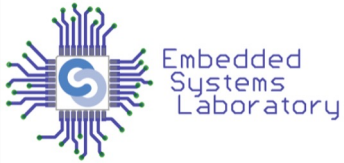


PROIECTAREA CU MICROPROCESOARE

Cursul 7

I^2C, I^3C, I^2S

Facultatea de Automatică și Calculatoare
Politehnica București

Ce este I²C?

- Inter-Integrated Circuit
- Protocol inventat de Philips, devenit standard pentru o largă gamă de circuite integrate
  - Senzori
  - Memorii
  - Display-uri
  - ...
- I2C este foarte potrivit pentru comunicația de date la distanțe mici (pe PCB) la viteze de la kHz până la câțiva MHz
- Protocolul permite crearea unei rețele de circuite integrate care comunică folosind aceeași linie partajată de date

Caracteristici

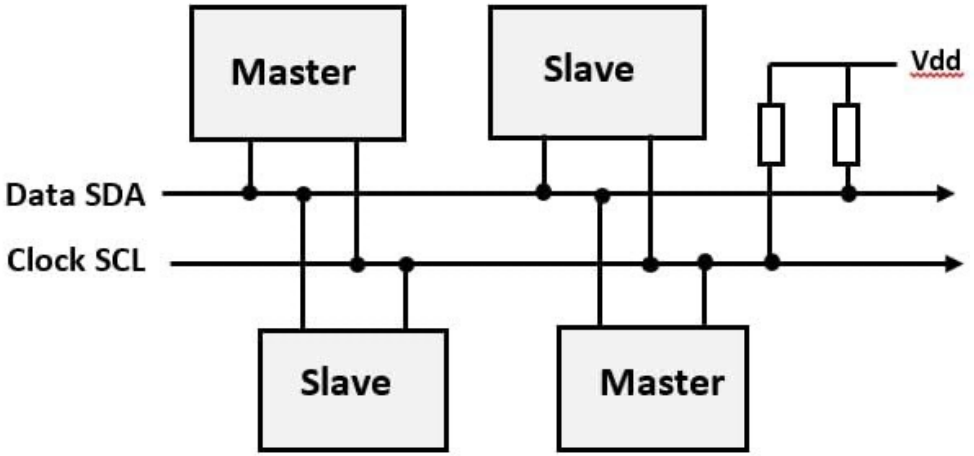
- Multi-Master, Multi-Slave
- Protocol sincron de date
  - O linie dedicată pentru ceas (SCL)
- Half-duplex
  - O singură linie pentru date (SDA)
- Se mai numește și Two-Wire Interface
  - Cele două linii sunt trase sus de rezistențe de pull-up
- Numărul redus de linii de comunicație scade pin count și consumul de energie

Specificații electrice

- Cele două linii sunt open-drain
  - De obicei este nevoie de 2 rezistențe de 4.7kOhmi pentru pull-up
- Asta înseamnă că liniile de I^2C sunt implicit ținute pe 1 logic, iar dacă unul sau mai multe dispozitive conectate la linie vor să comunice, trebuie să tragă linia la 0 logic
- Sunt trei stări pe linia de date I^2C:
  - Idle (1 logic)
  - Master transmit mode
  - Master receive mode

Arhitectura I^2C

- Două linii pentru a conecta dispozitivele
  - Serial data line (SDA)
  - Serial clock line (SCL)
- Adrese pe 7 biți
  - Maxim 128 entități

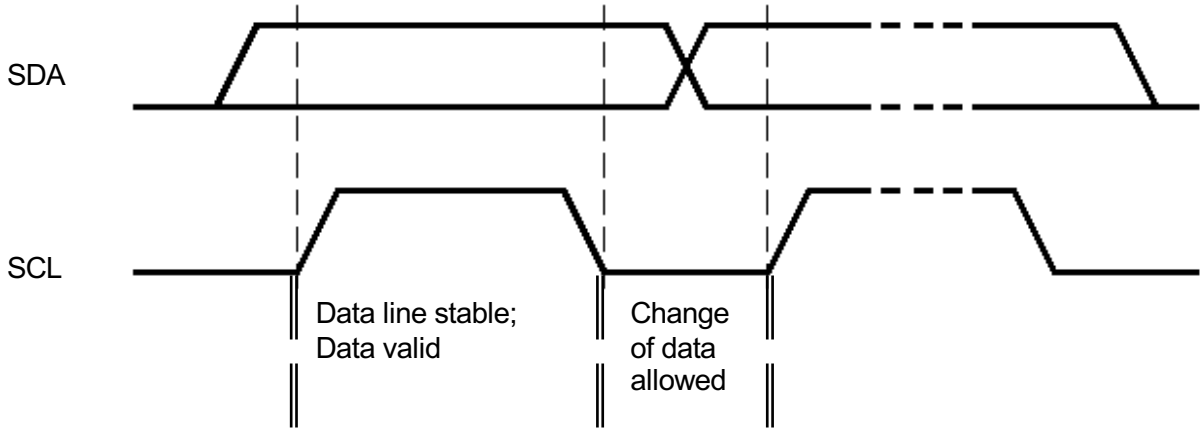


Detalii I^2C

- Fiecare dispozitiv are o adresă unică pe bus
- Fiecare dispozitiv poate fi un emițător sau un receptor
- Dispozitivele I^2C pot fi master sau slave pentru un transfer de date
  - Master, de obicei un microcontroller, inițializează transferul pe magistrală și generează semnalul de ceas care permite transferul de date de la și la Slave
  - Slave este orice device adresat de Master la un moment dat

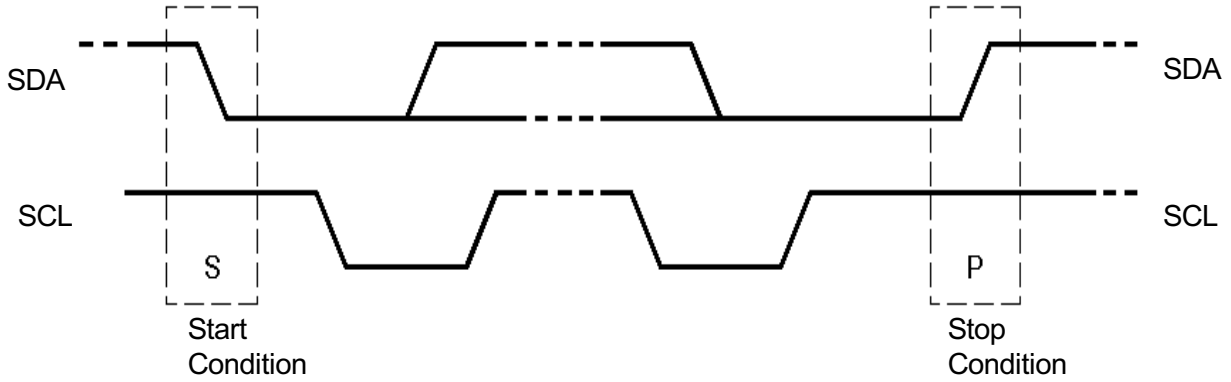
Transferul de date pe I^2C

- La un transfer normal de date, linia de date își schimbă starea doar când ceasul este pe 0 logic



Condițiile de Start și Stop

- O tranziție pe linia de date cât timp ceasul este pe 1 logic este definită ca o condiție de Start (pe negedge) sau de Stop (pe posedge)
- Ambele condiții pot fi generate doar de Master
- Linia de date este rezervată între un semnal de Start și unul de Stop



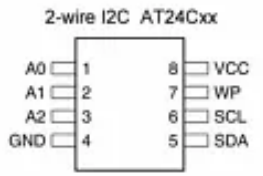
Adresare

Fiecare nod are o adresă unică pe 7 biți

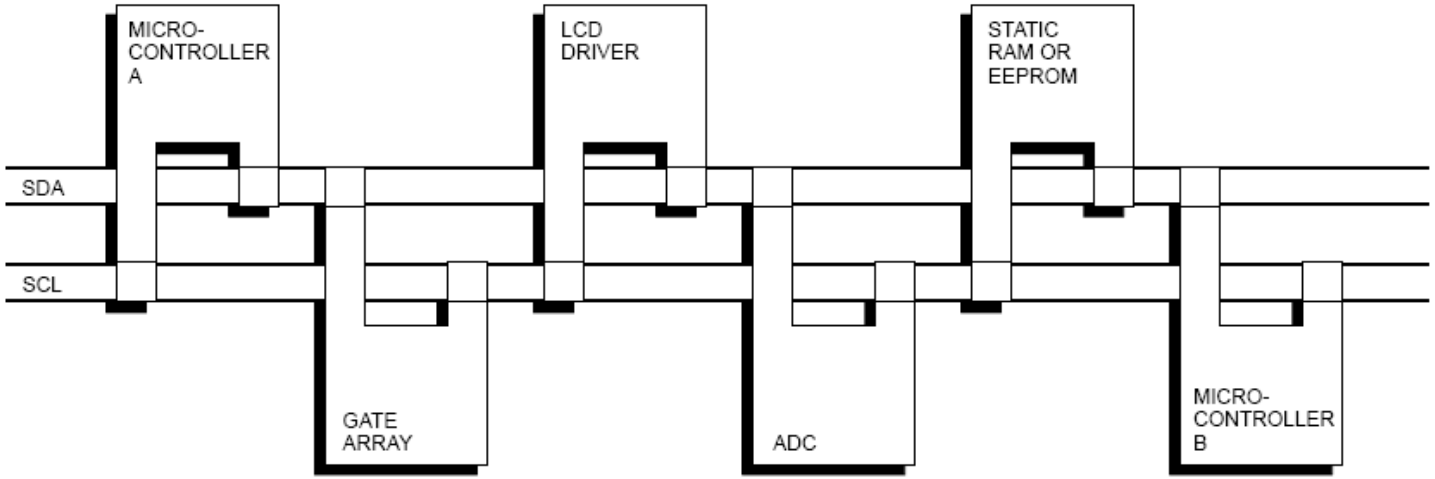
Perifericele au de obicei adrese fixe, sau care pot fi schimbate pentru ultimii biți

Adresele care încep cu 0000 sau 1111 au roluri speciale

- 0000000 este adresă de General Call
- 0000001 este adresă Null
- 1111xxx Address Extension
- 1111111 Address Extension – următorii biți sunt adresa efectivă



I²C-Connected System



Example I2C-connected system with two microcontrollers

(Source: I2C Specification, Philips)

Relațiile Master-Slave

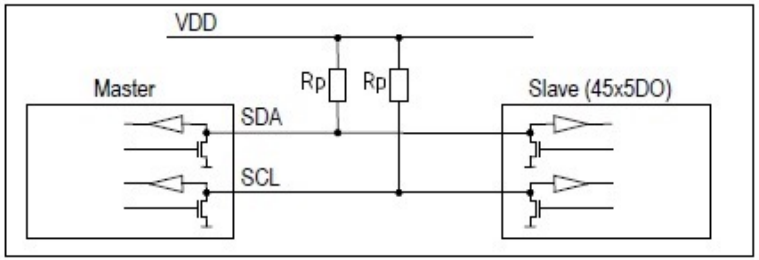
- Cine este Master?
  - master-transmitters
  - master-receivers

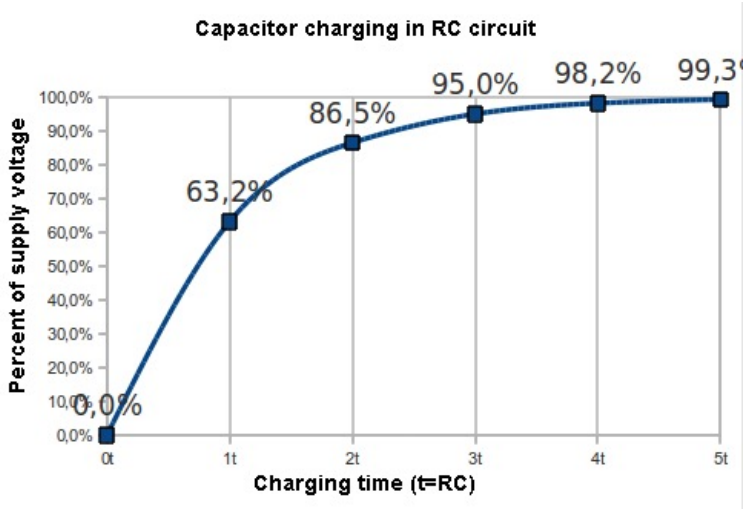
- Să presupunem că microcontrolerul A vrea să trimită informație la microcontroler B
  - A (master) adresează B (slave)
  - A (master-transmitter), trimite date la B (slave-receiver)
  - A termină transferul.

- Dacă microcontrolerul A vrea să recepționeze informație de la microcontroler B
  - A (master) adresează microcontroler B (slave)
  - A (master-receiver) primește date de la B (slave-transmitter)
  - A termină the transfer

- În ambele cazuri Master-ul generază semnalul de ceas și inițializează/termină transferul

Exercițiu: Cât de repede poate să transfere I²C datele?





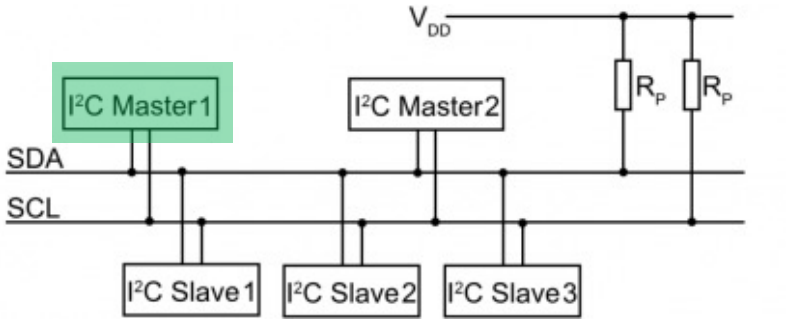
- Cât de repede poate să meargă?
- Presupuneri
  - 0 este acționat
  - 1 este tras sus de rezistențe
- Câteva date numerice
  - R_p = 10 kΩ
  - C_{cap} = 100 pF
  - V_{DD} = 5 V
  - V_{in_high} = 3.5 V
- Pentru un circuit RC
  - $V_{cap}(t) = V_{DD}(1-e^{-t/\tau})$
  - Unde $\tau = RC$

Exercițiu: Bus bit rate vs Useful data rate

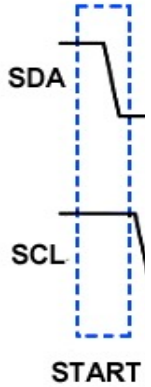
- O “tranzacție” I2C presupune următoarele transferuri de biți
  - <START><A6:A0><R/W><ACK><D7:D0><ACK><STOP>
- Care din cele de mai sus conține date utile?
- Deci, dacă I^2C merge la 400kHz
  - Care este perioada ceasului?
  - Care este data throughput (i.e. data-bits/second)?
  - Care este bus “efficiency”?

Cum arată un transfer I²C?

- Master 1 trimite condiția de Start (S) și preia controlul semnalului de ceas (setează SCL high și SDA low)

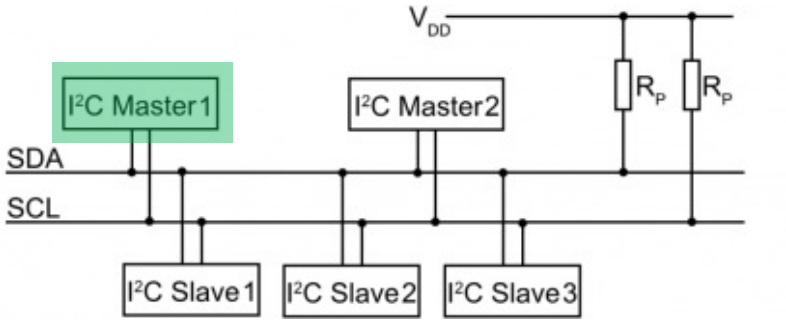


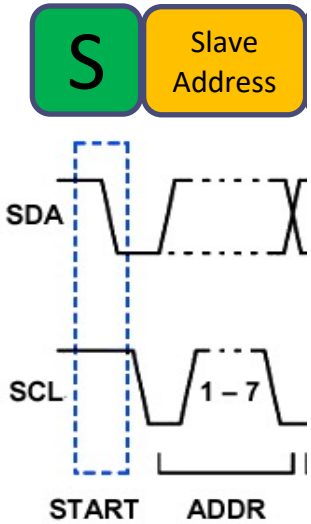
S



Cum arată un transfer I²C?

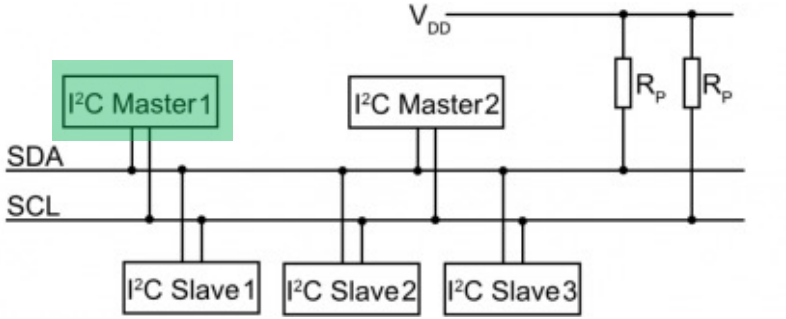
- Master 1 trimite o adresă unică pe 7 biți

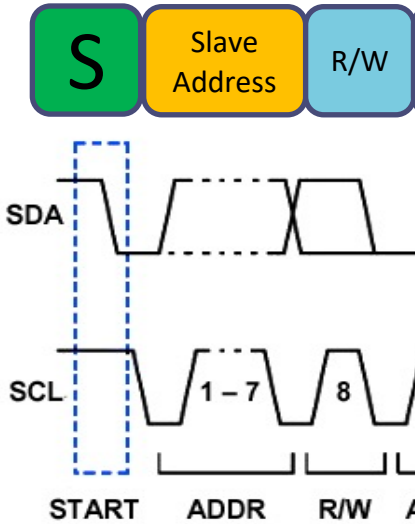




Cum arată un transfer I²C?

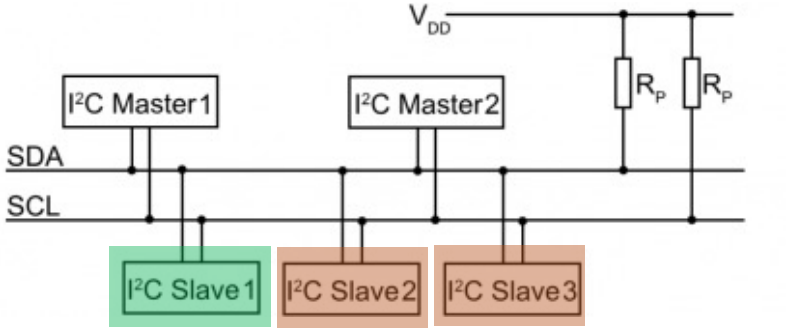
- Master 1 trimite un bit care codifică operația pe bus
  - 0 înseamnă Write (slave receive)
  - 1 înseamnă Read (slave transmit)

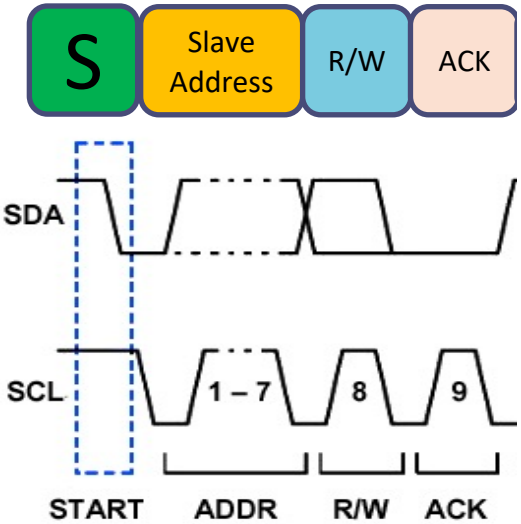




Cum arată un transfer I²C?

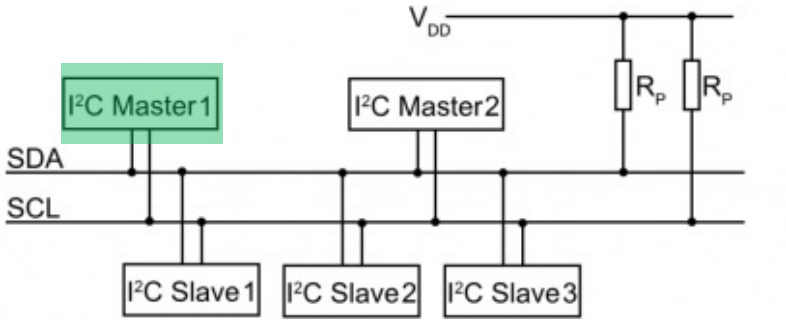
- Slave 1 trimite un bit de Acknowledge
- Toți ceilalți Slave vor ignora transferurile de date pe magistrală până la următoare condiție de Stop

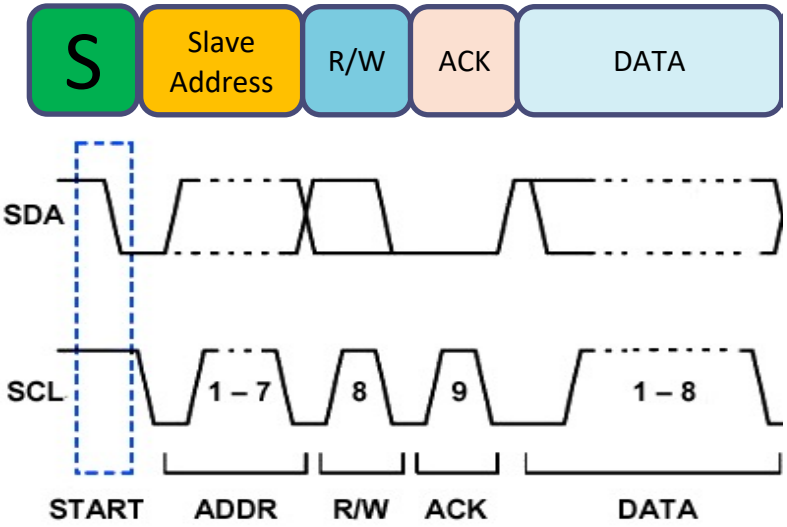




Cum arată un transfer I²C?

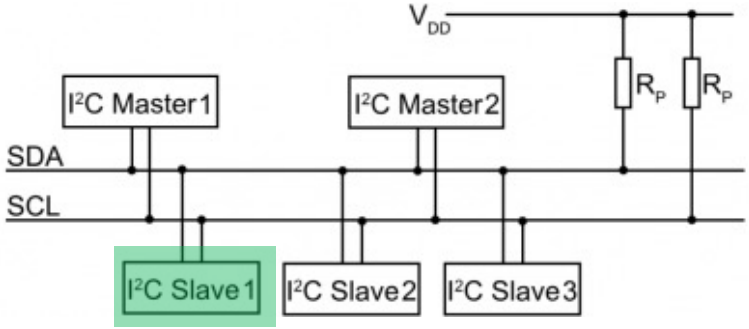
- Transmițătorul (Slave sau Master) trimite un octet de date
- Aici presupunem că Master trimite datele

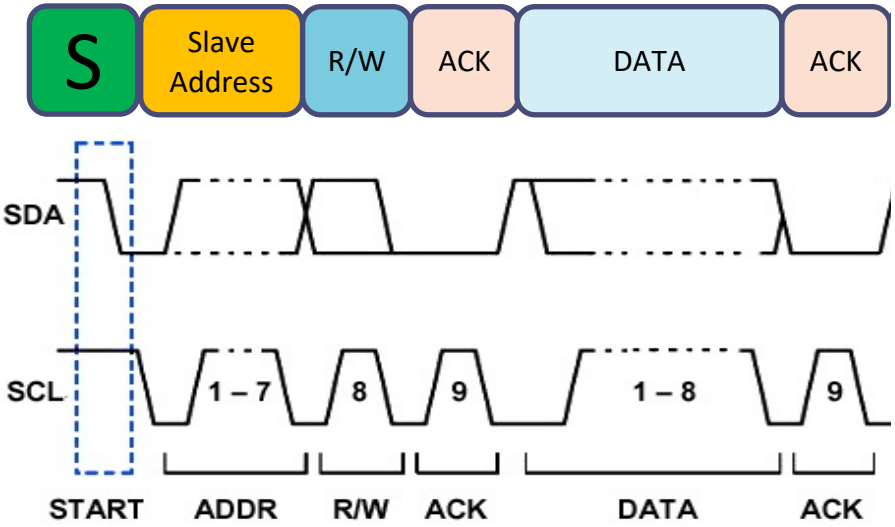




Cum arată un transfer I²C?

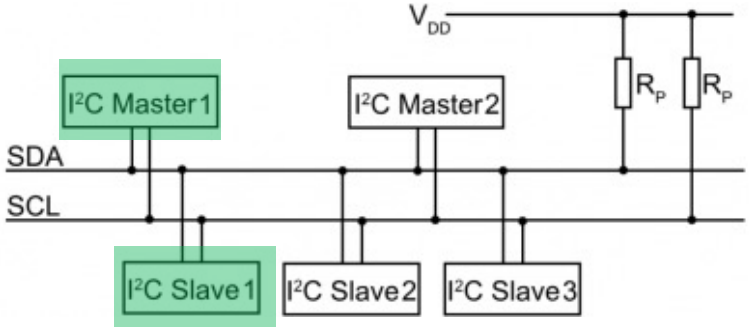
- Slave răspunde cu un ACK la recepționare corectă

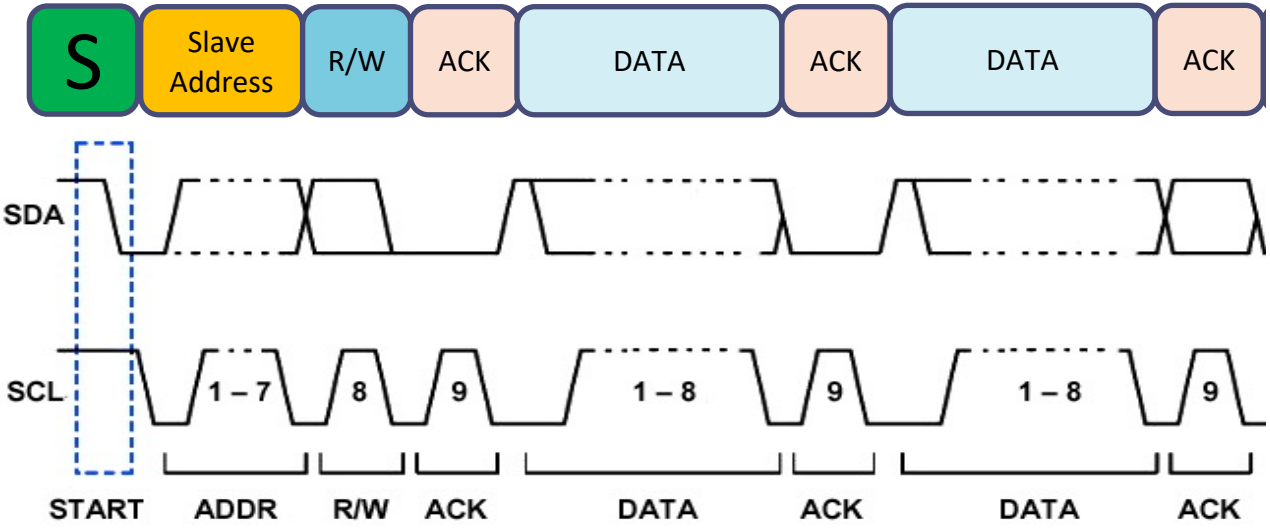




Cum arată un transfer I²C?

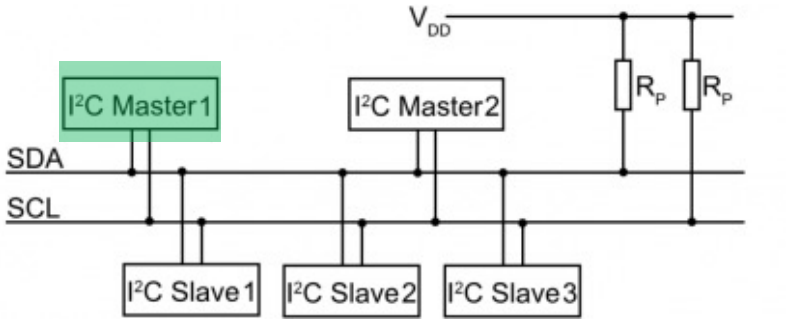
- Ciclul se repetă de oricâte ori este nevoie

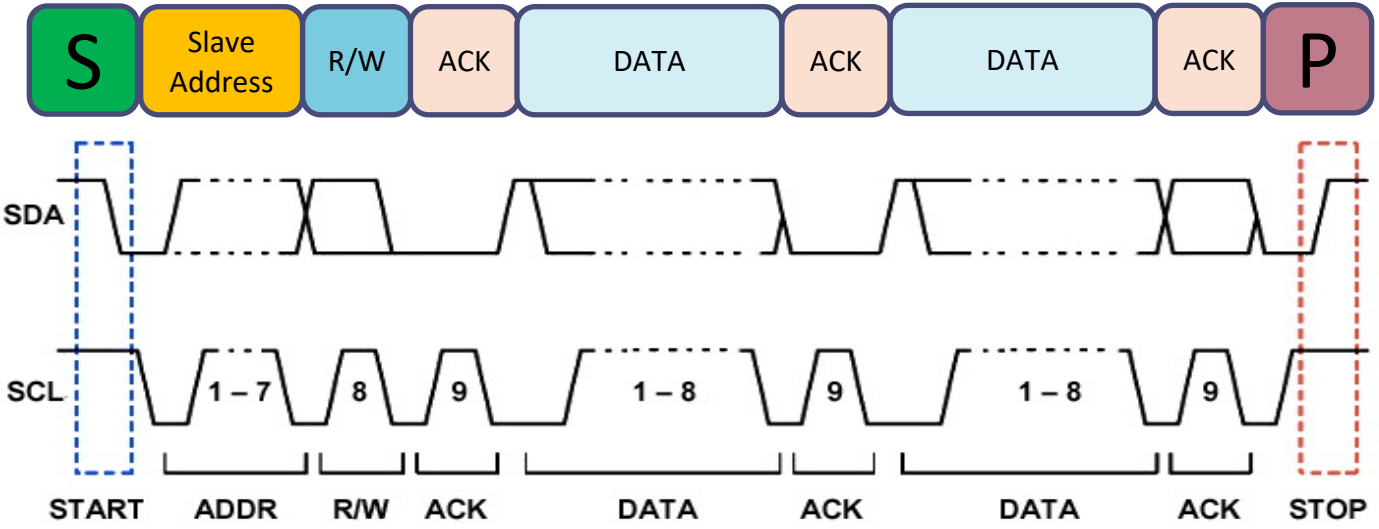




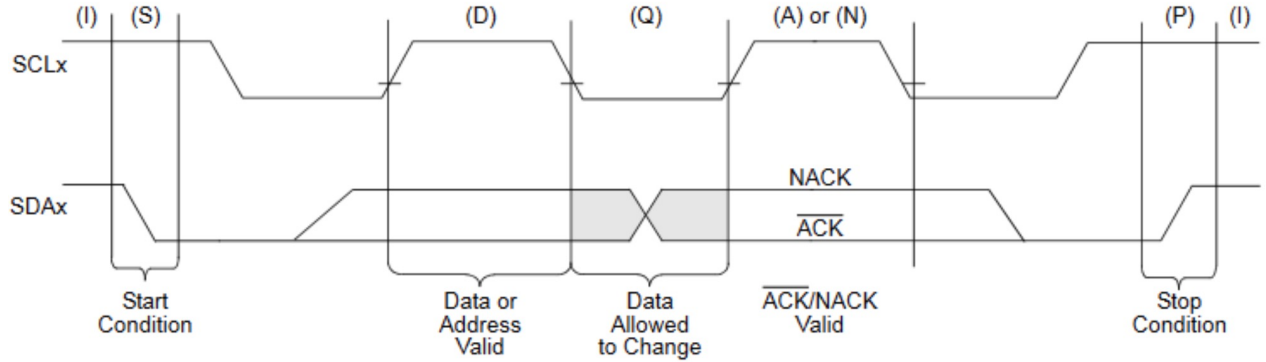
Cum arată un transfer I²C?

- Transmisia e încheiată de Master prin generarea unei condiții de Stop



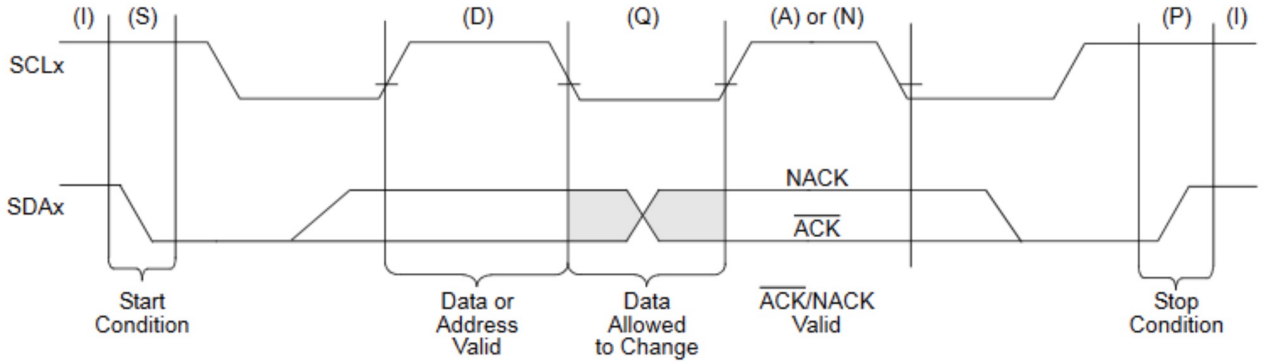


Condiții pe bus



- **Condiție de START (S)** După o stare de inactivitate a magistralei, o tranziție de la 1 la 0 a liniei SDA în timp ce ceasul (SCL) este 1 determină o condiție Start. Toate transferurile de date trebuie precedate de o condiție de Start.
- **Condiție de STOP (P)** O tranziție de la 0 la 1 a liniei SDA în timp ce ceasul (SCL) este 1 determină o condiție Stop. Toate transferurile de date trebuie să se încheie cu o condiție de Stop.
- **Repeated START (nu este prezentat)** O tranziție de la 1 la 0 a liniei SDA în timp ce ceasul (SCL) este r1 permite o condiție Start Repetat atunci când nu a apărut niciun STOP. Starturile Repetate permit unui master să schimbe direcția magistralei dispozitivului periferic adresat fără a renunța la controlul magistralei.

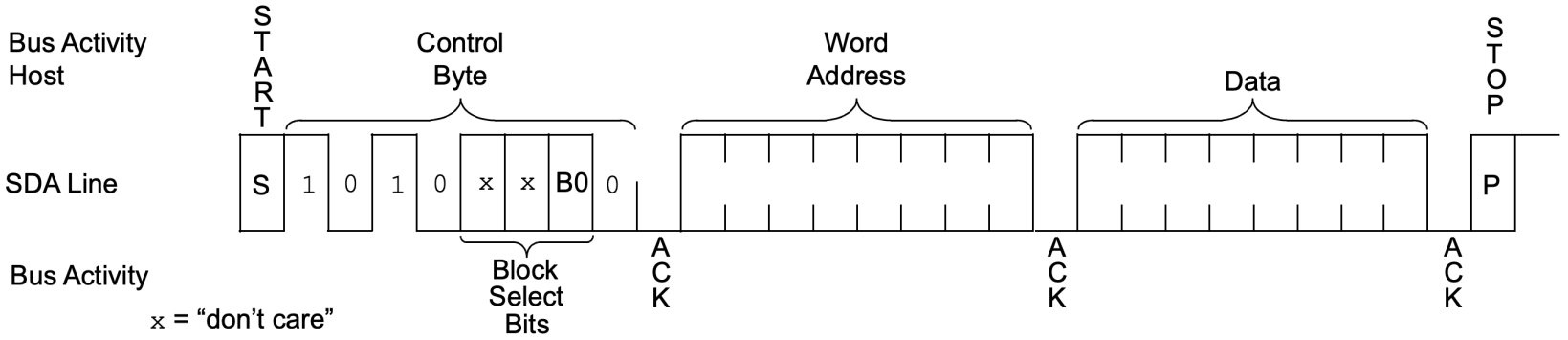
Condiții pe bus



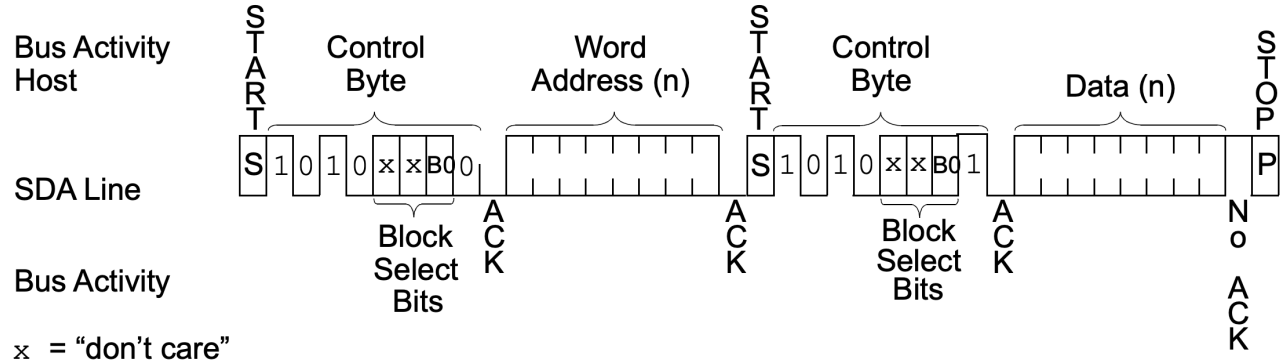
- **DATE VALIDE (D)** Starea liniei SDA reprezintă date valide atunci când, după o condiție Start, linia SDA este stabilă pe durata perioadei ridicate a semnalului de ceas. Există un bit de date per ciclu de ceas SCL.
- **ACK (A) SAU NACK (N)** Toate transmisiile de octeți de date trebuie confirmate (ACK) sau neconfirmate (NACK) de către receptor. Receptorul va trage linia SDA în jos pentru un ACK sau va elibera linia SDA pentru un NACK. Confirmarea este o perioadă de un bit folosind un ciclu de ceas SCL.
- **WAIT / DATA INVALID (Q)** Datele de pe linie trebuie modificate în timpul perioadei joase a semnalului de ceas. Dispozitivele pot prelungi, de asemenea, timpul de ceas scăzut prin activarea unui nivel scăzut pe linia SCL, provocând o așteptare pe magistrală.
- **MAGISTRALĂ INACTIVĂ (I)** Atât liniile de date, cât și cele de ceas rămân ridicate în acele momente după o condiție Stop și înainte de o condiție Start.

Exemplu: scriere și citire într-un EEPROM 24C04

Byte Write:



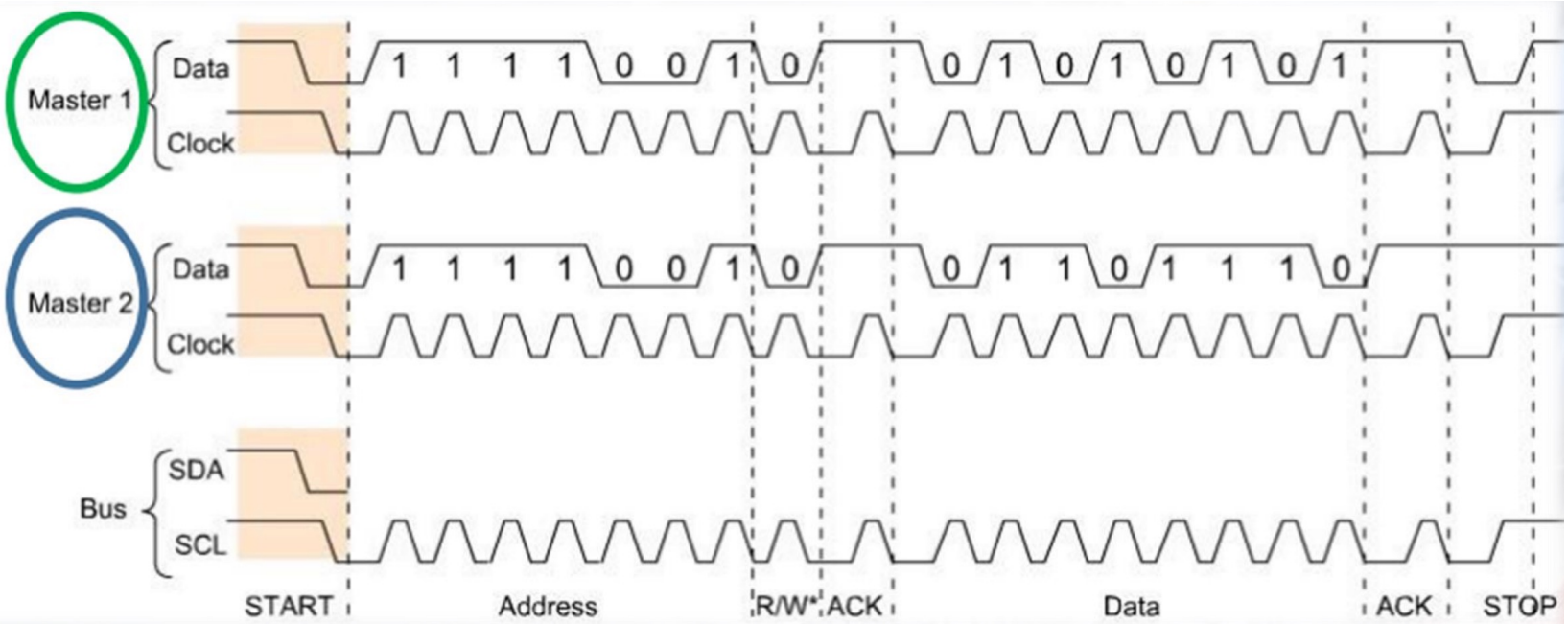
Byte Read:



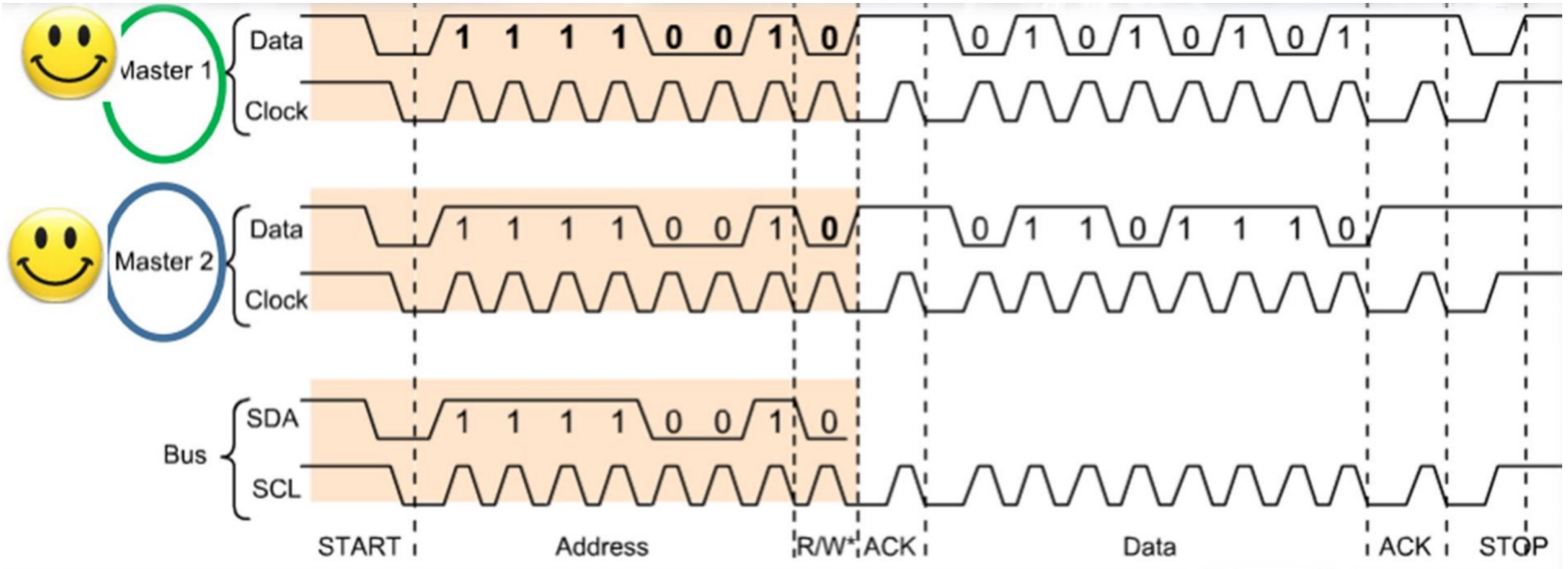
Arbitrare

- Ce se întâmplă dacă doi Master încep comunicația în același timp sau mai mulți Slave răspund la aceeași adresă?
- Fiecare device care scrie date pe SDA va și citi nivelul tensiunii pe linie ca să se asigure că este valoarea potrivită
- Dacă un device vede că linia are valoarea 0 logic, când se aștepta să aibă valoarea 1 logic, va trage concluzia că un alt dispozitiv folosește concurent linia SDA
  - În acel moment, dispozitivul va face back-off până la apariția unei condiții de Stop pe bus

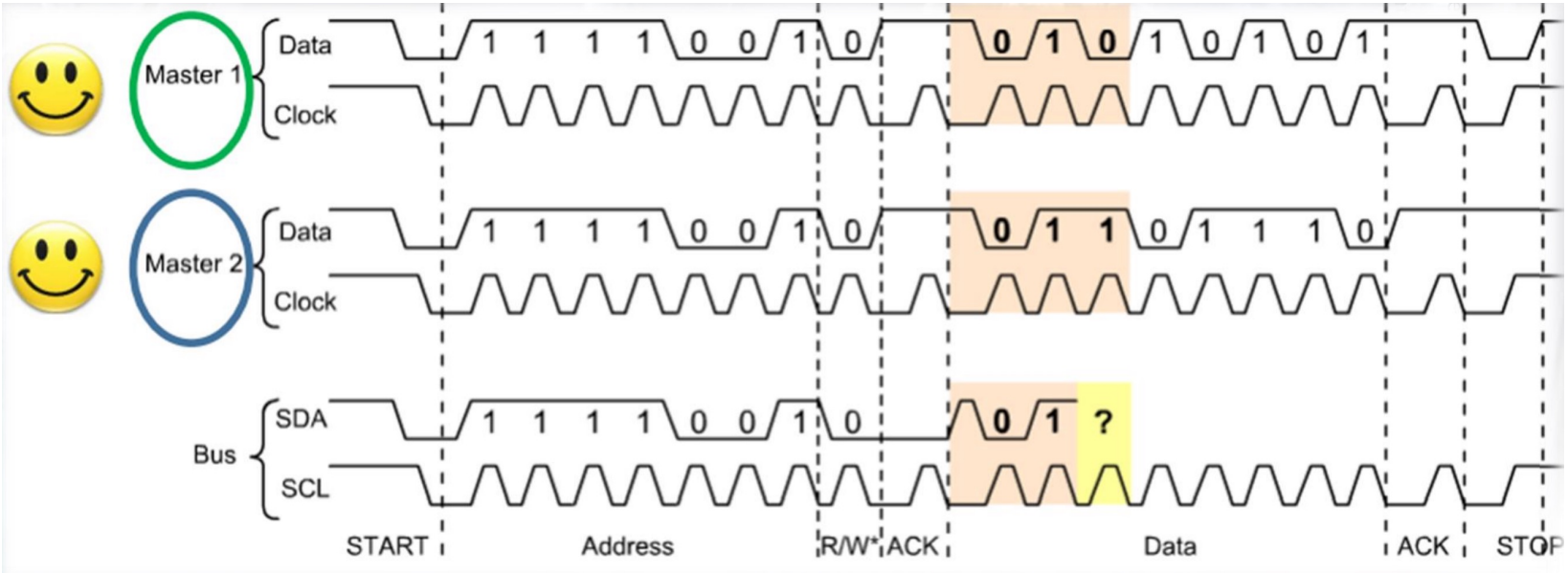
Exemplu de arbitrare



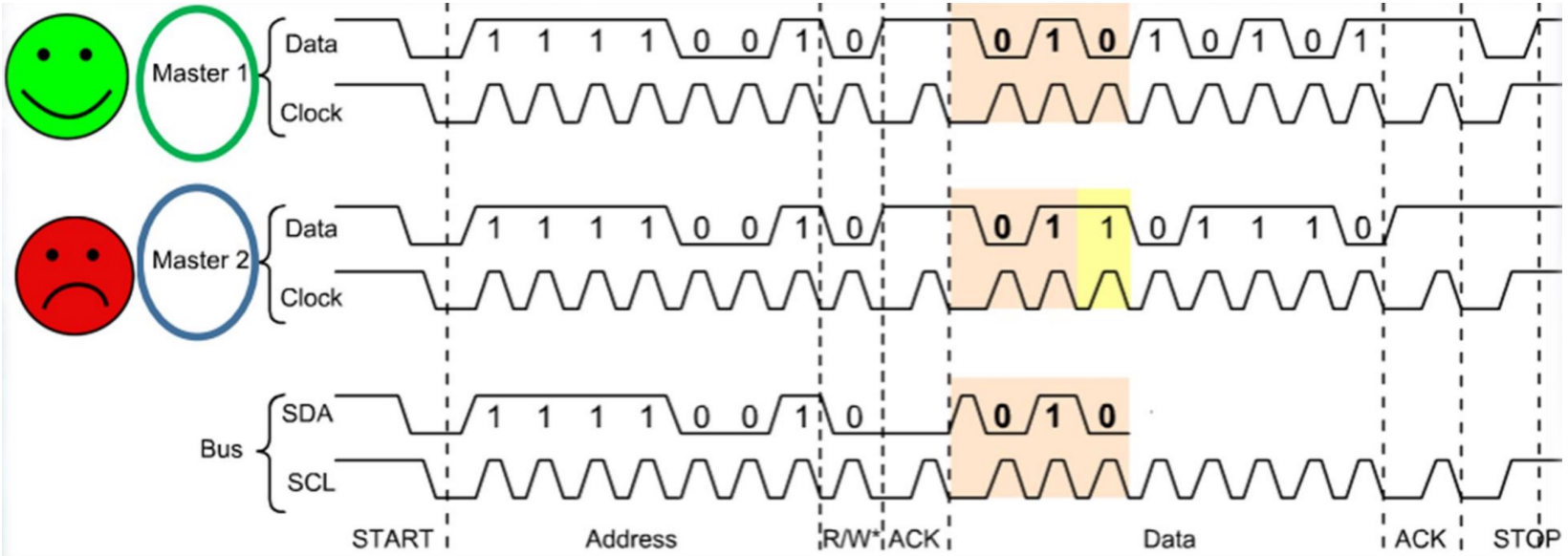
Exemplu de arbitrare



Exemplu de arbitrare



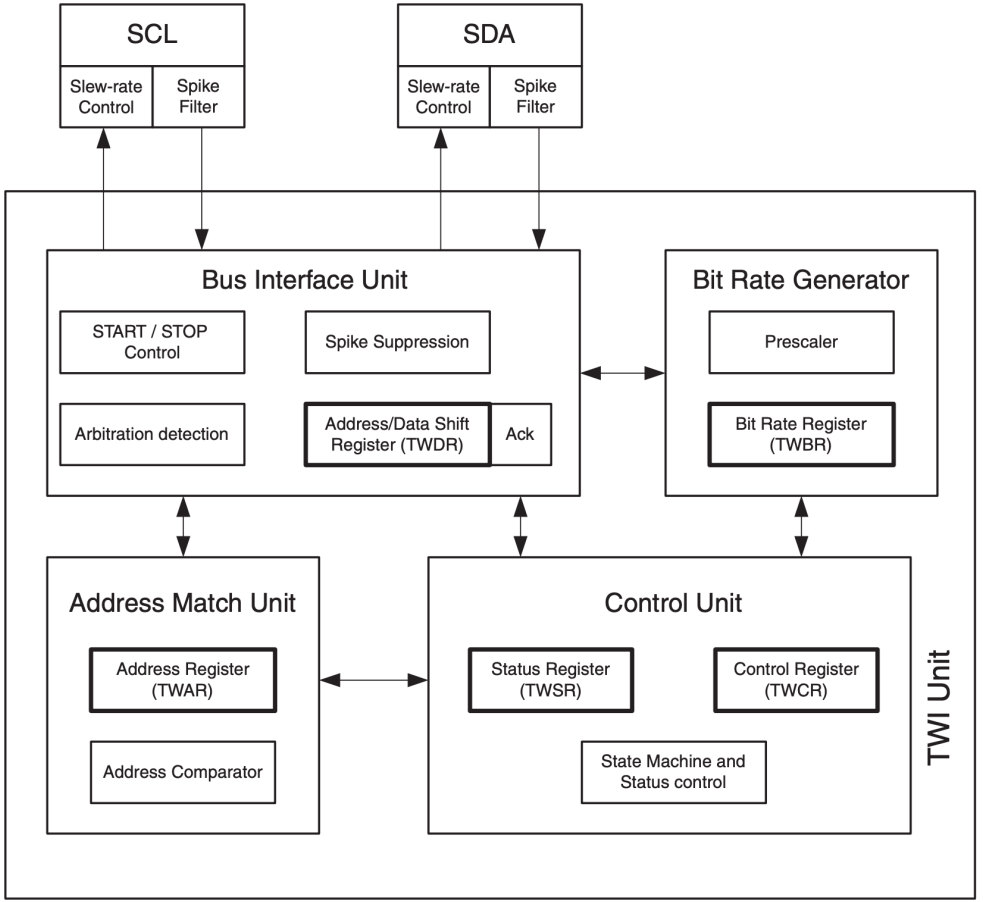
Exemplu de arbitrare



- Master 2 citește starea liniei SDA și vede că ea nu are valoarea implicită de 1 logic, ci este trasă la 0 de altcineva.
- În acest moment, Master 2 o să abandoneze transferul, până la apariția unei condiții de Stop.

I²C la ATMega324P

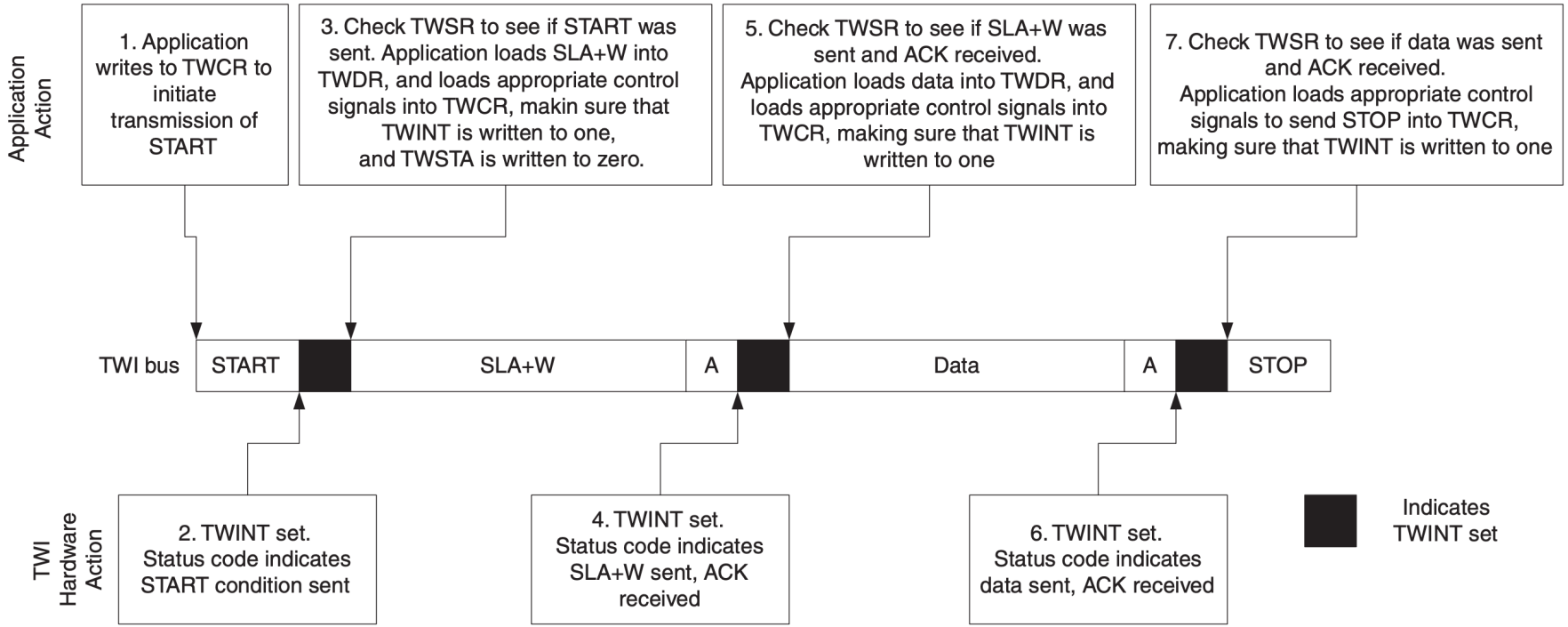
- I²C se numește 2-Wire Interface (TWI) la AVR
  - Complet compatibil I²C
  - Nu folosește numele pentru că e marcă înregistrată



Registre

- TWCR – flag-uri de control
- TWSR – biți de status
- TWDR – buffer de date
- TWAR – adresa dispozitivului (dacă e Slave)
- TWBR – viteza de transmisie
- Întreruperile se generează atunci când perifericul a terminat operația curentă și așteaptă următoarea comandă

O transmisie tipică pentru AVR



Implementare C

Mașina de stări I^2C este parțial implementată în hardware

Asta înseamnă că o bună parte din funcționalitate pentru un transfer trebuie să fie implementată în software

C Example	Comments
`TWCR = (1<<TWINT)|(1<<TWSTA)| (1<<TWEN)`	Send START condition
`while (!(TWCR & (1<<TWINT))) ;`	Wait for TWINT Flag set. This indicates that the START condition has been transmitted
`if ((TWSR & 0xF8) != START)` `ERROR();`	Check value of TWI Status Register. Mask prescaler bits. If status different from START go to ERROR
`TWDR = SLA_W;` `TWCR = (1<<TWINT) | (1<<TWEN);`	Load SLA_W into TWDR Register. Clear TWINT bit in TWCR to start transmission of address
`while (!(TWCR & (1<<TWINT))) ;`	Wait for TWINT Flag set. This indicates that the SLA+W has been transmitted, and ACK/NACK has been received.

Implementare C

Starea transceiverului I²C este vizibilă în registrul TWSR

La fiecare condiție (Start, Stop, ACK, NACK etc.) trebuie inspectat TWSR și luat o decizie în funcție de codul citit din registru

C Example	Comments

`if ((TWSR & 0xF8) != MT_SLA_ACK)`
 	
`ERROR();`
	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_SLA_ACK go to ERROR

`TWDR = DATA;`
`TWCR = (1<<TWINT) |`
`(1<<TWEN);`
	Load DATA into TWDR Register. Clear TWINT bit in TWCR to start transmission of data

`while (!(TWCR &`
`(1<<TWINT)))`
`;`
	Wait for TWINT Flag set. This indicates that the DATA has been transmitted, and ACK/NACK has been received.

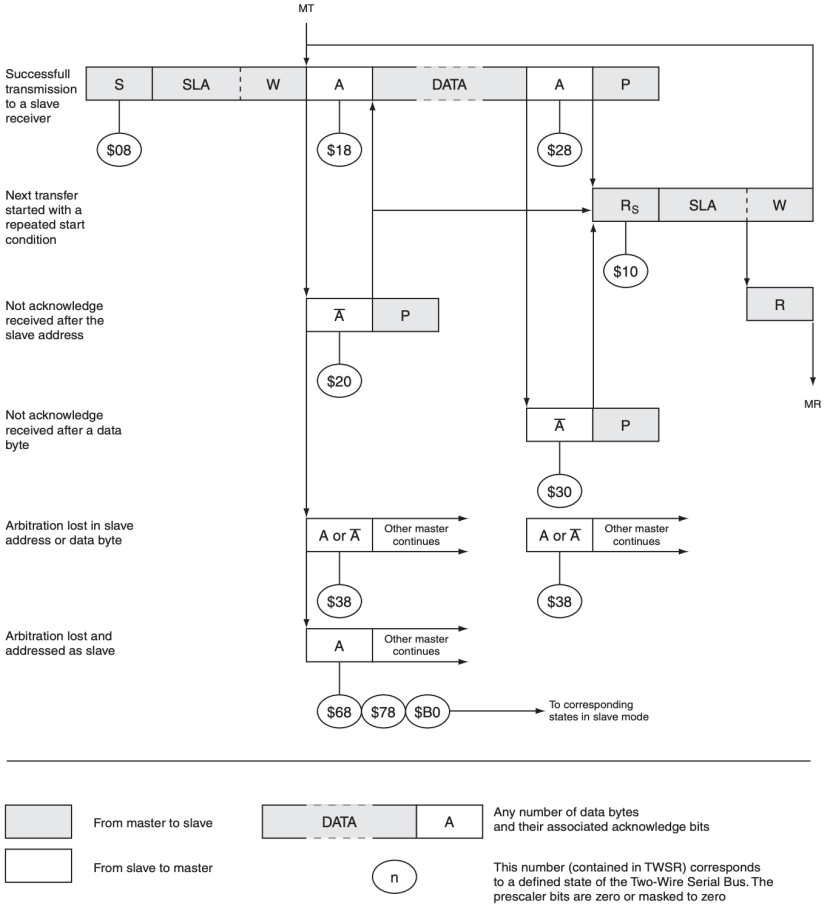
`if ((TWSR & 0xF8) != MT_DATA_ACK)`
 	
`ERROR();`
	Check value of TWI Status Register. Mask prescaler bits. If status different from MT_DATA_ACK go to ERROR

`TWCR =`
`(1<<TWINT)|(1<<TWEN)|`
`(1<<TWSTO);`
	Transmit STOP condition

Coduri status din TWSR pentru modul Master Transmitter

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	Next Action Taken by TWI Hardware
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
0x18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free

Aceleași coduri, dar în o diagramă

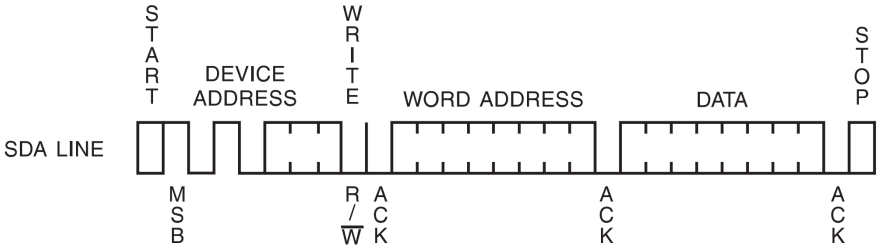


Implementare C

- Funcții de bază pentru transmisia de condiții de Start/Stop și de citire cu ACK/NACK
- Citirea codurilor de status din TWSR

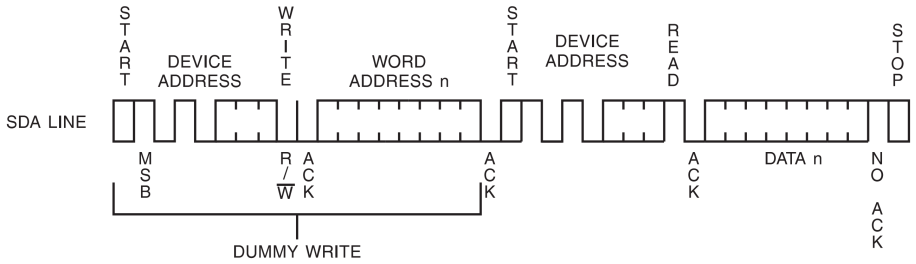
```
void TWIInit(void)
{
    //set SCL to 400kHz
    TWSR = 0x00;
    TWBR = 0x0C;
    //enable TWI
    TWCR = (1<<TWEN);
}

void TWIStart(void) /* send START condition */
{
    TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
    while ((TWCR & (1<<TWINT)) == 0);
}

void TWIStop(void) /* send STOP condition */
{
    TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
}

void TWIWrite(uint8_t u8data) /* write byte */
{
    TWDR = u8data;
    TWCR = (1<<TWINT)|(1<<TWEN);
    while ((TWCR & (1<<TWINT)) == 0);
}

uint8_t TWIReadACK(void) /* read with ACK */
{
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
    while ((TWCR & (1<<TWINT)) == 0);
    return TWDR;
}

uint8_t TWIReadNACK(void) /* read with a NACK */
{
    TWCR = (1<<TWINT)|(1<<TWEN);
    while ((TWCR & (1<<TWINT)) == 0);
    return TWDR;
}

uint8_t TWIGetStatus(void) /* read status codes */
{
    uint8_t status;
    //mask status
    status = TWSR & 0xF8;
    return status;
}
```

Scriere în EEPROM




```
uint8_t EEWriteByte(uint16_t u16addr, uint8_t u8data)
{
    TWIStart();
    if (TWIGetStatus() != 0x08)
        return ERROR;
    //select devise and send A2 A1 A0 address bits
    TWIWrite((EEDEVADR)|(uint8_t)((u16addr & 0x0700)>>7));
    if (TWIGetStatus() != 0x18)
        return ERROR;
    //send the rest of address
    TWIWrite((uint8_t)(u16addr));
    if (TWIGetStatus() != 0x28)
        return ERROR;
    //write byte to eeprom
    TWIWrite(u8data);
    if (TWIGetStatus() != 0x28)
        return ERROR;
    TWIStop();
    return SUCCESS;
}
```

Citire din EEPROM

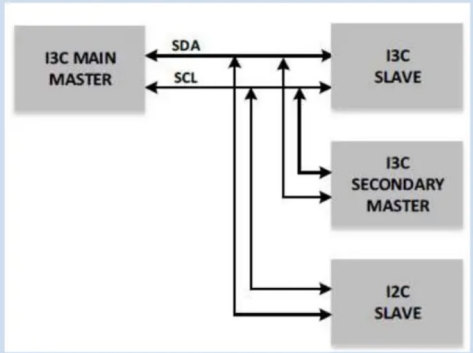
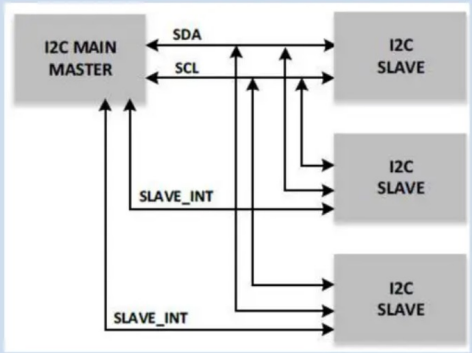
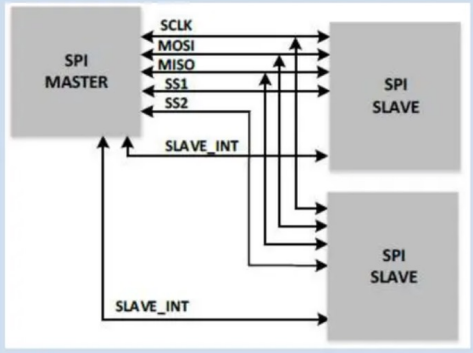



```
uint8_t EEReadByte(uint16_t u16addr, uint8_t *u8data)
{
    TWIStart();
    if (TWIGetStatus() != 0x08)
        return ERROR;
    //select device and send A2 A1 A0 address bits
    TWIWrite((EEDEVADR)|((uint8_t)((u16addr & 0x0700)>>7)));
    if (TWIGetStatus() != 0x18)
        return ERROR;
    //send the rest of address
    TWIWrite((uint8_t)(u16addr));
    if (TWIGetStatus() != 0x28)
        return ERROR;
    //send start
    TWIStart();
    if (TWIGetStatus() != 0x10)
        return ERROR;
    //select device and send read bit
    TWIWrite((EEDEVADR)|((uint8_t)((u16addr & 0x0700)>>7))|1);
    if (TWIGetStatus() != 0x40)
        return ERROR;
    *u8data = TWIReadNACK();
    if (TWIGetStatus() != 0x58)
        return ERROR;
    TWIStop();
    return SUCCESS;
}
```

Variante de I^2C

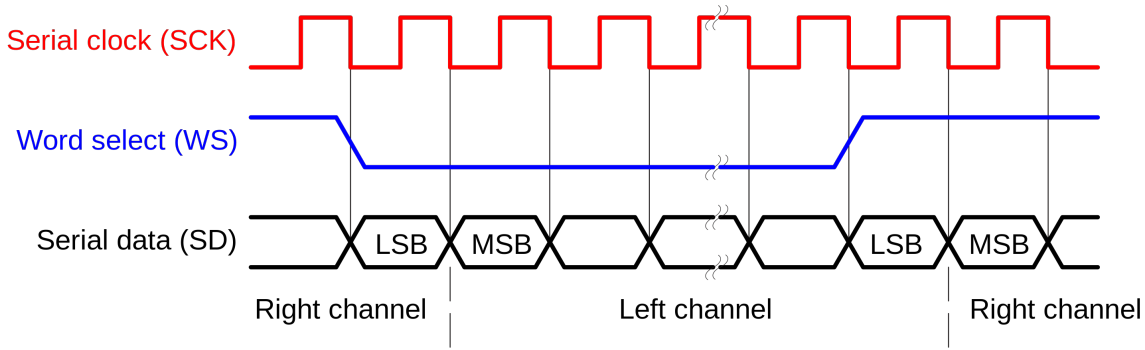
- SMB
  - System Management Bus – standard derivat din I2C pentru comunicația pe placa de bază a unui PC
  - Interfațează senzori de temperatură, ventilatoare, senzori de tensiune, lid switches, iluminare RGB etc.
- I^3C
  - Standard apărut în 2017 făcut de NXP (Philips legacy)
  - Viteze mai mari de transfer (max 30Mbps) – alternativă SPI
  - Eficiență energetică mărită
  - Funcții avansate (ex. in-band Interrupts – Slave poate să notifice Master)

I^2C vs. I^3C vs. SPI

Parameter	MIPI I3C	I2C	SPI
Overview			
# of lines	2-wire	2-wire (plus separate wires for each required interrupt signal)	4-wire (plus separate wires for eachrequired interrupt signal
Effective Data Bitrate	33.3 Mbps max at 12.5 MHz (Typ.:10.6 Mbps at 12MHz SDR)	3 Mbps max at 3.4 MHz (Hs) 0.8 Mbps max at 1 MHz (Fm+) 0.35 Mbps max at 400 KHz (Fm)	Approx. 60 Mbps max at 60 MHz for conventional implementations (Typically: 10 Mbps at 10 MHz

I²S

- Inter-IC Sound
- Trei linii de comunicație



- Serial Clock (SCK) este neîntrerupt
  - Frecvența ceasului depinde de rata de eșantionare și mărimea unui eșantion
  - De exemplu, 44.1kHz × 16 biți per sample × 2 canale = 1.4112 MHz
- Word Select (WS) discriminează între canalul audio stânga și dreapta
- Serial Data (SD) codifică bit cu bit informația sonoră în format PCM

Concluzii

- I^2C este un protocol serial foarte versatil
- Adresarea se face prin linia de date
- Suport multi-Master
- Viteză de transfer mai mică decât SPI
- Necesită software mai complex