C8

Interfetele seriale

I²C, SPI, I²S, I3C

https://circuitdigest.com/tutorial/serial-communication-protocols
https://circuitdigest.com/article/introduction-to-bit-banging-spi-communication-in-arduino-via-bit-banging

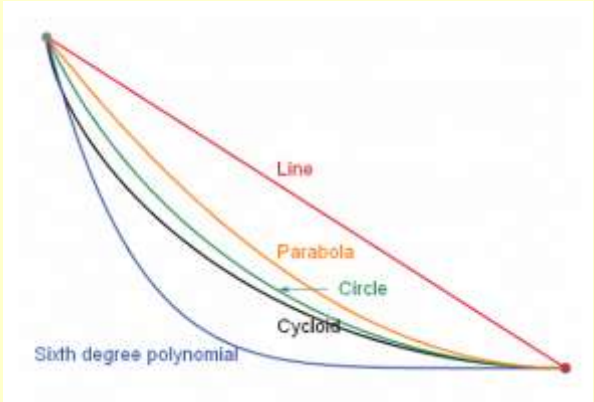
https://www.edn.com/fundamentals-of-i3c-interface-communication/
https://www.mipi.org/resources/i3c-frequently-asked-questions

Interface	Format	Number of Devices (maximum)	Distance (maximum, ft)	Speed (maximum, bps)	Typical Use
RS-232 (TIA-232)	asynchronous serial	2	50-100	20k (faster with some hardware)	Modem, basic communications
RS-485 (TIA-485)	asynchronous serial	32 unit loads (up to 256 devices with some hardware)	4000	10M	Data acquisition and control systems
Ethernet	serial	1024	1600	10G	PC network communications
IEEE-1394b (FireWire 800)	serial	64	300	3.2G	Video, mass storage
IEEE-488 (GPIB)	parallel	15	60	8M	Instrumentation
I^2C	synchronous serial	40	18	3.4M	Microcontroller communications
Microwire	synchronous serial	8	10	2M	Microcontroller communications
MIDI	serial current loop	2 (more with flow-through mode)	50	31.5k	Music, show control
Parallel Printer Port	parallel	2 (8 with daisy-chain support)	10–30	8M	Printer
SPI	synchronous serial	8	10	2.1M	Microcontroller communications
USB	asynchronous serial	127	16 (up to 98 ft with 5 hubs)	1.5M, 12M, 480M	PC peripherals

Bus-urile I^2C si SPI

- Protocoale de comunicare seriala, sincrona
- Concepute ptr. distante scurte “inside the box”
- Complexitate redusa
- Cost redus
- Viteza (~ Mbps)

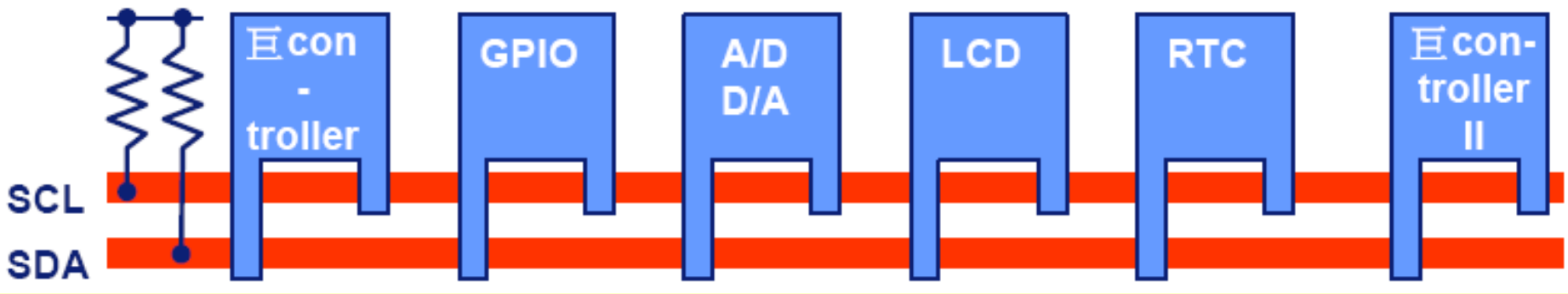
https://www.youtube.com/watch?v=DYCTwqOeHMM



Ce reprezinta bus-ul I^2C?

- I^2C prescurtarea de la "Inter-integrated circuit" bus
- Dezvoltat de Philips/NXP Semiconductor ptr. TV in anii '80
- Dispozitive cu interfata I^2C : EEPROMs, sensori de temperatura, real-time clocks, ADC, port I/O etc.
- I^2C este utilizata ca interfata de control la dispozitive tip "signal processing" care au interfete de date separate, de ex. RF tuners, video decoders & encoders si audio processors
- I^2C bus are 3 viteze:
    - Slow (<100 Kbps)
    - Fast (<400 Kbps)
    - High-speed (3.4 Mbps) – I^2C v.2.0
- Distanta limitata la ~10 ft (3m) la viteze moderate
- Numarul dispozitivelor este limitat de capacitate < 400 pF/linie

Configurarea Bus-ului I²C



Caracteristici ale bus-ului I2C

- bus serial, 2-fire - Serial data (SDA)
  - Serial clock (SCL)
  - *Bus sincron, half-duplex, multi-master*
- Nu necesita chip select (CS) sau logica de arbitrare
- Dispozitivele au adresa unica
- Liniile sunt “pulled-up” prin rezistente si “pulled down” prin drivere open-drain (SI cablat)

Configurarea Bus-ului I²C

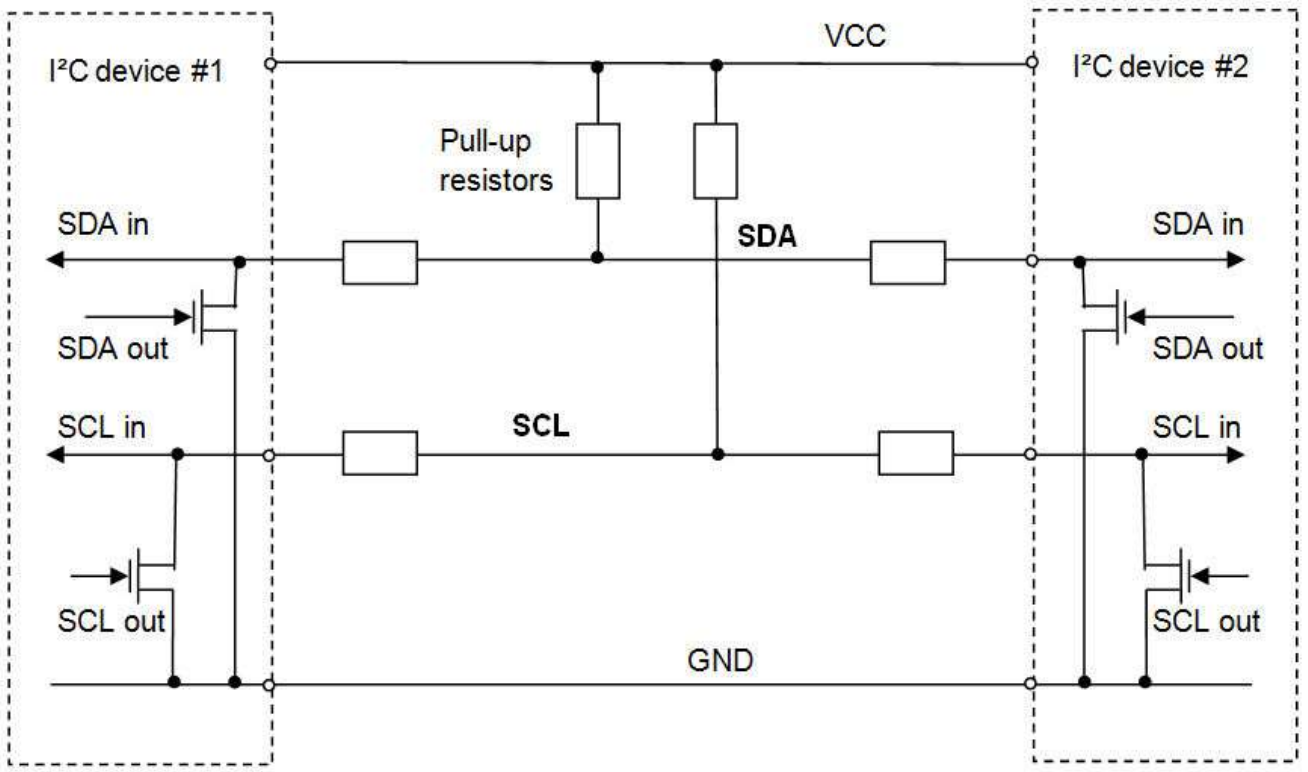
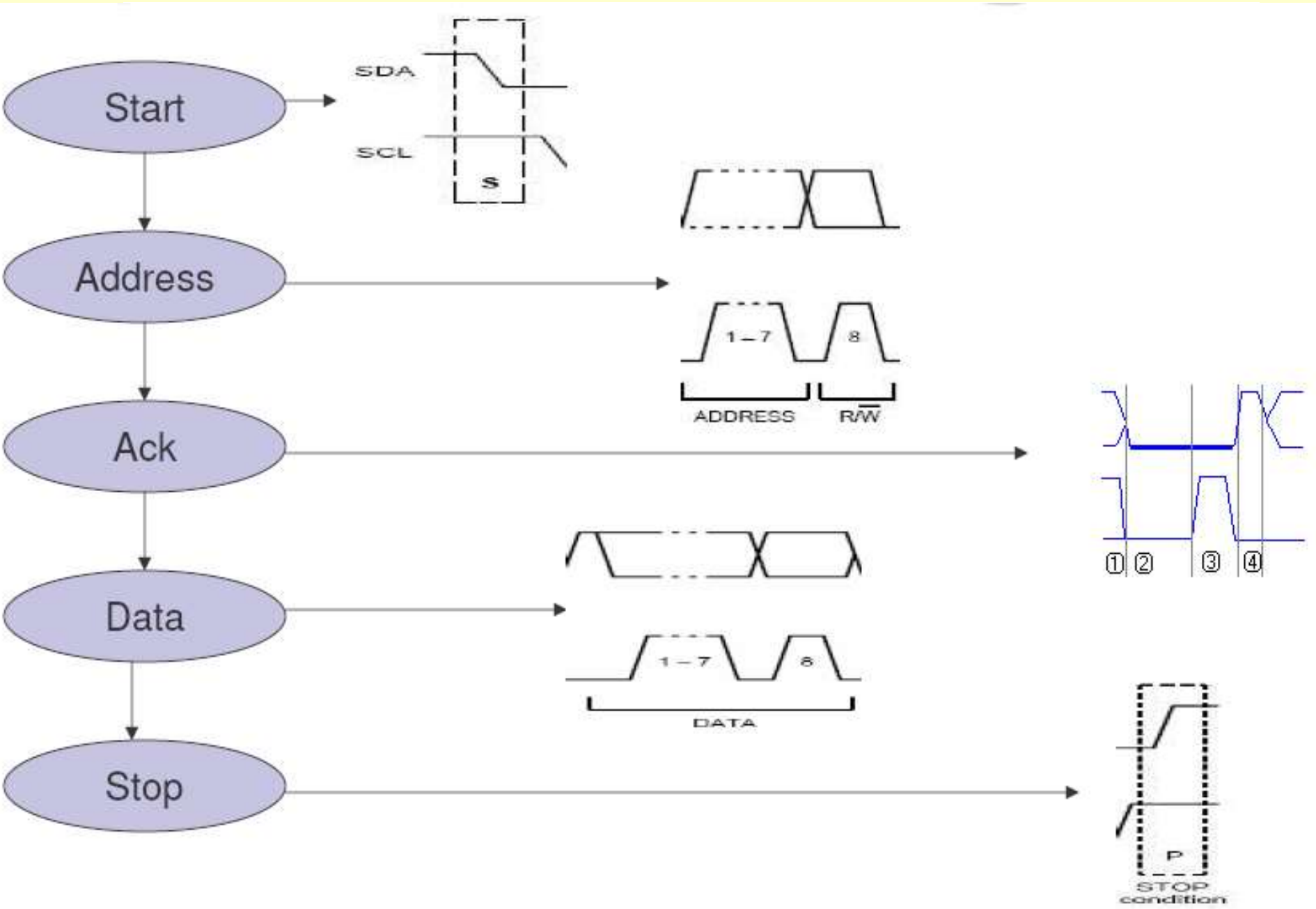


Figure 4: I²C bus with 2 devices connected. SDA and SCL are connected to VCC through pull-up resistors. Each device controls the bus lines outputs with open drain buffers.

- Utilizarea “R de pull-up” pentru a stabili un “1’’ logic, limitează viteza maximă a bus-ului. Acest lucru poate fi un factor de limitare pentru multe aplicații. Acesta este motivul pentru care a fost introdus modul de ***3.4 Mbps de mare viteză***. Înainte de a folosi acest mod, master-ul bus-ului trebuie să emită un cod specific “High speed Master", intr-un mod de viteză mai mică de ex. 400kbps.

Protocolul I²C



Protocolul I²C

1. Master-ul trimite conditia de start (S) si *controleaza clock-ul*
2. Master-ul trimite o adresa unica de 7-biti a unui dispozitiv slave +
3. 1 bit read/write (R/W) – 0 - slave primeste,

   1 - slave transmite
4. Receptorul trimite bitul de acknowledge (ACK)
5. Transmitatorul (slave/master) transmite 1 byte de date
6. Receptorul trimite bit acknowledge (ACK) pentru byte-ul receptionat

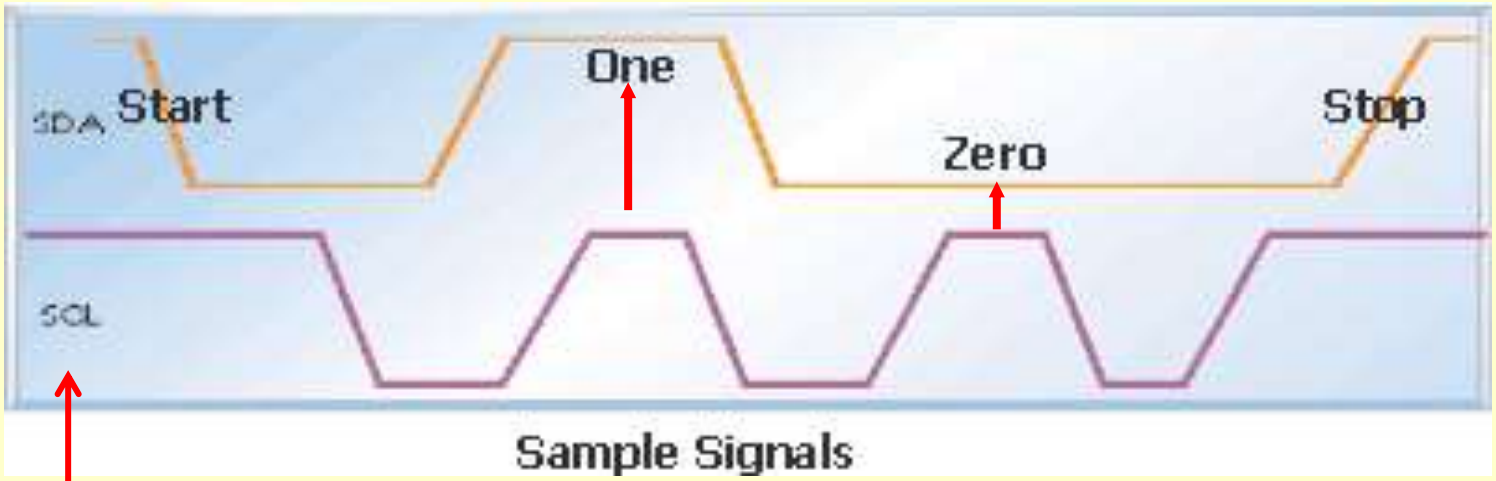
Protocolul I²C (cont.)

7. Repet pasii 5+6 daca se transmit mai multi bytes de date

8a) Ptr. tranzactie de scriere (master transmite), master genereaza conditia de stop (P) dupa ultimul byte de date

8b) Ptr. tranzactie de citire (master receptioneaza), master-ul nu trimite ACK ptr. byte-ul final, ci genereaza conditia de stop (P), ptr. a anunta slave-ul ca transmisia este terminata

Semnalele bus-ului I^2C

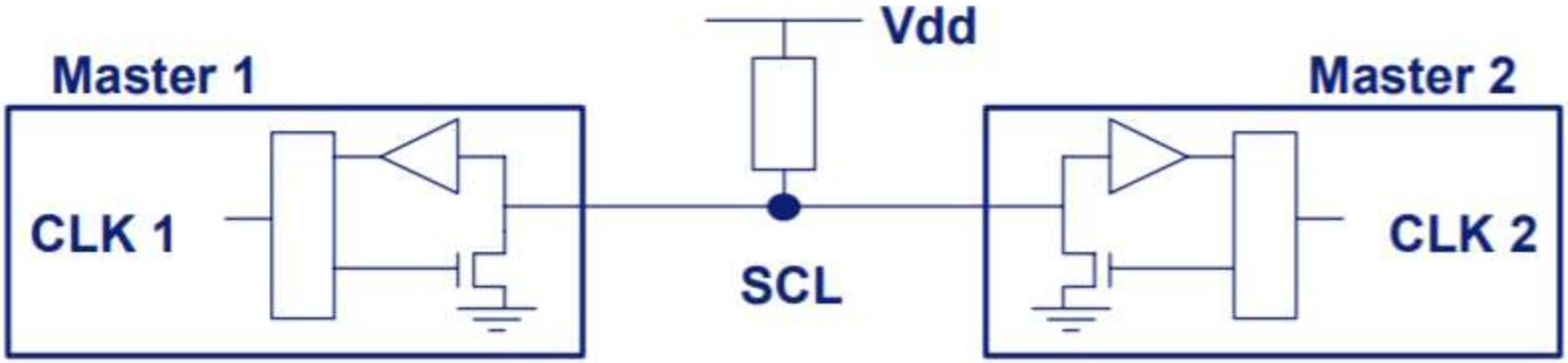


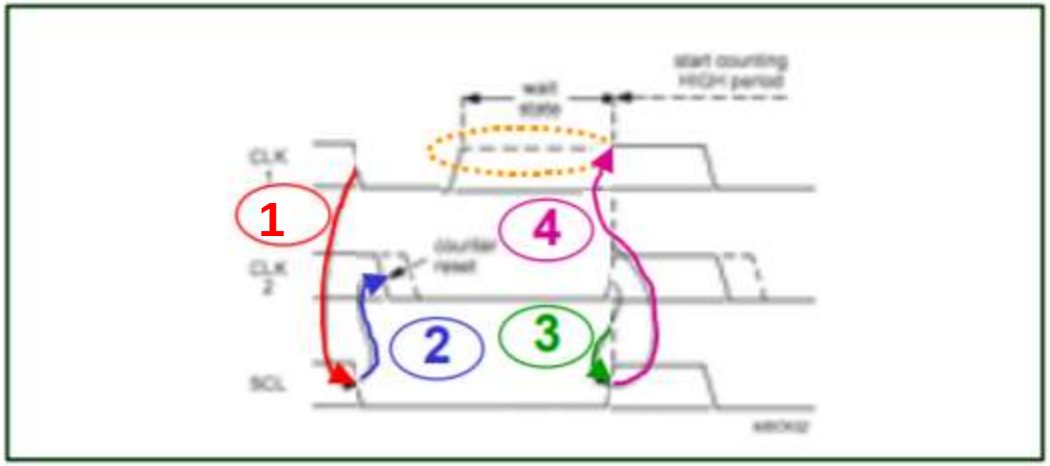
- Idle state - SDA si SCL sunt inactive, H
- Start – tranzitie H-L a liniei SDA cand linia SCL este H
- Stop – tranzitie L-H a liniei SDA cand linia SCL este H
- Ack – receptorul trage SDA low, in timp ce transmitatorul permite aceasta tinand linia high
- Data –***tranzitia pe date se face cand SCL=L*** si e valida cand SCL=H

Caracteristici I^2C

- "Clock stretching" – Cand un slave (receiver) necesita mai mult timp sa proceseze un bit, acesta poate forta SCL low. Master-ul asteapta pâna slave-ul a deblocat SCL inainte de a trimite bitul urmator.
- "General call" broadcast – adreseaza toate dispozitivele de pe bus
- Adresare extinsa - pe 10-biti pentru noile device-uri, adresarea pe 7-biti este complet epuizata.
- Sincronizarea clock-ului
- Arbitrajul pe bus

I²C Protocol - Clock Synchronization

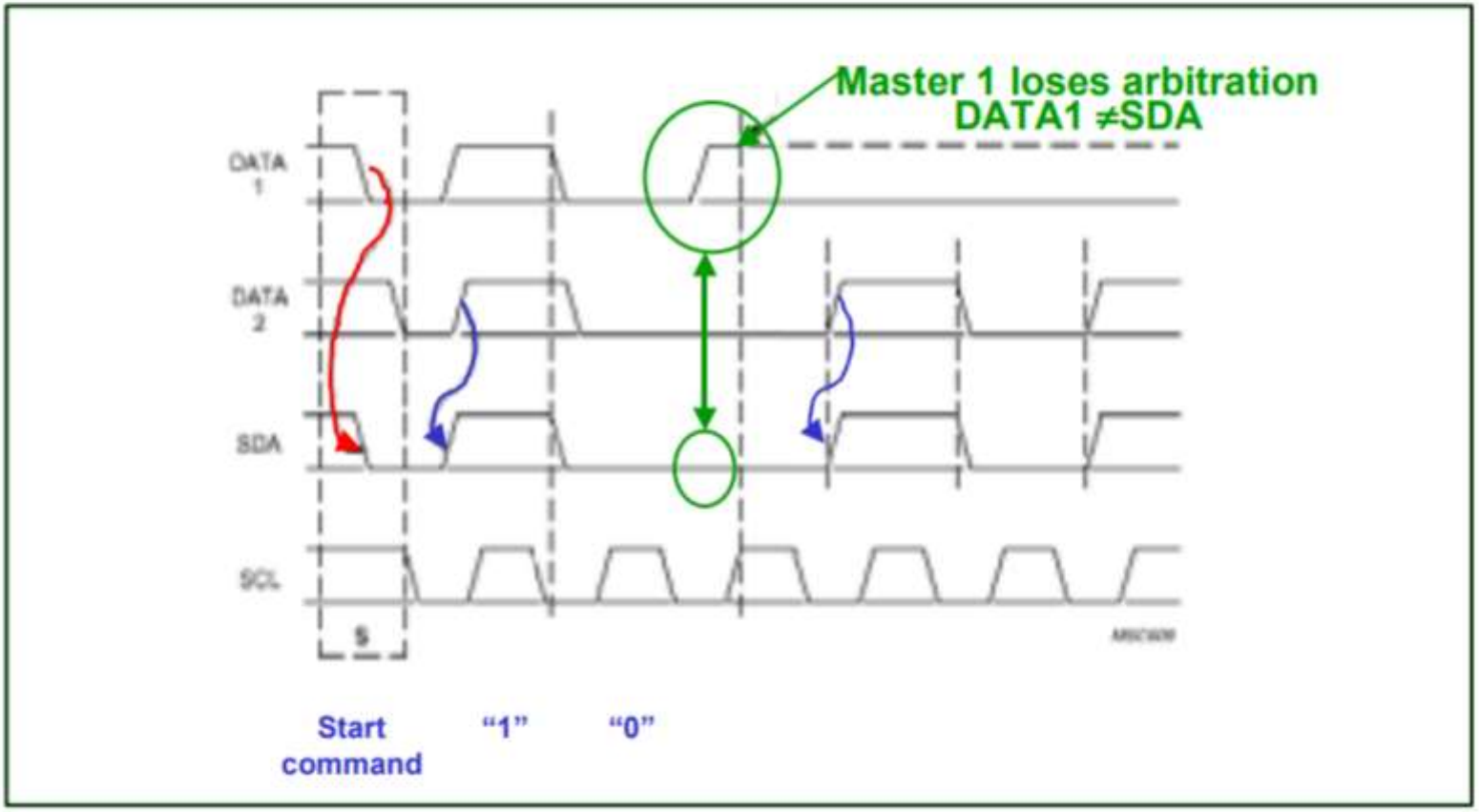


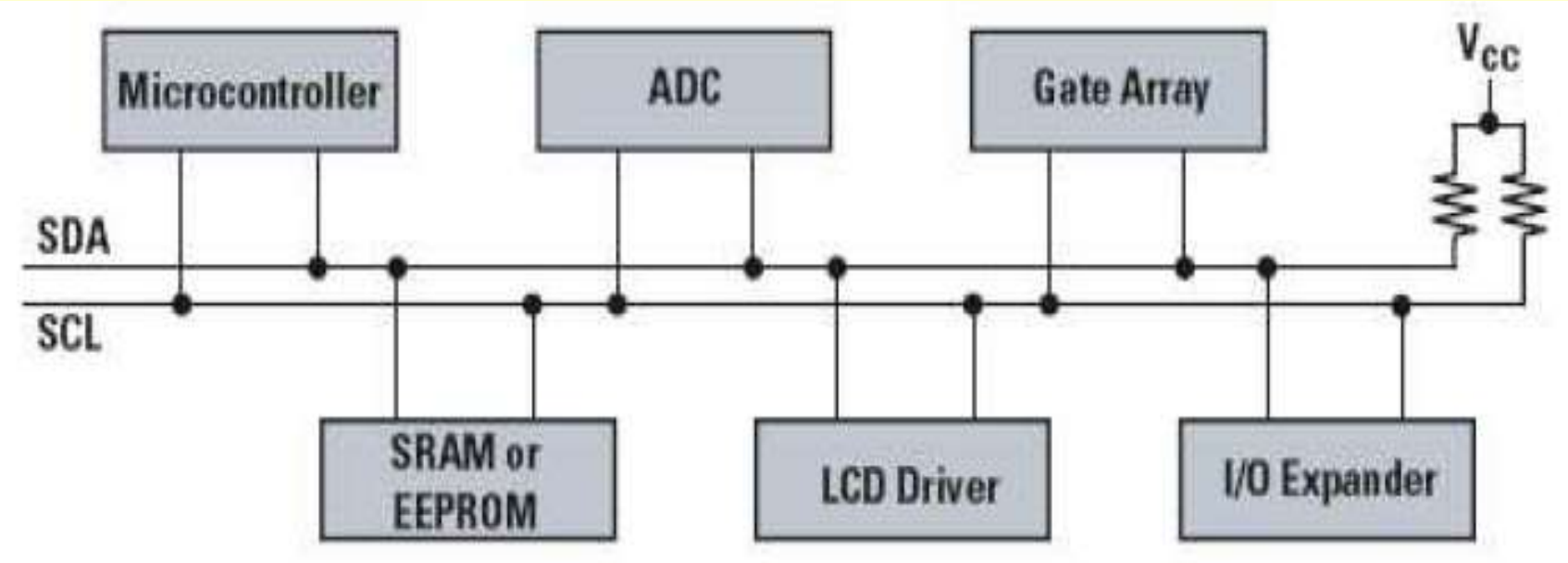


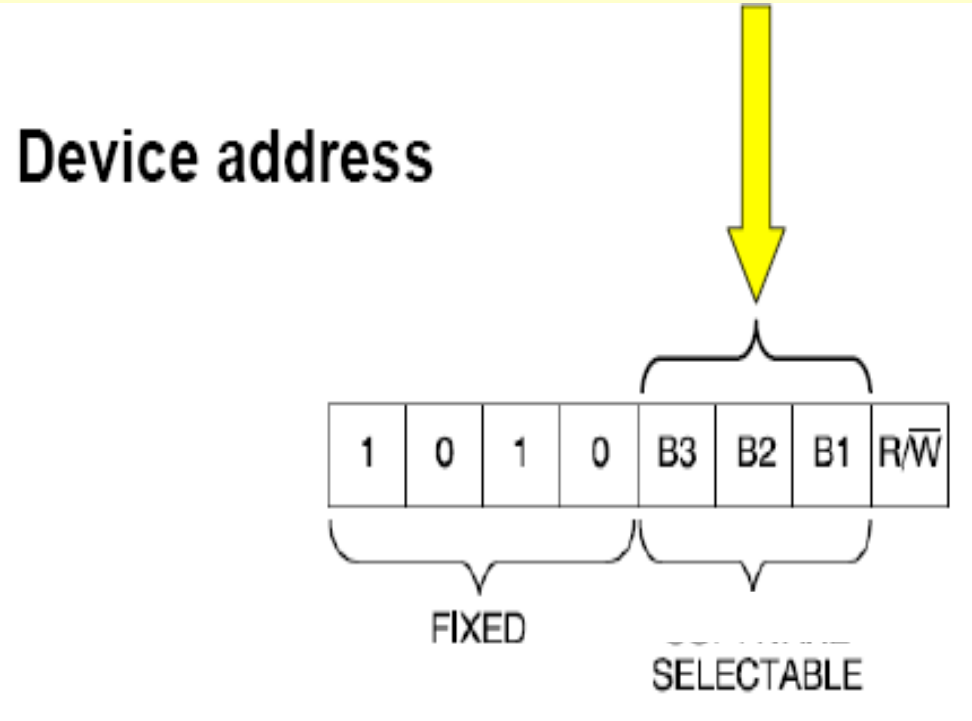
- **LOW** period determined by the **longest clock LOW** period
- **HIGH** period determined by **shortest clock HIGH** period

I²C Protocol - Arbitration

- Two or more masters may generate a START condition at the same time
- Arbitration is done on SDA while SCL is HIGH - Slaves are not involved

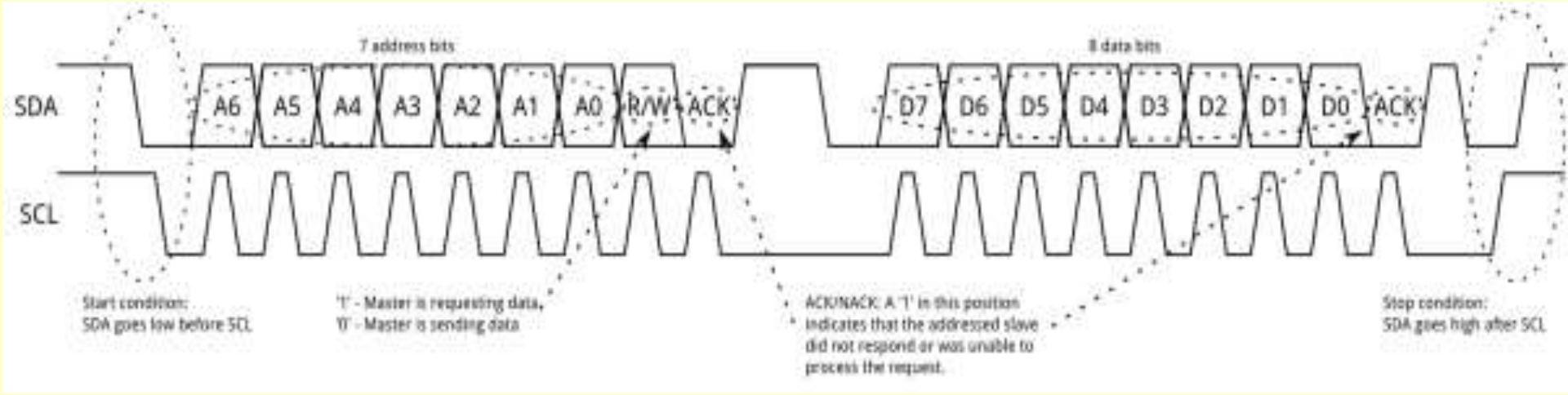






Adrese Speciale

- general call 0000 000 | 0
- start byte 0000 000 | 1
- CBUS address 0000 001 | *
  - – used for cooperation of I2C and CBUS
- High-speed mode master code 0000 1**|*
- 10-bit slave addressing 1111 0**|*

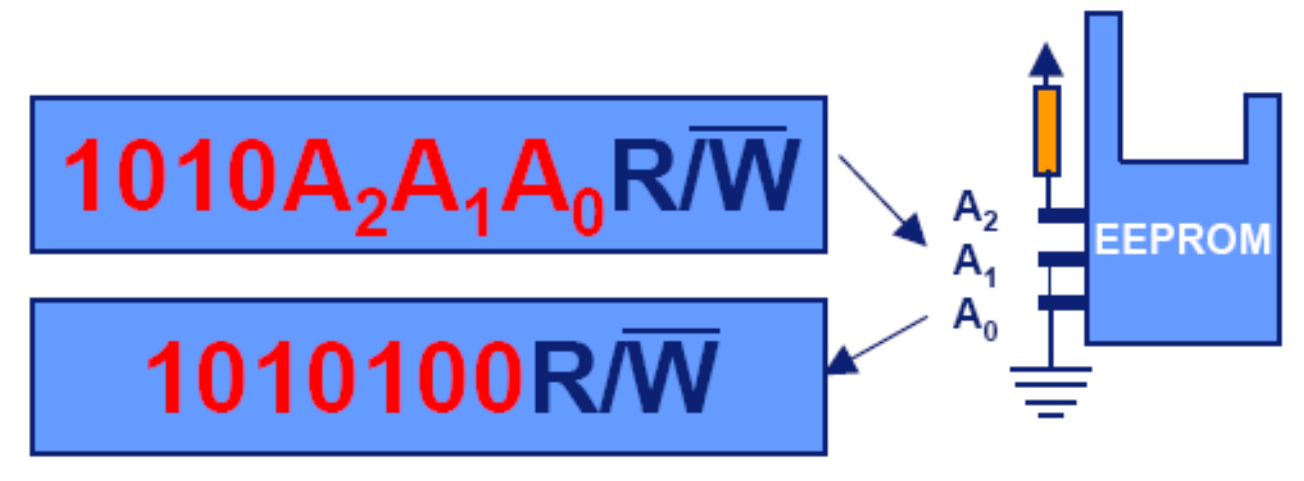


- Într-un sistem de adresare pe 10 biți, sunt transmise două cadre ptr. adresa slave
- Primul cadru va consta din codul binar 11110xyz, caz în care "x" este MSB-ul adresei slave(9), y este bitul 8 din adresa slave, iar z este bitul de R/W
- Primul ACK va fi generat de toti slaves care au acesti primi doi biți ai adresei. Ca și în cazul unui transfer normal de 7 biți, imediat incepe un transfer care conține biți 7:0 ai adresei. În acest moment, slave-ul adresat va răspunde cu un bit ACK. Dacă nu, este activat modul ca la un sistem de adresa pe 7-biti.
- Dispozitivele cu adresa pe 10 biți pot coexista cu dispozitivele pe adresa de 7 biți, deoarece prefixul "11110" din partea adresei nu apare la nici o adresa valida de 7 biți

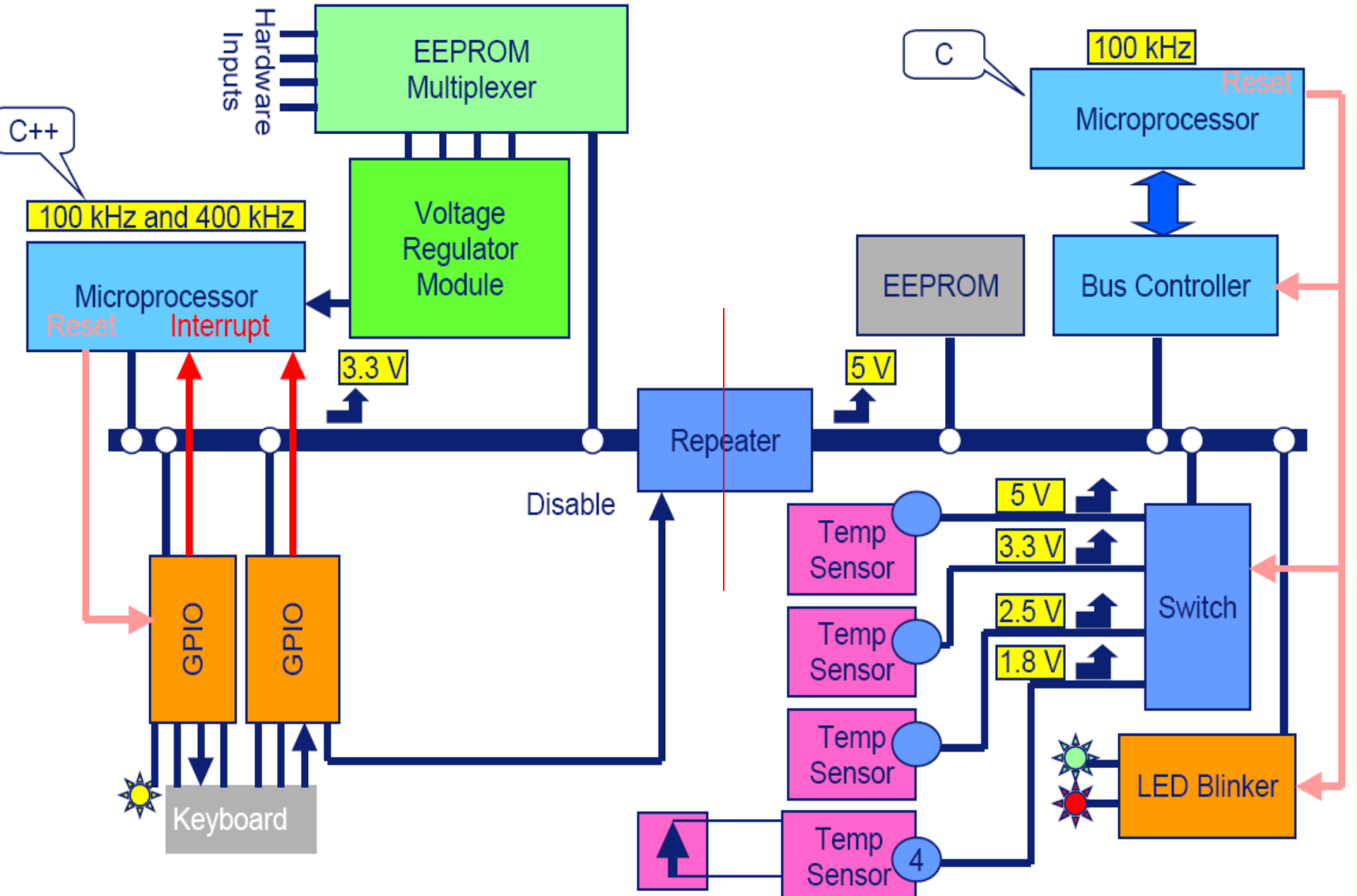
Philips/NXP I2C Logic Devices

- Dispozitivele I2C sunt impartite in 10 clase diferite
- Philips ofera > 63 de diferite dispozitive I2C

- Bus Controllers
- I/O Expanders
- Serial EEPROMs
- Multiplexers and Switches
- 7 Segment Drivers
- Temperature Sensors
- LED Blinkers
- DIP Switches
- Repeaters/Hubs/Extenders
- Analog/Digital Converters



Complex I²C Bus Arrangement



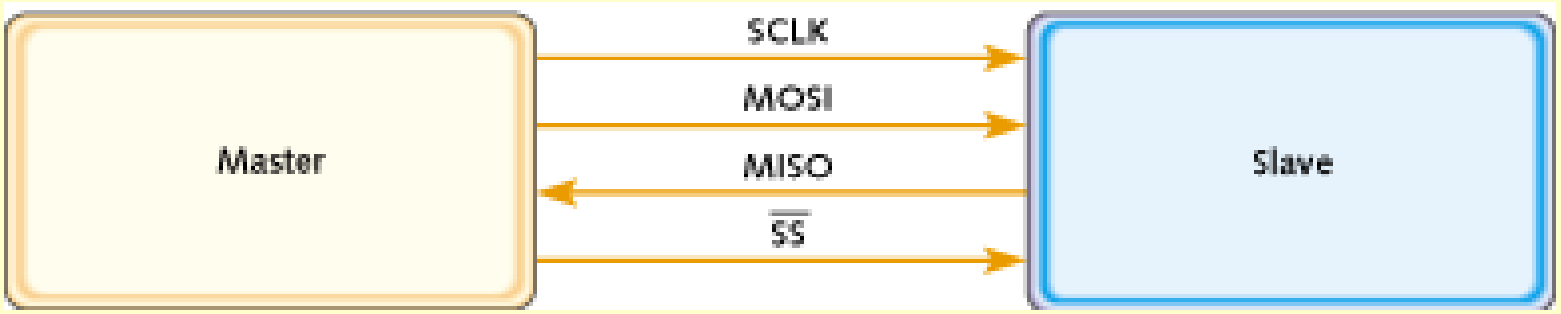
Bus-ul SPI

- Prescurtare de la “Serial Peripheral Interface”
- Definit de Motorola la familia de microcontrollere MC68HCxx
- In general, mai rapid decat I^2C, capabil de a transmite ~Mbps

Aplicatii:

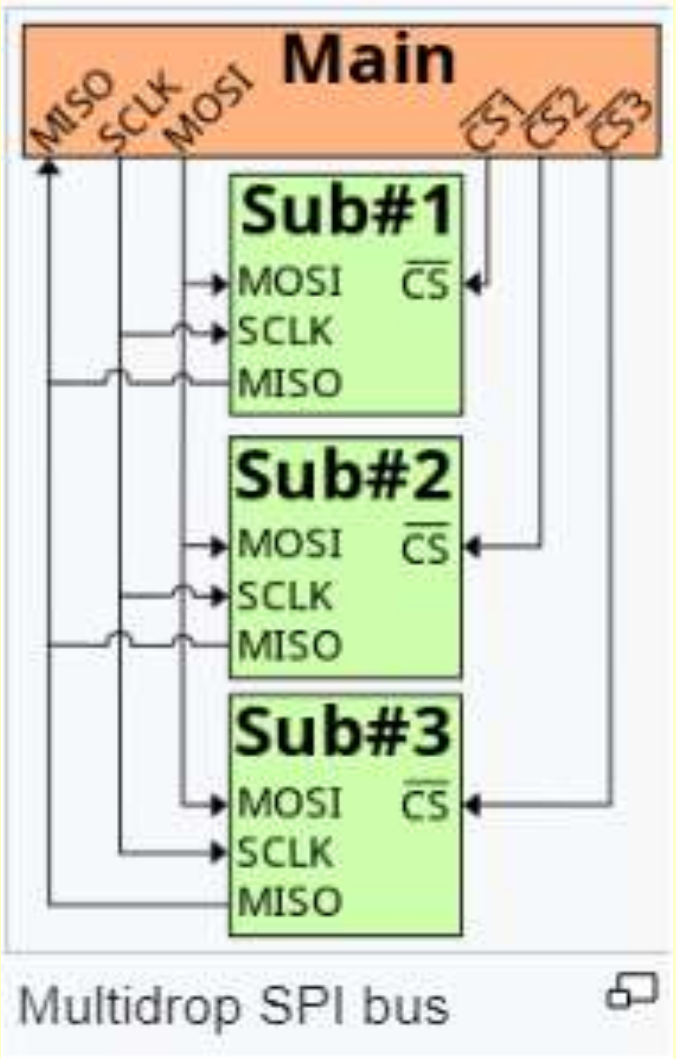
- Ca si I^2C, folosit la EEPROM, Flash, real time clocks, CAN,...
- Mai bun ptr. aplicatii “data streams”, e.g. convertoare ADC
- Full duplex, ex. comunicatii intre un codec si un procesor de semnal (DSP)

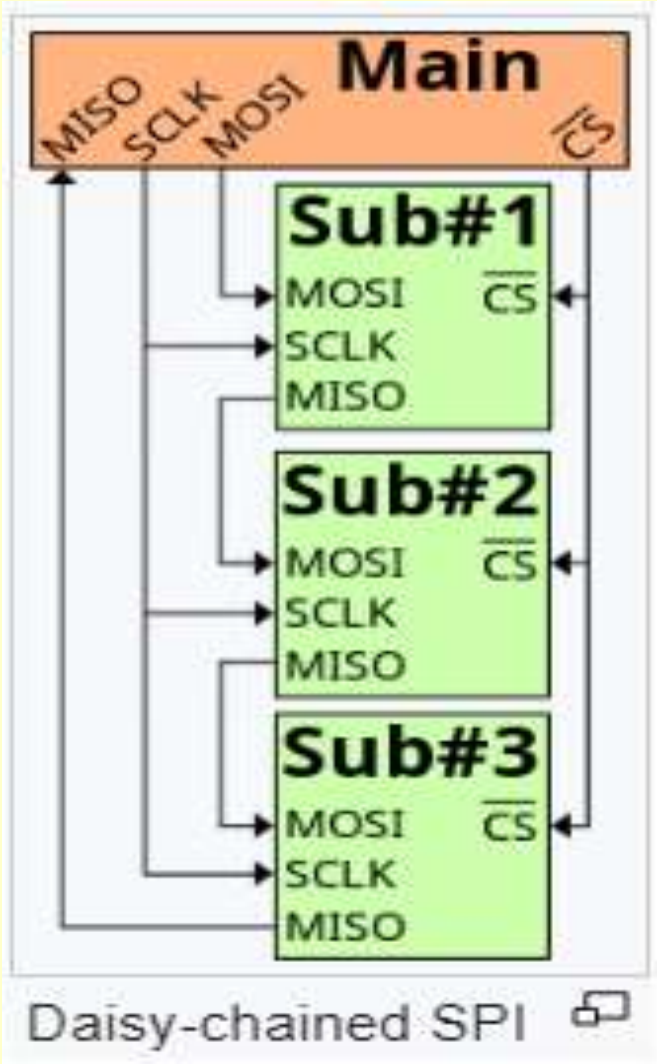
Configurarea Bus-ului SPI



- Opereaza ca **bus** de date sincron, full duplex, cu master unic
- Relatii intre dispozitive Master/slave
- 2 semnale de date :
  - MOSI – master data output, slave data input
  - MISO – master data input, slave data output
- 2 semnale de control:
  - SCLK – clock
  - /SS – slave select (fara adresare)

SPI vs. I²C

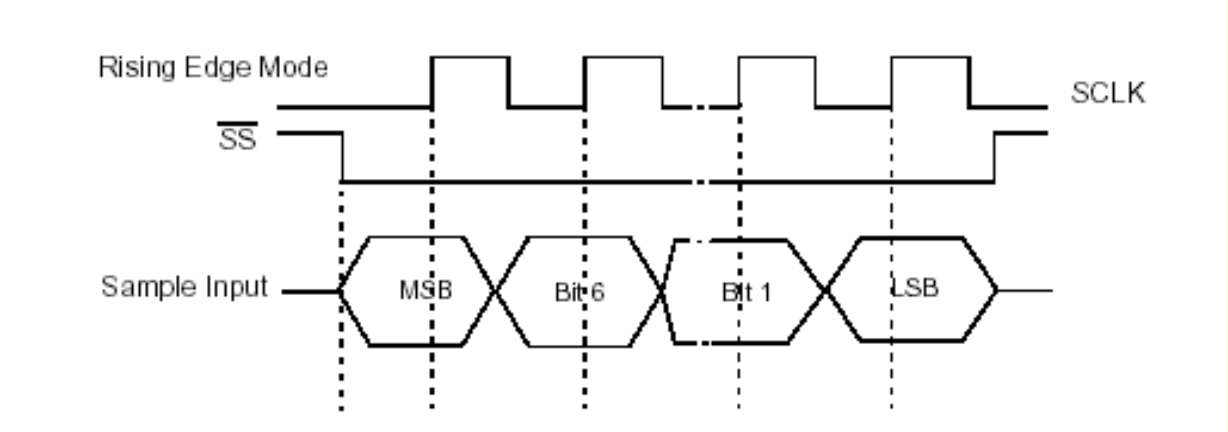




- Ptr. conexiuni point-to-point, SPI este mai simplu si eficient
  - "Less overhead" decat I²C datorita lipsei adresarii, plus SPI este full-duplex
- Ptr. slaves multipli, fiecare slave necesita semnal SS separate (multidrop)
  - Necesita mai mult hardware decat I²C

Protocolul SPI

- interfata SPI defineste numai liniile de comunicatie si frontul activ al clock-ului
- Nu este specificat fluxul de control!
- Nu exista mecanism de acknowledge pentru a confirma receptia datelor
- 2 Parameteri: *Clock Polarity (CPOL) si Clock Phase (CPHA),* determina frontul activ al clock-ului
- Master si slave trebuie sa accepte perechea de parametri ai CLK pentru a comunica



SPI Protocol

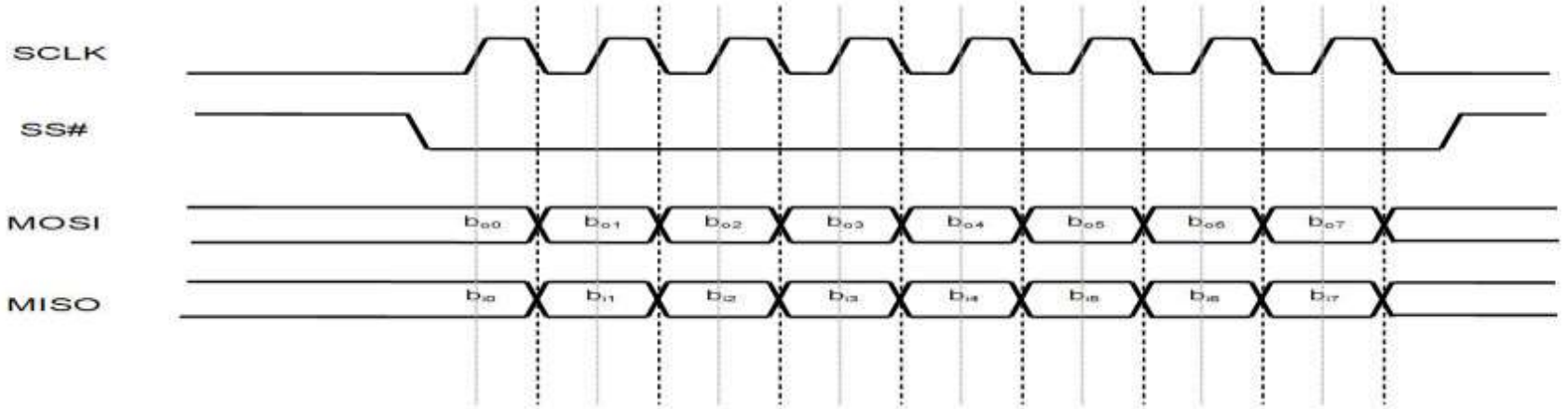
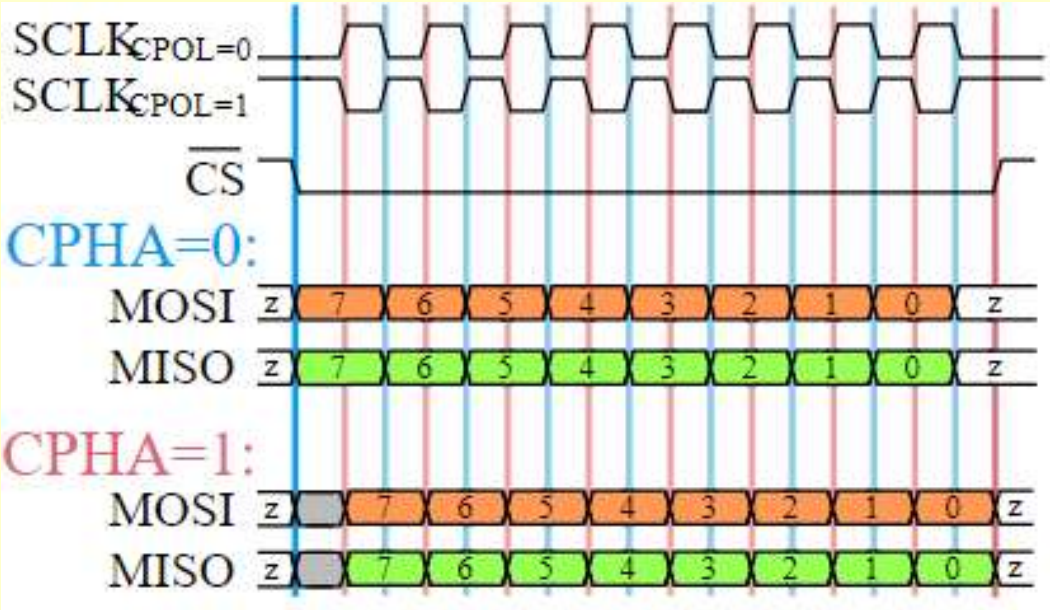
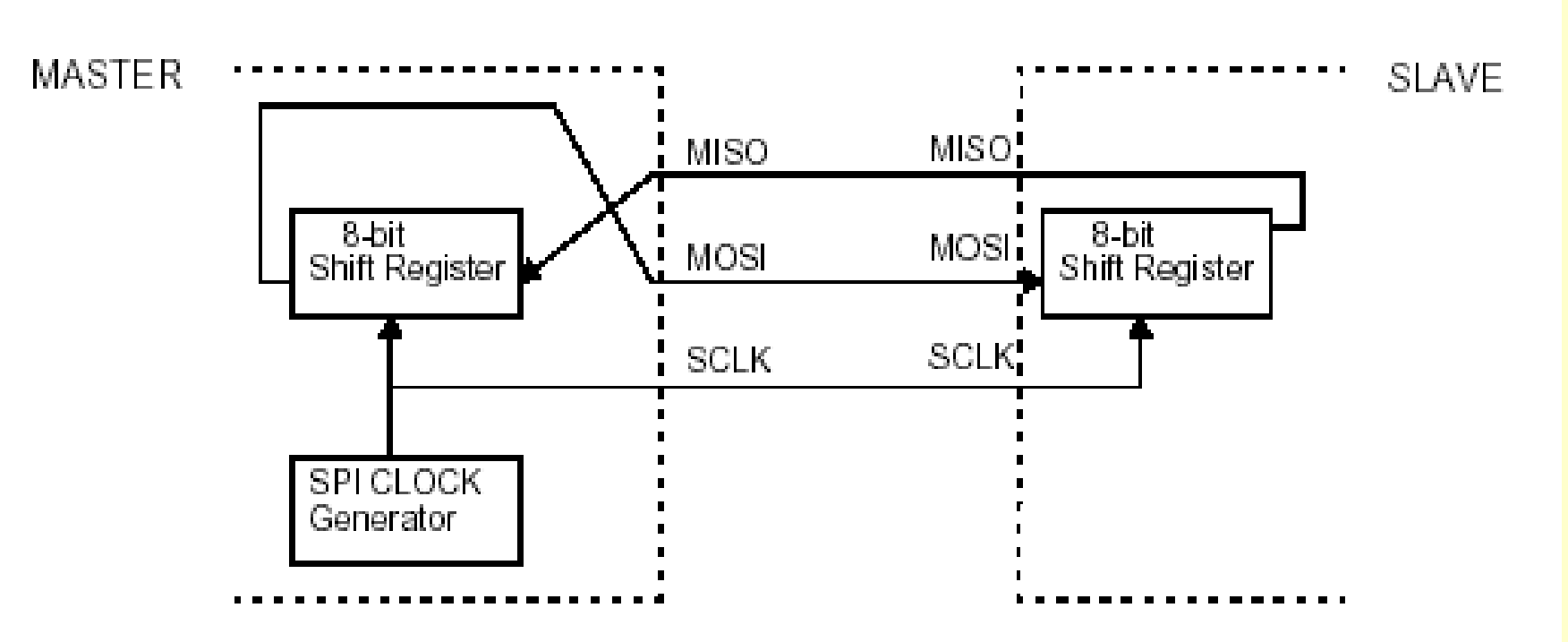


Figure 2 : A simple SPI communication. Data bits on MOSI and MISO toggle on the SCLK falling edge and are sampled on the SCLK rising edge. The SPI mode defines which SCLK edge is used for toggling data and which SCLK edge is used for sampling data.

CPOL	CPHA	Active edge
0	0	Rising
0	1	Falling
1	0	Falling
1	1	Rising



Protocolul SPI (cont.)



- Realizarea hardware-ului este uzual facuta cu un registru de deplasare (shift-register)

SPI avantaje/dezavantaje

Avantaje ale SPI:

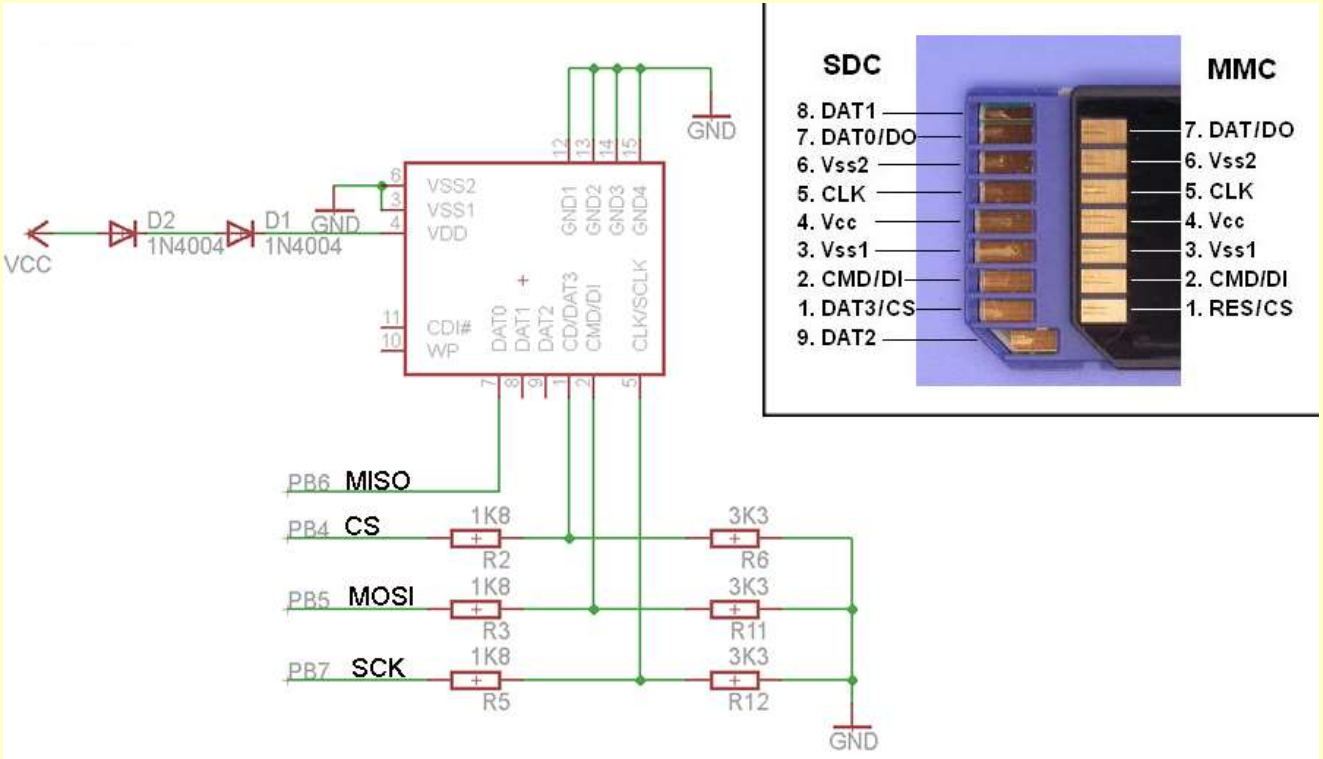
- Comunicatie full-duplex
- Transfer de date semnificativ mai mare
- Bus-ul SPI nu are viteză maximă de ceas
- Flexibilitate pentru bitii transferati (nu exista limitarea la cuvinte de 8-biti, se pot trimite mesaje de orice lungime)
- Se interfateaza usor, usurinta in folosire
- Mai eficient in comunicarea point-to-point

Dezavantaje SPI:

- Foloseste mai multe fire/conexiuni
- Nu are suport pentru adresarea dispozitivelor;
- Este necesar cate un semnal de Slave Select pentru fiecare slave
- Nu exista flow control si slave acknowledgement (master poate“vorbi in gol” fara sa stie).
- Suporta numai *un singur master*

Aplicatii Bus SPI

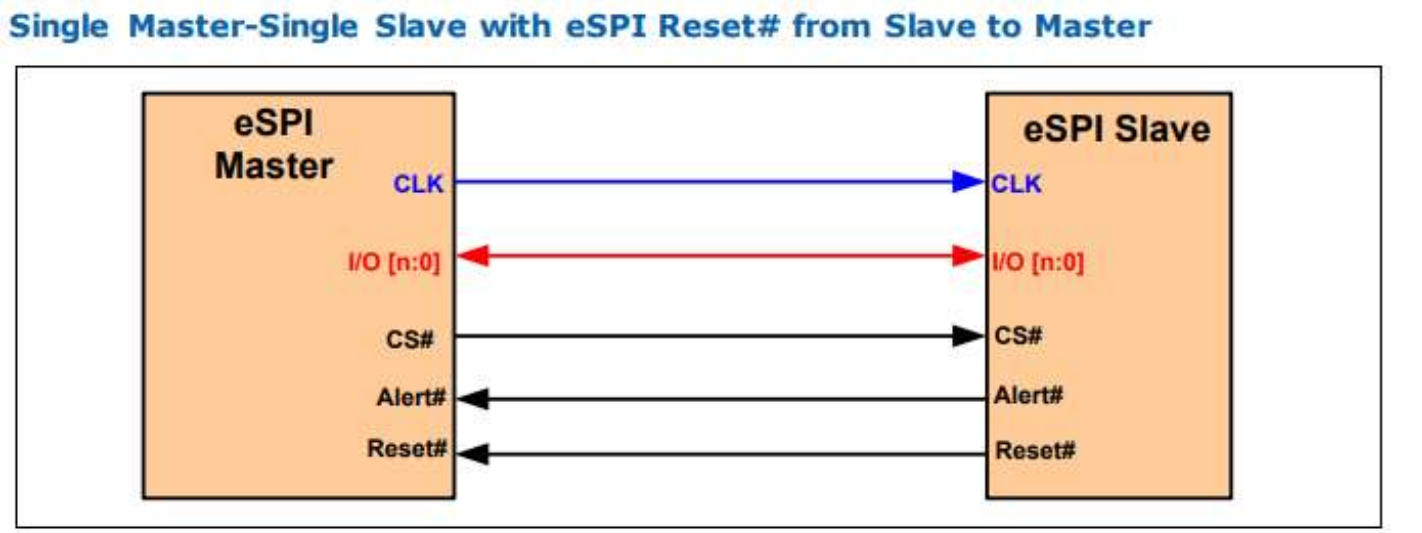
- Sensors: temperature, pressure, ADC, touchscreens, video game controllers
- Control devices: audio codecs, digital potentiometers, DACs
- Camera lenses: Canon EF lens mount
- Communications: Ethernet, USB, USART, CAN, IEEE 802.15.4, IEEE 802.11
- Memory: flash and EEPROMs
- Real-time clocks
- LCDs, sometimes even for managing image data
- Any MMC or SD card (including SDIO variant)
- Shift registers for additional I/O



eSPI (enhanced SPI)

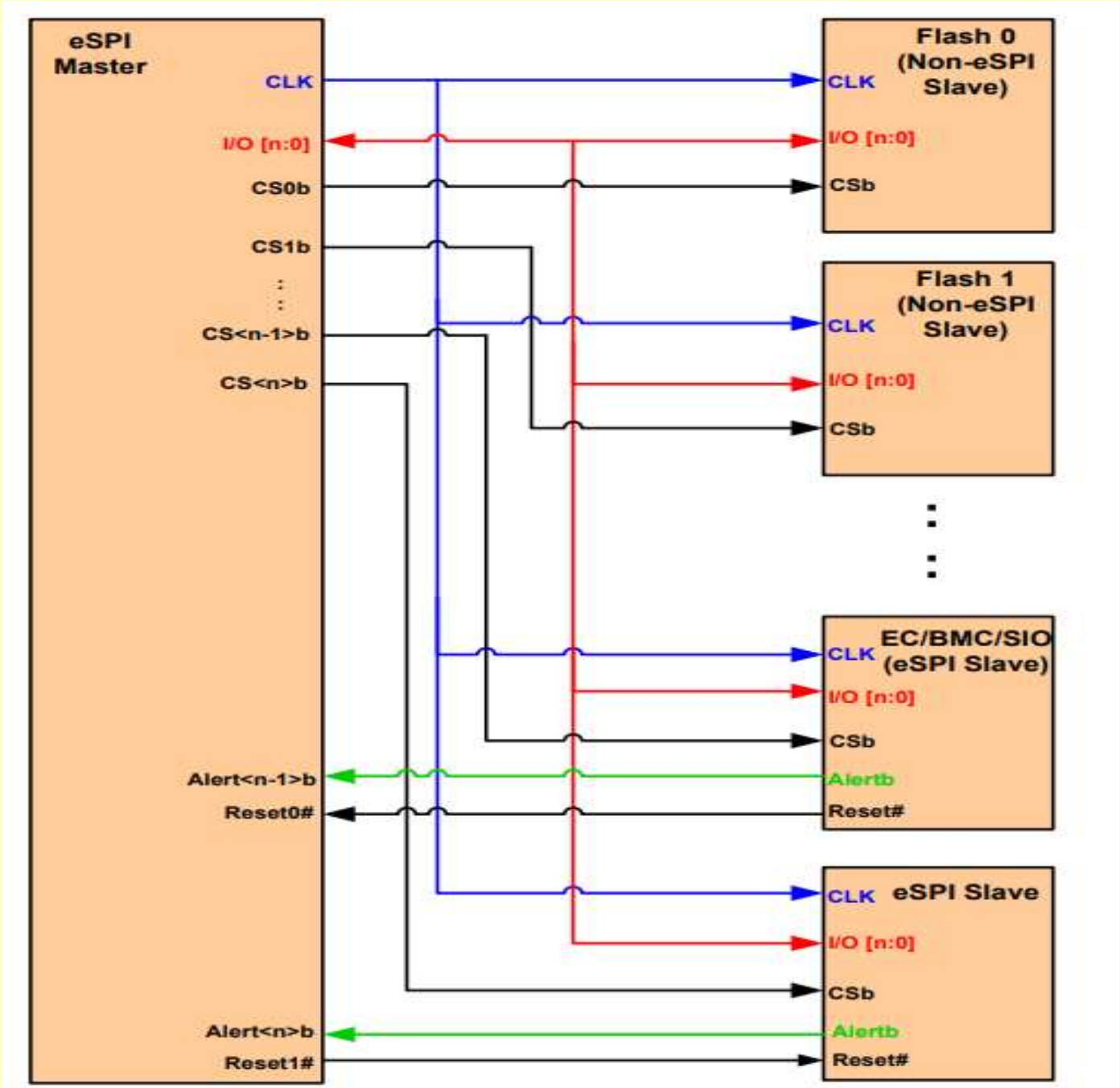
•Acest standard definește un semnal Alert# care este utilizat de un sclav eSPI pentru a solicita servicii de la master. Într-o aplicatie orientata spre performanță sau cu un singur sclav eSPI, fiecare sclav eSPI va avea pinul Alert# conectat la un pin, Alert# de pe magistrala eSPI care este dedicat fiecărui sclav, permițând magistralei eSPI să acorde servicii cu latență redusă, deoarece magistrala eSPI va ști care sclav eSPI are nevoie de servicii și nu va trebui să interogheze toate sclavii pentru a determina ce dispozitiv are nevoie de servicii.

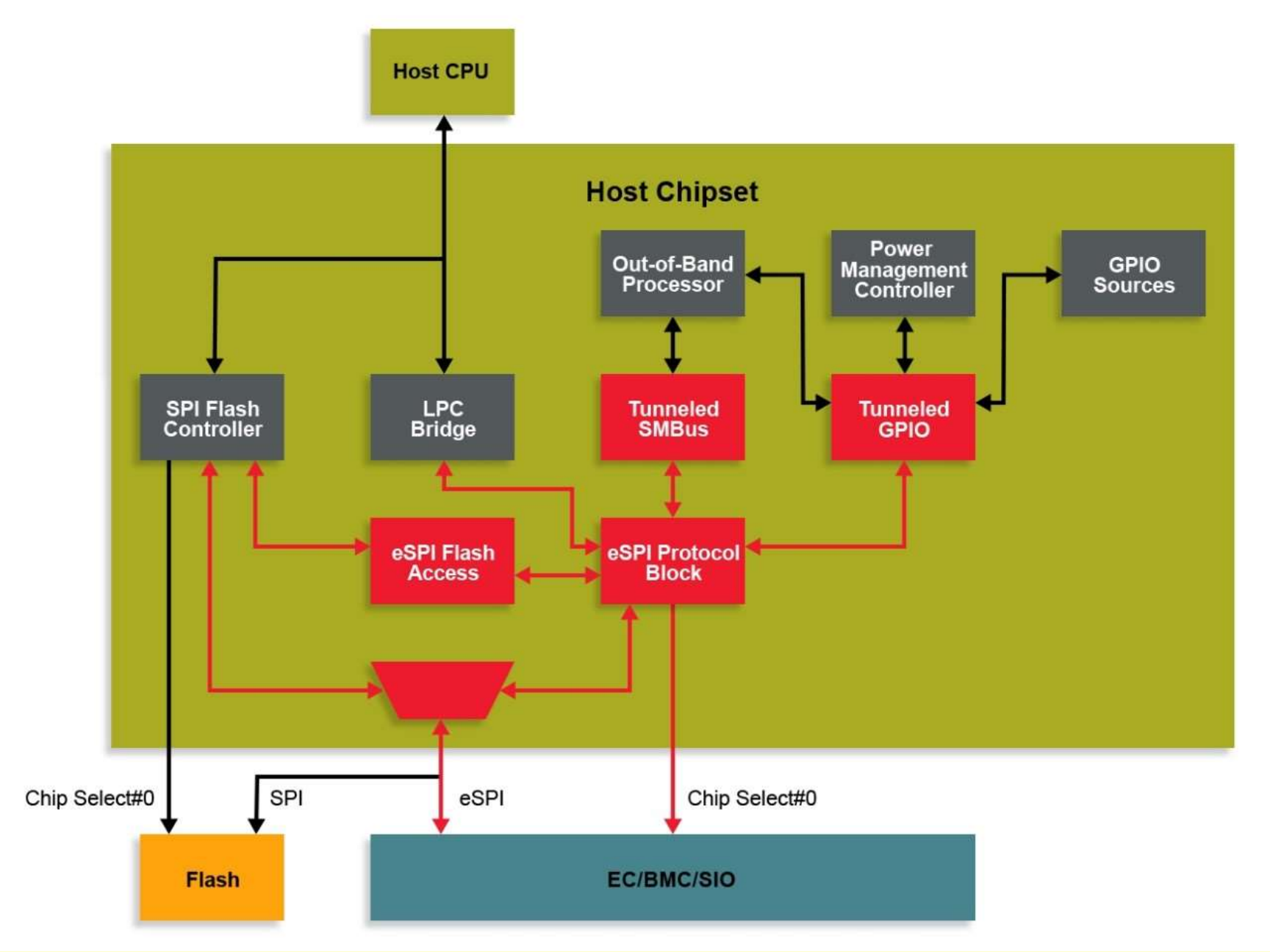
• Într-un proiect economic cu mai mulți sclavi eSPI, toți pinii Alert# ai sclavilor sunt conectați la un singur pin Alert# al masterului eSPI într-o conexiune de tip ***"wired-OR",*** ceea ce va necesita ca masterul să sondeze toți sclavii pentru a determina care dintre ei au nevoie de serviciu, atunci când semnalul Alert# este tras la nivel scăzut de unul sau mai multe periferice care au nevoie de serviciu. Numai după ce toate dispozitivele sunt deservite, semnalul Alert# va fi tras la nivel înalt, deoarece niciunul dintre sclavii eSPI nu mai are nevoie de deservire.



Enhanced Serial Peripheral Interface (eSPI) Interface Base Specification (for Client and Server Platforms) (intel.com)

Single Master-Multiple Slaves with Two eSPI Reset#





- Embedded Controller (EC), Base-board Management Controller (BMC), Super I/O (SIO) controller
- https://www.microchip.com/en-us/solutions/data-centers-and-computing/computing-solutions/technologies/espi

SPI vs I2C

Ambele standarde sunt folosite cu succes pentru comunicatia cu periferice lente ce sunt accesate intermitent (EEPROM-urile si ceasurile de timp real sunt exemple de astfel de device-uri).

- SPI este mai adecvat decat I2C la aplicatiile care folosesc stream-uri de date (spre deosebire de aplicatiile unde se R/W diverse locatii din slave device). Un ex. de aplicatie ce foloseste stream-uri este comunicatia dintre micropocesoare sau DSP-uri

- SPI poate atinge rate de transfer semnificativ mai mari decat I2C - interfetele SPI *putand functiona la zeci de MHz.*

- SPI este eficient mai ales in aplicatii ce ii folosesc capacitatea de a realiza conexiuni full duplex, ca de exemplu comunicarea dintre un “codec” (coder-decoder) si un DSP, ce presupune trimiterea de date in ambele directii.

- Datorita faptului ca nu exista suport pentru adresarea device-urilor, SPI necesita mai mult efort si mai multe resurse hardware decat I2C, cand avem mai multi slaves.

- SPI este mai simplu si mai eficient in aplicatii point-to-point (single master, single slave) din acelasi motive: lipsa adresarii dispozitivelor presupune mai putine date transferate.

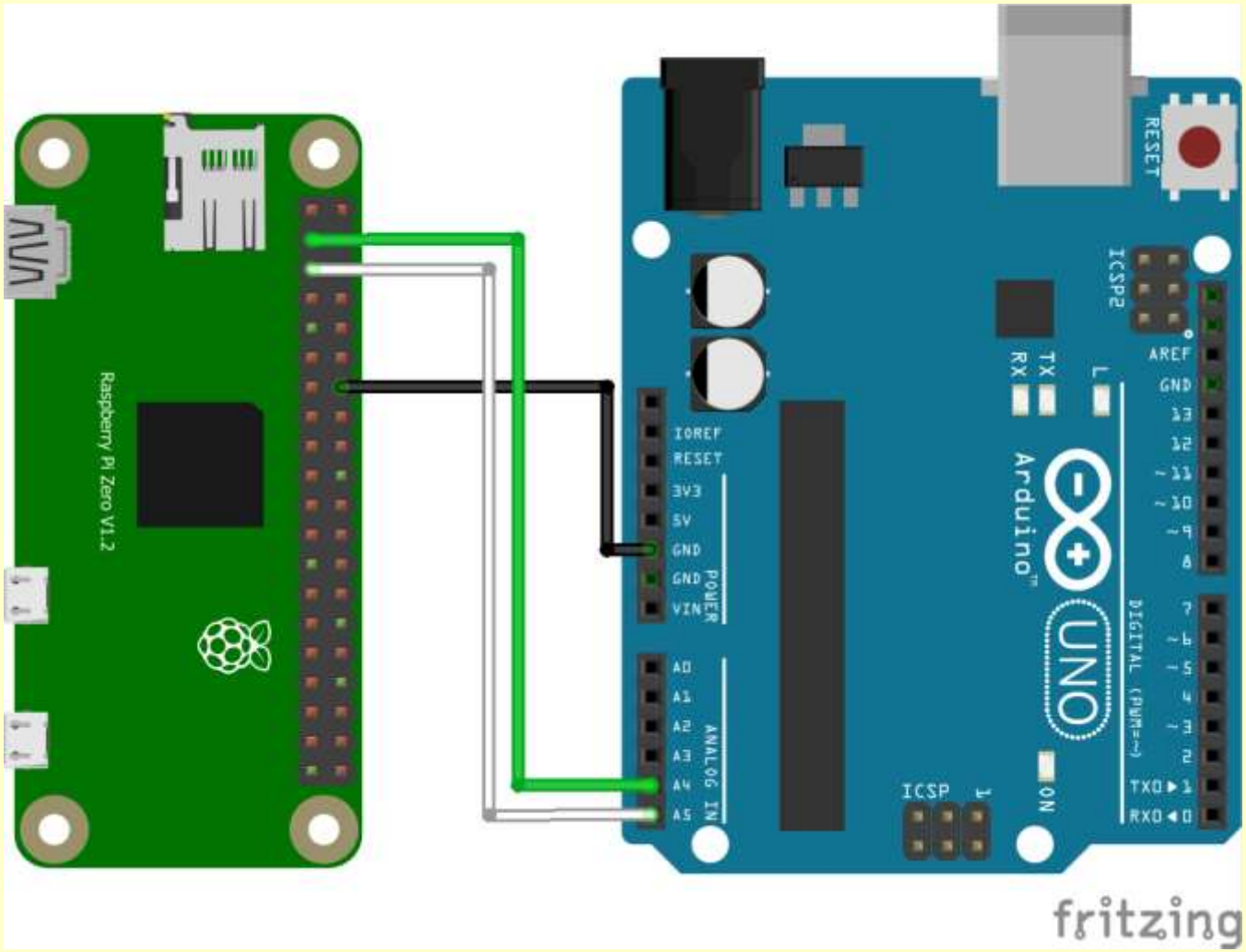
Concluzii

- I^2C si SPI ofera suport bun ptr. comunicatii cu periferice lente accesate intermitent: EEPROMs, real-time clocks, …

- I^2C se adapteaza mai usor la dispozitive multiple pe un singur bus.

- SPI este mai rapid, dar apar complicatii cand sunt mai multi slaves in aplicatie

I²C Arduino



Functions

- begin()
- requestFrom()
- beginTransmission()
- endTransmission()
- write()
- available()
- read()
- SetClock()
- onReceive()
- onRequest()

Referinte

I²C:

- http://www.esacademy.com/faq/i2c/index.htm
- http://www.i2cchip.com/

SPI:

- MC68HC11 manual
- http://links.epanorama.net/links/serialbus.html

TEME. **1.** Studiati UNI/O® bus: a low-cost, easy-to-implement solution requiring only a single I/O signal for communication. (Microchip)

2. Studiati utilitatea eSPI in PC.

https://www.intel.com/content/dam/support/us/en/documents/software/chipset-software/327432-004_espi_base_specification_rev1.0_cb.pdf

3. Studiati modul de implementare "bit banging" al interfetelor seriale.

https://circuitdigest.com/article/introduction-to-bit-banging-spi-communication-in-arduino-via-bit-banging

4. Studiati si gasiti diferentele intre I^2C Bus si SM Bus.

SMBus este System Management Bus definit de Intel® in 1995. Este folosit in PC si servers ptr. low-speed system management communications

	I^2C	SMBus
Timeout	No	Yes
Minimum Clock Speed	DC	10kHz
Maximum Clock Speed	100kHz (400kHz and 2MHz also available)	100kHz
V_{HIGH}	0.7 × V_{DD}, 3.0V Fixed	2.1V
V_{LOW}	0.3 × V_{DD}, 1.5V Fixed	0.8V
Max I	3mA	350µA
Clock Nomenclature	SCL	SMBCLK
Data Nomenclature	SDA	SMBDAT
General Call	Yes	Yes
Alert#	No	Yes

Table I2C vs SMBus **Comparisons**

5. Ex. I2C Master Code

This example shows how to implement a software I2C master, including clock stretching. It is written in C for the PIC processor but should be applicable to most processors with minor changes to the I/O pin definitions. It is suitable for controlling all of our I2C based robot modules. Since the SCL and SDA lines are open drain type, we use the tristate control register to control the output, keeping the output register low. The port pins still need to be read though, so they're defined as SCL_IN and SDA_IN. This definition and the initialization is probably all you'll need to change for a different processor.

```
#define SCL     TRISB4 // I2C bus
#define SDA     TRISB1 //
#define SCL_IN  RB4    //
#define SDA_IN  RB1    //
```

To initialize the ports set the output resisters to 0 and the tristate registers to 1 which disables the outputs and allows them to be pulled high by the resistors.

```
SDA = SCL = 1;
SCL_IN = SDA_IN = 0;
```

We use a small delay routine between SDA and SCL changes to give a clear sequence on the I2C bus. This is nothing more than a subroutine call and return.

```
void i2c_dly(void)
{
}
```

The following 4 functions provide the primitive start, stop, read and write sequences. All I2C transactions can be built up from these.

```
void i2c_start(void)
{
  SDA = 1;            // i2c start bit sequence
  i2c_dly();
  SCL = 1;
  i2c_dly();
  SDA = 0;
  i2c_dly();
  SCL = 0;
  i2c_dly();
}
void i2c_stop(void)
{
  SDA = 0;            // i2c stop bit sequence
  i2c_dly();
  SCL = 1;
  i2c_dly();
  SDA = 1;
  i2c_dly();
}
```

```
unsigned char i2c_rx(char ack)
                        {
char x, d=0;
  SDA = 1;
  for(x=0; x<8; x++) {
    d <<= 1;
    do {
      SCL = 1;
                         }
    while(SCL_IN==0);    // wait for any SCL clock stretching
    i2c_dly();
    if(SDA_IN) d |= 1;
    SCL = 0;
                        }
  if(ack) SDA = 0;
  else SDA = 1;
  SCL = 1;
  i2c_dly();            // send (N)ACK bit
  SCL = 0;
  SDA = 1;
  return d;
                        }
bit i2c_tx(unsigned char d)
                        {
char x;
static bit b;
  for(x=8; x; x--) {
    if(d&0x80) SDA = 1;
    else SDA = 0;
    SCL = 1;
    d <<= 1;
    SCL = 0;
                         }
  SDA = 1;
  SCL = 1;
  i2c_dly();
  b = SDA_IN;          // possible ACK bit
  SCL = 0;
  return b;
                        }
```

```
//The 4 primitive functions above can easily be put together to form complete I2C
//transactions. Here's and example to start an SRF08 ranging in cm:
i2c_start();             // send start sequence
i2c_tx(0xE0);              // SRF08 I2C address with R/W bit clear
i2c_tx(0x00);              // SRF08 command register address
i2c_tx(0x51);              // command to start ranging in cm
i2c_stop();              // send stop sequence
//Now after waiting 65mS for the ranging to complete (I've left that to you) the following
//example shows how to read the light sensor value from register 1 and the range result
//from registers 2 & 3.
i2c_start();             // send start sequence
i2c_tx(0xE0);              // SRF08 I2C address with R/W bit clear
i2c_tx(0x01);              // SRF08 light sensor register address
i2c_start();             // send a restart sequence
i2c_tx(0xE1);              // SRF08 I2C address with R/W bit set
lightsensor = i2c_rx(1);  // get light sensor and send acknowledge. Internal register
//address will increment automatically.
rangehigh = i2c_rx(1);    // get the high byte of the range and send acknowledge.
rangelow = i2c_rx(0);     // get low byte of the range - note we don't acknowledge the last
                                                                                    //byte.
i2c_stop();              // send stop sequence
```

Interfata I²S

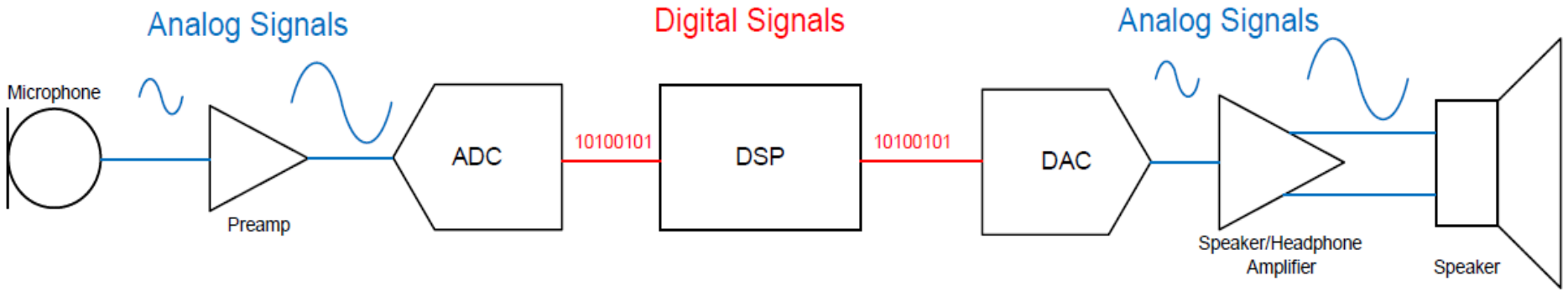


Figure 1. Traditional Audio Signal Chain

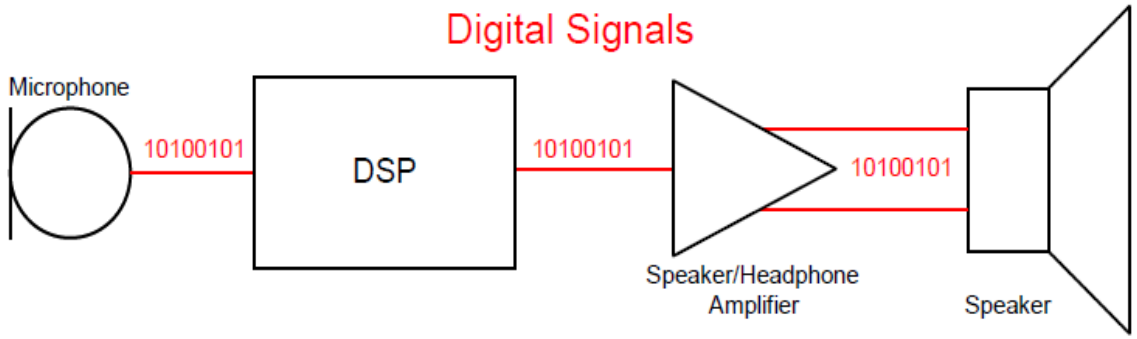
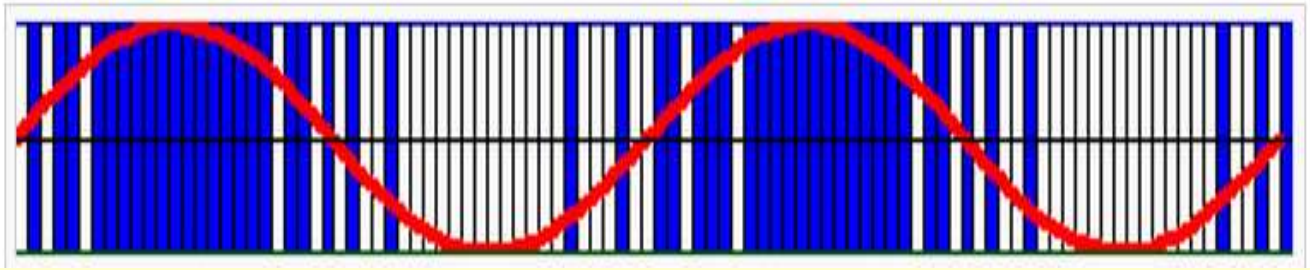


Figure 2. Fully Digital Audio Signal Chain

Pulse Density Modulation

- Într-un semnal PDM, valorile amplitudinii specifice nu sunt codificate în valori diferite, așa cum sunt în PCM.
- Modulația în lățime a impulsurilor (PWM) este cazul special al PDM, unde toate impulsurile corespunzătoare unui eșantion sunt contigue în semnalul digital.
- Un sir de biti ***PDM este codificat*** plecand de la un semnal analogic prin *procesul de modulare delta-sigma.*
- ***Procesul de decodificare*** al unui semnal PDM într-unul analogic este simplu: *trebuie să se treacă semnalul PDM printr-un filtru trece-jos*. Aceasta funcționează deoarece funcția unui filtru trece-jos este în esență, să medieze semnalul.

0101101111111111111101101010010000000000000100010011011101111111111111011010100100000000000000100101



$$x[n] = -A(-1)^{a[n]}$$

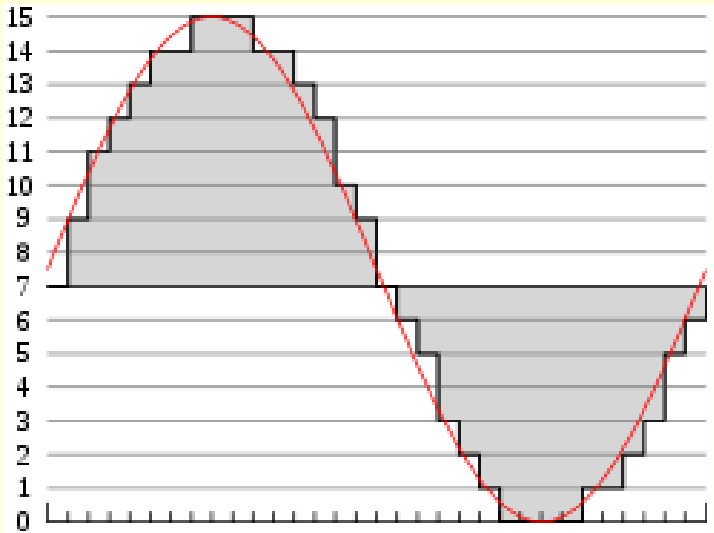
I^2S

- I^2S lansat de Philips Semiconductor - NXP Semiconductors, (1986)
- Protocol: Serial, sincron
- I^2S (Inter-IC Sound), este un standard de bus serial folosit ptr. conectarea dispozitivelor/circuitelor audio digitale
- Este utilizat sa comunice datele PCM audio intre CI in interiorul dispozitivelor electronice.
- Busul I^2S separa clock-ul si semnalele seriale de date, rezultand receptoare mai simple decat cele cerute ptr. comunicatiile asincrone

Digital Audio

- **PCM**
  - Este forma standard a sunetului digital în computere, CD, telefonie digitală și alte aplicații audio digitale. Într-un flux PCM, amplitudinea semnalului analogic este prelevată în mod regulat, la intervale uniforme și fiecare esantion este cuantizat la cea mai apropiată valoare într-un interval de pași numerici.

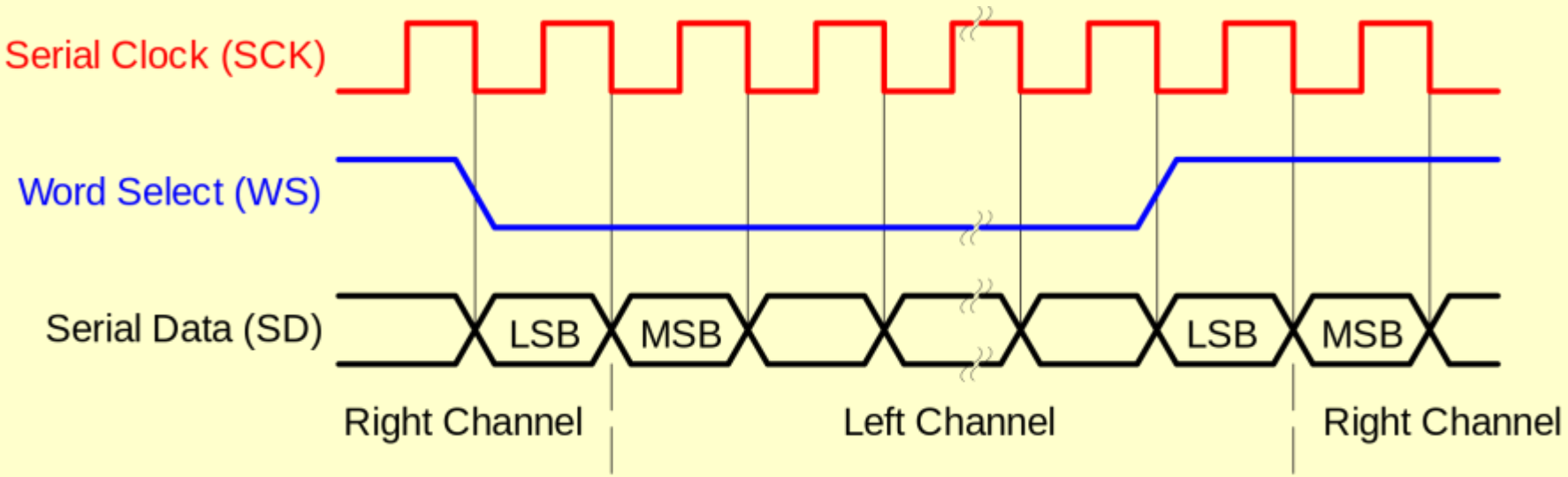
  - Fluxurile PCM au două caracteristici de bază care determină fidelitatea lor față de semnalul analogic original: *rata de eșantionare,* numărul de eșantioane prelevate pe secundă și *rezolutia CAN,* care determină numărul de valori digitale posibile pe care le poate lua fiecare esantion.



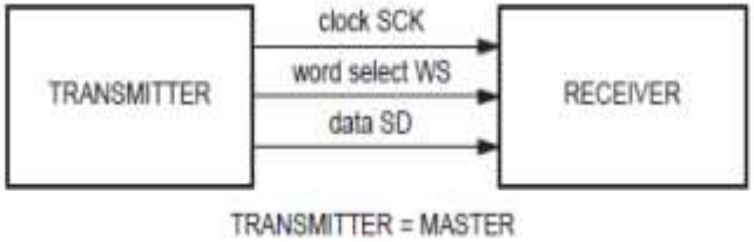
http://shepherdingelectrons.blogspot.com/2018/09/esp8266-minimum-i2s-code.html

https://www.arduino.cc/en/Tutorial/I2SSimpleTone

- **I²S este o**ptimizat ptr. transmisia de date audio digitale
  - Are 3-linii seriale
- SD: 2 canale de date multiplexate in timp
- WS: Word Select (0=left channel, 1 = right channel)
- SCK: Clock
    - Masterul genereaza SCK si WS
    - Masterul = transmitator sau controller separat
    - SCK sincronizeaza transmitatorul si receptorul
- Formatul datelor seriale sunt in complement fata de 2, MSB trimis primul

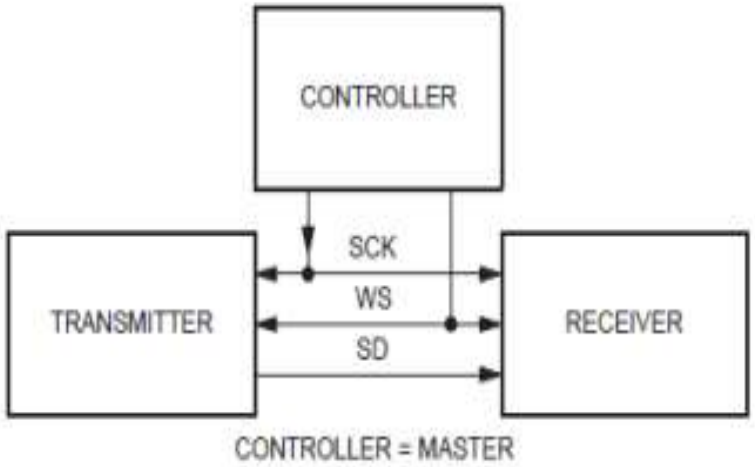


I²S Configurare / Timing

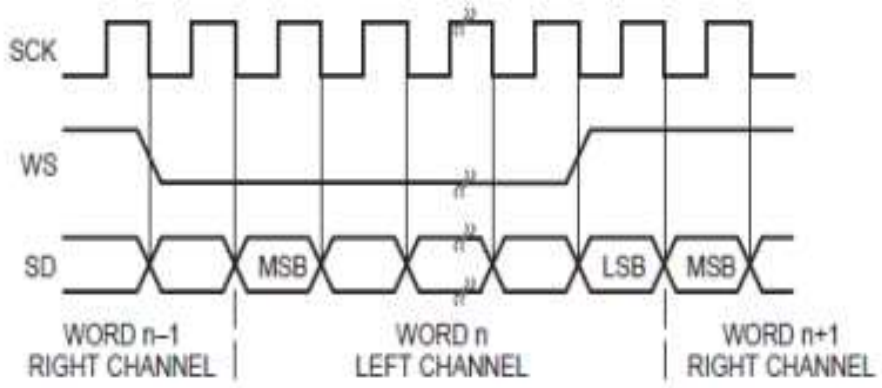


TRANSMITTER = MASTER

RECEIVER = MASTER

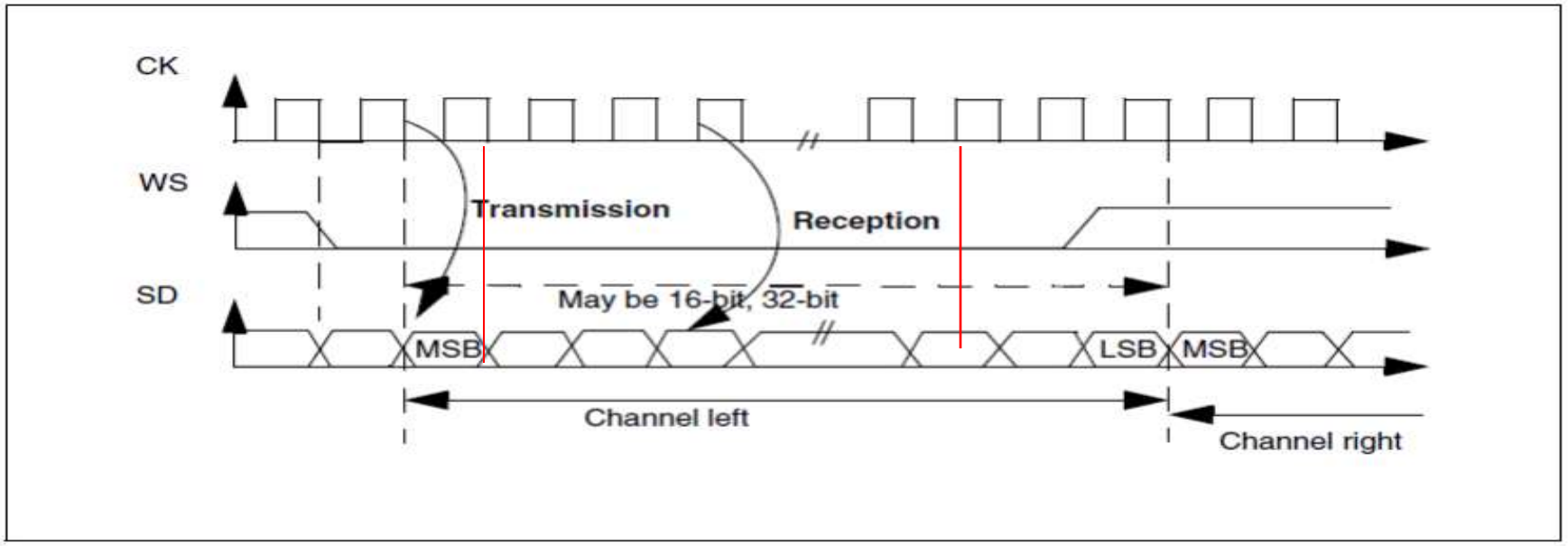


CONTROLLER = MASTER



SN00119

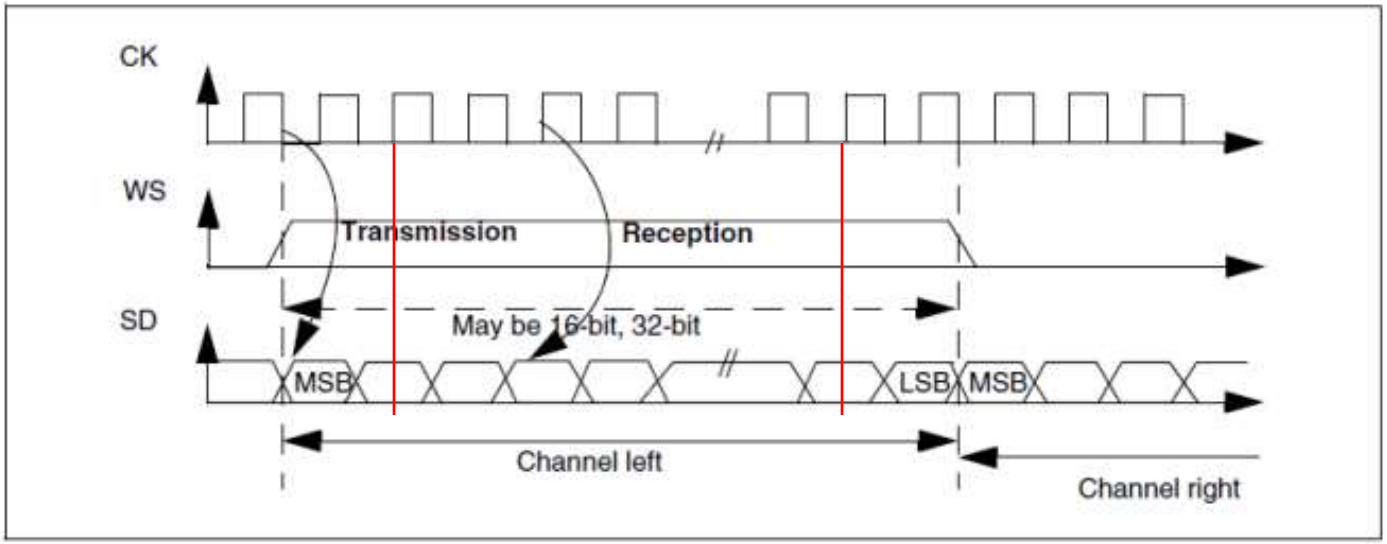
I²S Philips protocol (CPOL = 0)



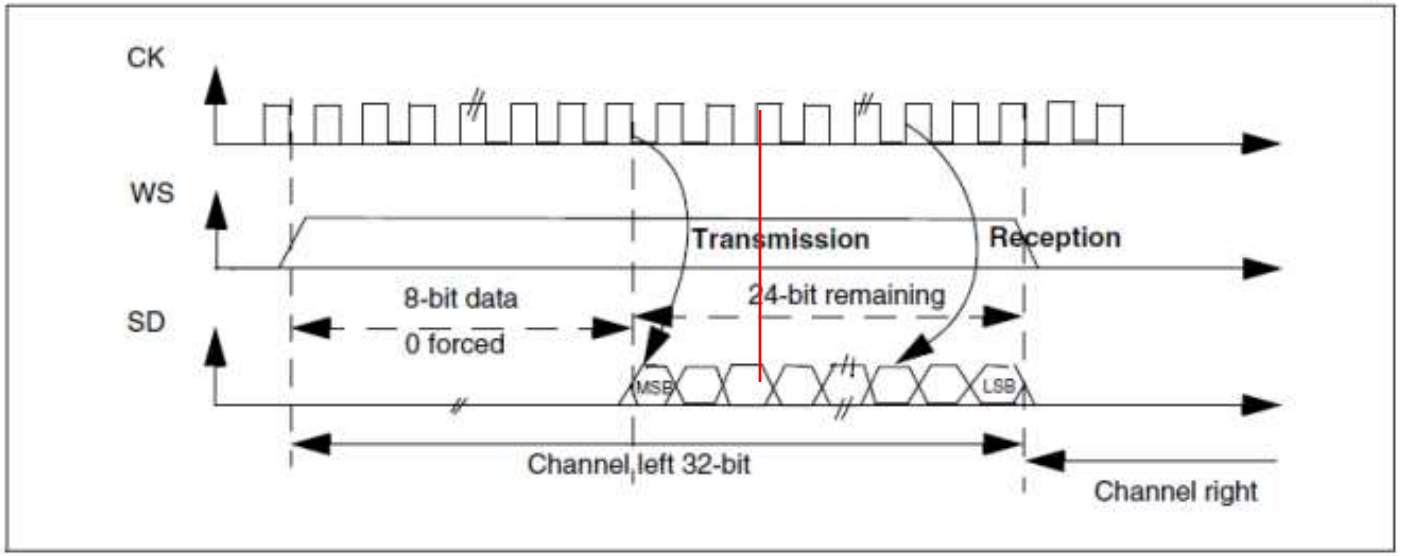
- WS latched on falling edge of CK
- Xmit: latch data on falling edge of CK
- Rcv: read on rising edge of CK

- Un flux de date I²S poate transporta unul/două canale de date cu o rată tipică a ceasului de bit între 512 kHz ptr. o frecvența de eșantionare de 8 kHz și 12.288 MHz, pentru o rată de eșantionare de 192 kHz.
- Lungimea cuvântului de date este adesea de 16, 24 sau 32 de biți.

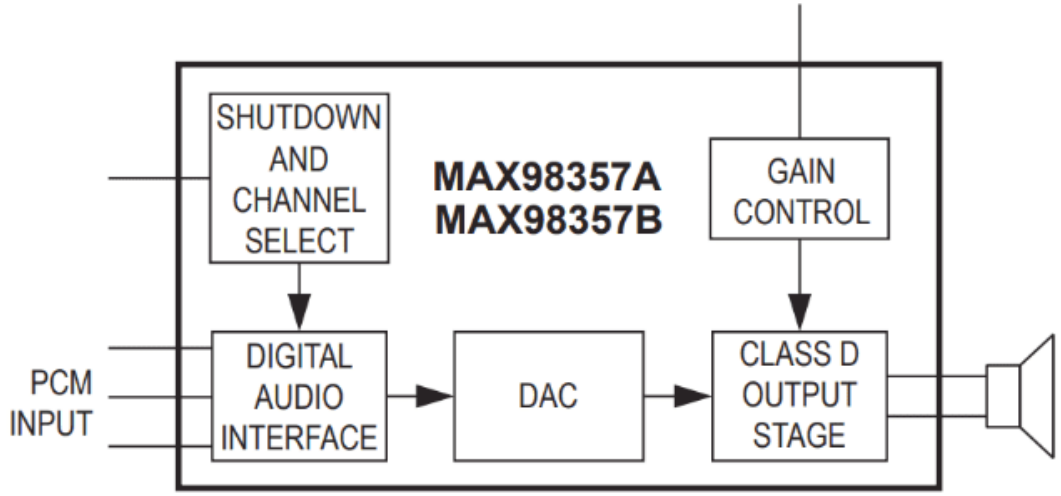
I²S LSB justified standard



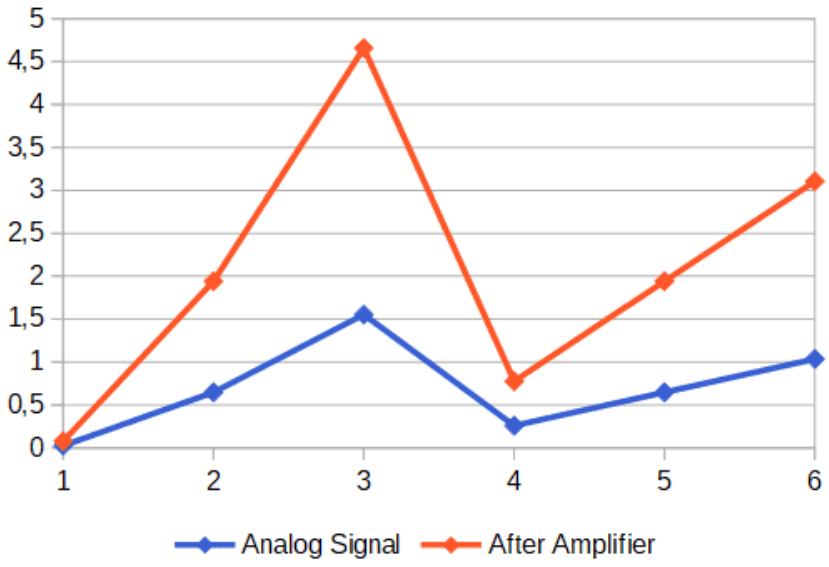
16/32 data in
16/32 frame



24 data in
32 frame



Digital Bit	Digital DEC	Analog Voltage
0 0 0 0 1 0 1 0	2	0.026V
0 0 1 1 0 0 1 0	50	0.647V
0 1 1 1 1 0 0 0	120	1.553V
0 0 0 1 0 1 0 0	20	0.259V
0 0 1 1 0 0 1 0	50	0.647V
0 1 0 1 0 0 0 0	80	1.035V



https://diyi0t.com/i2s-sound-tutorial-for-esp32/
https://github.com/earlephilhower/ESP8266Audio