebuie să fie ținut extern pe 0 logic pentru a activa SPI. Când SS este pe 1 logic, SPI este dezactivat

Exemplu Master

SPI în mod Master

Transmisie cu busy waiting

```c
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#define DD_MOSI 5
#define DD_MISO 6
#define DD_SCK 7
#define DDR_SPI DDRB

void SPI_MasterInit(void)
{
    // Set MOSI and SCK output, all others input
    DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
    // Enable SPI, Master, set clock rate fck/16
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}


void SPI_MasterTransmit(char cData)
{

    SPDR = cData; // Start transmission
    while(!(SPSR & (1<<SPIF))); // Wait for transmission complete
}

int main()
{

    SPI_MasterInit();
    SPI_MasterTransmit(0xFF);

    return 0;
}
```

Exemplu Slave

Interfața configurată în mod Slave

Recepție cu busy-waiting

```c
#include <avr/io.h>

#define DD_MOSI 5
#define DD_MISO 6
#define DD_SCK 7
#define DDR_SPI DDRB

void SPI_SlaveInit(void)
{
    // Set MISO output, all others input
    DDR_SPI = (1<<DD_MISO);
    // Enable SPI
    SPCR = (1<<SPE);
}

char SPI_SlaveReceive(void)
{
    // Wait for reception complete
    // SPI Status Reg & 1<<SPI Interrupt Flag
    while(!(SPSR & (1<<SPIF)));
    // Return data register
    return SPDR;
}

int main()
{
    SPI_SlaveInit();

    while(1)
    {
        PORTC=SPI_SlaveReceive();
    }
    return 0;
}
```

Exemplu întreruperi

Activăm întreruperea și adăugăm octeții recepționați într-un buffer

Aceeași întrerupere pentru transmisie – se poate genera o transmisie la îndeplinirea unei condiții

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define DD_MOSI 5
#define DD_MISO 6
#define DD_SCK 7
#define DDR_SPI DDRB

uint8_t data[10];
uint8_t x = 0;


void SPI_SlaveInterruptInit(void) // For Interrupted SPI
{
    DDR_SPI = (1<<DD_MISO);
    // set SPI enable, clock polarity, interrupts enable
    SPCR = (1<<SPE)|(1<<CPOL)|(1<<SPIE);
}

ISR (SPI_STC_vect)                  // SPI interrupts
{
    data[x++] = SPDR; // receive and store byte
    x %= 10;          // when 10 = reset
}


int main()
{

    SPI_SlaveInterruptInit();

    sei();  //Enable interrupts

    while(1)
    {
        ...
    }
    return 0;
}
```

SPI – Exemple Cod

- Funcție transfer full-duplex:

```
uint8_t spi_transfer(uint8_t data)
{
    SPDR = data;
    loop_until_bit_is_set(SPSR, SPIF);
    return SPDR;
}
```

- Citire SPI:

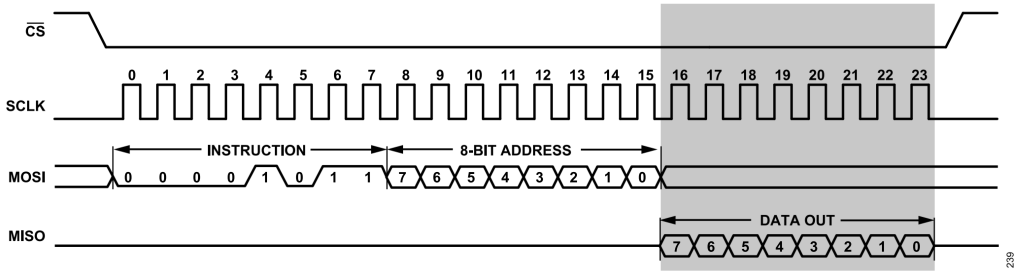
```
uint8_t spi_read()
{
    return spi_transfer(0xFF);
}
```

- Scriere SPI:

```
uint8_t spi_write(uint8_t data)
{
    spi_transfer(data);
}
```

Exemplu: accelerometrul ADXL362

Este imposibil ca accelerometrul să răspundă instantaneu la o cerere!



Vrem să citim de la adresa 0x08 din memoria accelerometrului

```
spi_write(0x0B);
uint8_t data = spi_transfer(0x08); // greșit!
```

Trebuie să așteptăm ca perifericul să aibă timp să recepționeze instrucțiunea și adresa, ca mai apoi să răspundă

```
spi_write(0x0B);
spi_write(0x08);
uint8_t data = spi_read(); // corect!
```

https://www.analog.com/media/en/technical-documentation/data-sheets/adxl362.pdf

SPI Pros and Cons

- Pros:
  - Rapid și ușor de folosit
    - Conexiuni punct la punct rapide
    - Permite streaming, flux constant de date
    - Fără adresare, ușor de implementat
  - Toată lumea îl folosește

- Cons:
  - Devine dificil dacă avem multe dispozitive Slave (multe semnale CS)
  - Nu există o metodă de a trimite un ACK
  - Nu are arbitrare
  - Nu are flow control