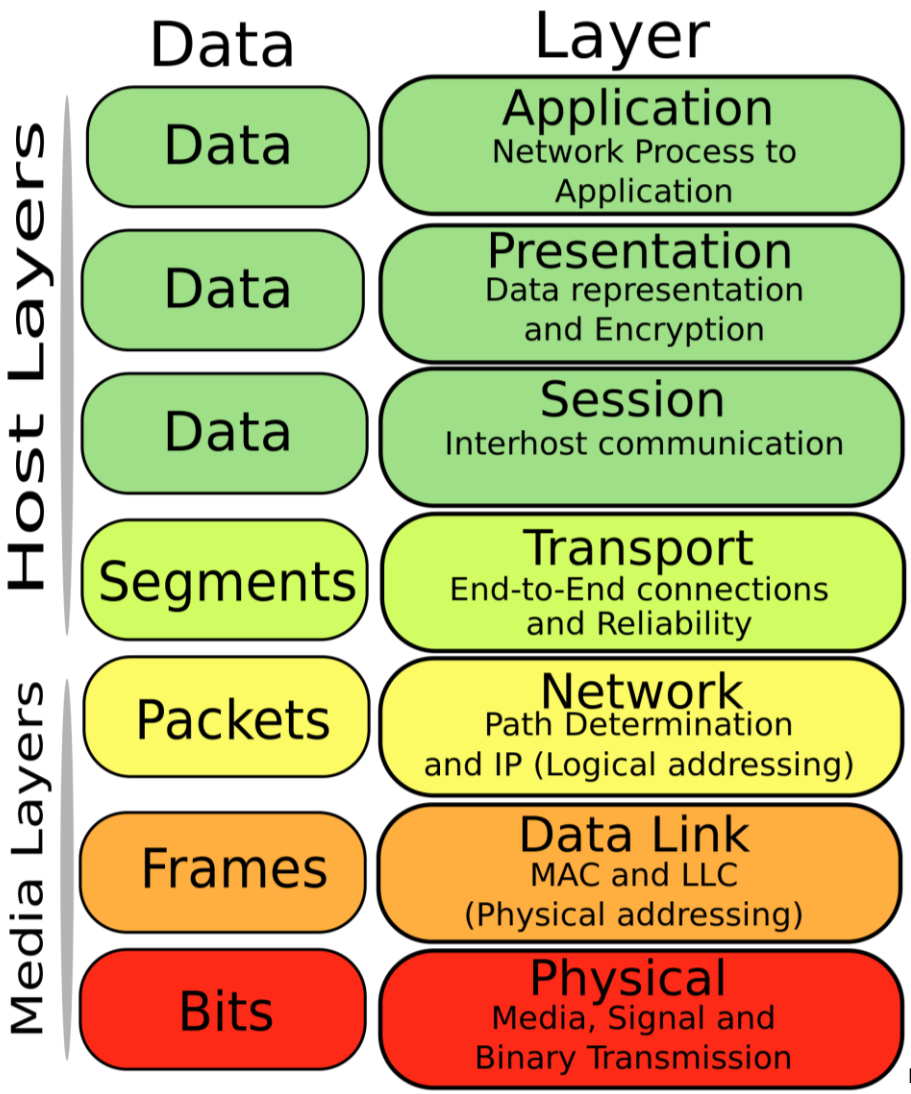
Funció RX d'una trama "*Status*" als Mòduls del Robot

// Aquest exemple no és complert, en principi RxPacket() torna una estructura*"Status packet*" que bàsicament consisteix en un array amb "Status Packet"+ un byte indicant si hi ha un TimeOut. Això s'ha fet així perquè en C no es pot posar un array com paràmetre de tornada.
Per altre banda, la part mostrada només llegeix els primers 4 bytes del *status packet*. Això és perquè el quart byte indica precisament quants bytes queden per llegir, el que vol dir que s'ha de fer un altre bucle "for" semblant per llegir els bytes que falten....
Evidentment, es podria fer d'altres maneres, per exemple enviant com paràmetre el número de bytes a llegir...

```
struct RxReturn RxPacket(void)
{
      struct RxReturn respuesta;
      byte bCount, bLenght, bChecksum;
      byte Rx_time_out=0;
      DataDirection_Rx();      //Ponemos la linea half duplex en Rx
      Activa_TimerA1_TimeOut();
      for(bCount = 0; bCount < 4; bCount++) //bRxPacketLength; bCount++)
      {
        Reset_Timeout();
        Byte_Recibido=No;    //No_se_ha_recibido_Byte();
        while (!Byte_Recibido) //Se_ha_recibido_Byte())
        {
          Rx_time_out=TimeOut(1000);        // tiempo en decenas de microsegundos (ara 10ms)
          if (Rx_time_out)break;//sale del while
         }
         if (Rx_time_out)break;  //sale del for si ha habido Timeout
         //Si no, es que todo ha ido bien, y leemos un dato:
         respuesta.StatusPacket[bCount] = DatoLeido_UART; //Get_Byte_Leido_UART();
      }//fin del for
      if (!Rx_time_out)
     // Continua llegint la resta de bytes del Status Packet
```

Afegiu un codi per garantir no escriure en les primeres posicions!

OSI Models


Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data representation and Encryption
Data
Session
Interhost communication
Segments
Transport
End-to-End connections and Reliability
Packets
Network
Path Determination and IP (Logical addressing)
Frames
Data Link
MAC and LLC (Physical addressing)
Bits
Physical
Media, Signal and Binary Transmission

UNIVERSITAT DE BARCELONA

Motors – Endless turn

If both values for the CW Angle Limit and the CCW Angle Limit are set to 0, an Endless Turn mode can be implemented by setting the Goal Speed. This feature can be used for implementing a **continuously rotating wheel**.

6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
10(0x0A)	(Reserved)		0(0x00)

Goal Speed Setting

BIT	15~11	10	9	8	7	6	5	4	3	2	1	0
Value	0	Turn Direction	Speed Value									

Turn Direction = 0 : CCW Direction Turn, Load Direction = 1: CW Direction Turn

32(0X20)	Moving Speed(L)	RD,WR	0
33(0X21)	Moving Speed(H)	RD,WR	0
34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value

Mov_speed_L = speed & 0xFF
Mov_speed_H = ((direction << 2) & 0x04) | ((speed >> 8) & 0x03)

Motors

Funcions per moure el robot. Per exemple endavant, endarrera, endavant dreta, endavant esquerra, gir dreta, gir esquerra

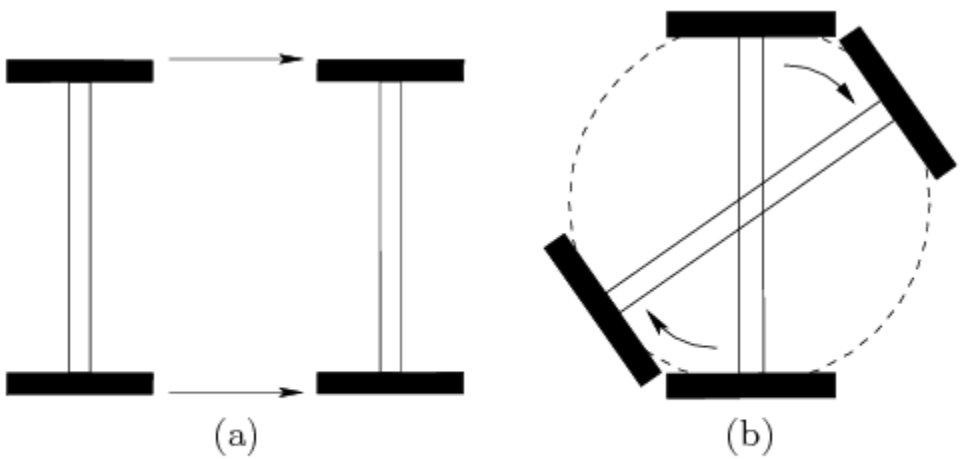


Figure 13.3: (a) Pure translation occurs when both wheels move at the same angular velocity; (b) pure rotation occurs when the wheels move at opposite velocities.

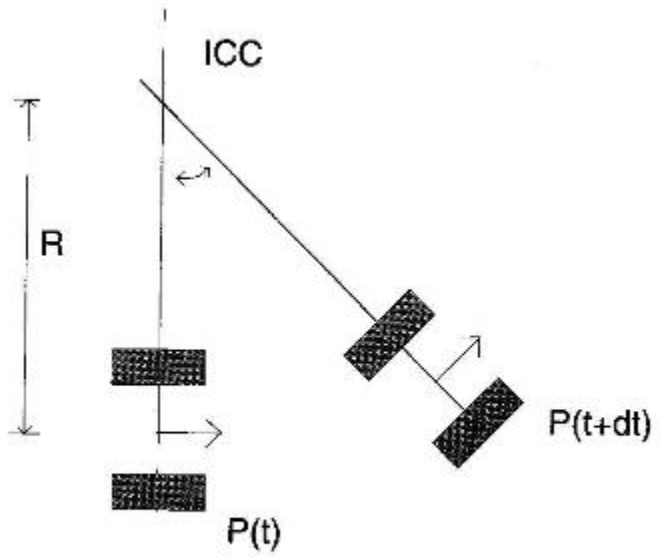
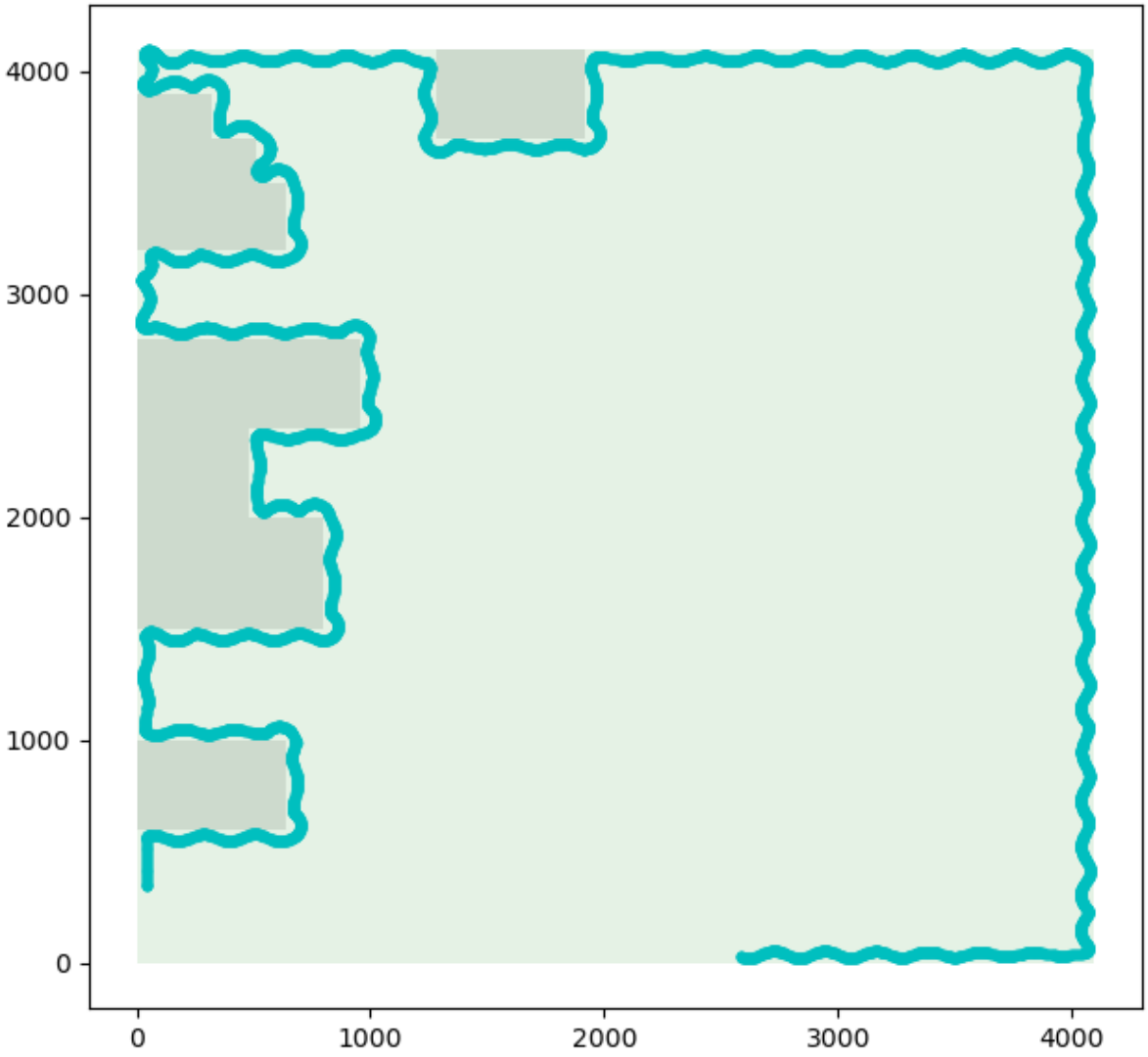


Figure 2: Forward kinematics for differential robot

Tingueu en compte que els motors són els mateixos i per tant el sentit de gir de les rodes està invertit entre elles.

Sensors – Registres

- **Infrared Sensor Data (Left/Center/Right)**
- Luminosity (Left/Center/Right)
- Sound Detected Count
- Buzzer


4000
3000
2000
1000
0
0
1000
2000
3000
4000

Idealitzacions del simulador

- El robot és puntual
- El valor del sensor d'infraroig de distància no te error i dona directament el valor de distància (el infraroig dona el valor de llum reflectida i per tant va al inversa).
- L'habitació per on és mou el robot fa 4095 x 4095 mm i te una resolució de 1 mm.

UNIVERSITAT DE BARCELONA

Protocols de comunicació

David Roma (droma@ub.edu)

Protocols de comunicació

Podem distingir entre:
- **Sèrie**: Les dades s'envien bit a bit. Exemples: UART, USB, ethernet
- **Paral·lel**: DDR, PCI

Els paral·lels és solen fer servir per molt curta distància i altes velocitats (> GB/s).
Els sèrie per comunicacions a mitja o llarga distància i velocitat per més baixes.
Els paral·lels sempre necessitaran més línies (fils, cablejat) entre els dispositius, per això no són adequats a llargues distàncies degut el cost associat.

En els dos casos també podem distingir:
- **Half-dúplex**: només és pot enviar o rebre, mai les dos coses a la vegada. Exemples: UART dynamixel, USB, DDR.
- **Full-dúplex**: és pot enviar i rebre a la vegada. Exemples: UART, PCI

Un full-dúplex te el potencial de transmetre més dades a la vegada, però pot tenir altres problemes com la gestió de quan s'originen, sincronisme entre lectura i la seva resposta, etc I en canvi sempre comportarà un increment del nombre de línies.

Protocols de comunicació

I també, segons el clock:

- Asíncron: no hi ha clock compartit. La sincronització és per temps, senyals o codificació de la linia. Exemple: UART, USB
- Síncron: una de les senyals és el clock. Exemple: ethernet, PCI

Els síncrons solen permetre velocitats més elevades, però incrementen el nombre de línies habitualment. La majoria dels paral·lels, donat que ja tenen un nombre elevat de línies involucrades, solen ser síncrons.

Per últim, segons la topologia:

- Punt-a-punt: Només hi ha dos dispositius connectats a la vegada. En cas que necessitem més, fa falta un switch/hub o similar. Exemple: UART, USB, ethernet
- Multi-punt: Podem tenir múltiples dispositius connectats. Important de distingir de multi-master o similar. Exemples: PCI, i2c, CAN

UNIVERSITAT DE BARCELONA

Protocols més usats en arquitectures encastades

Protocol	#Fils	Velocitat max	Clock	Direcció	Definició?	Topologia
CAN	2	~ 5 Mbps	No	Half-duplex	Completa	Missatge
I^2C	3	~3.4 Mbps	Si	Half-duplex	Completa	Multi-master
UART	3 - 5	~ 15 Mbps	No	Full-duplex	Trama	Punt-a-punt
SPI	5	~ 150 Mbps	Si	Full-duplex	Bit	Multi-esclau

Tots són sèrie, per reduir el nombre de fils.

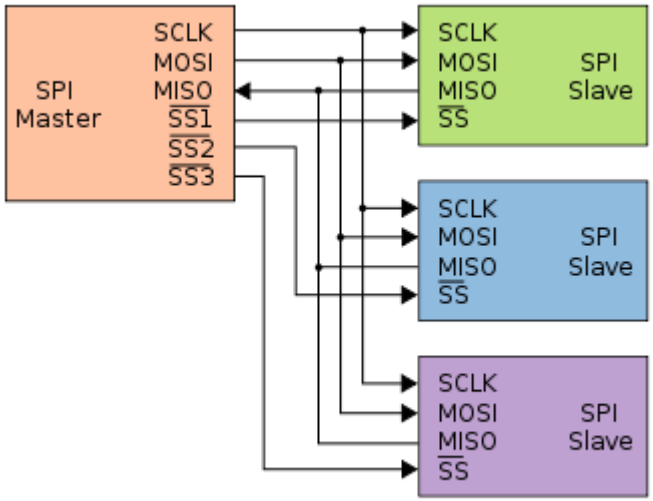
SPI

Velocitat habituals desenes de Mbps.

Línies del bus:

- Serial Clock Line (SCL): Senyal de rellotge.
- Master Output Slave Input (MOSI): Senyal de dades del master al esclau.
- Master Input Slave Output (MISO): Senyal de dades del esclau al master.
- Chip select o slave select (nCS, NSS): Senyal d'inhabilitació del esclau.
- Ground: massa, referencia elèctrica dels senyals.

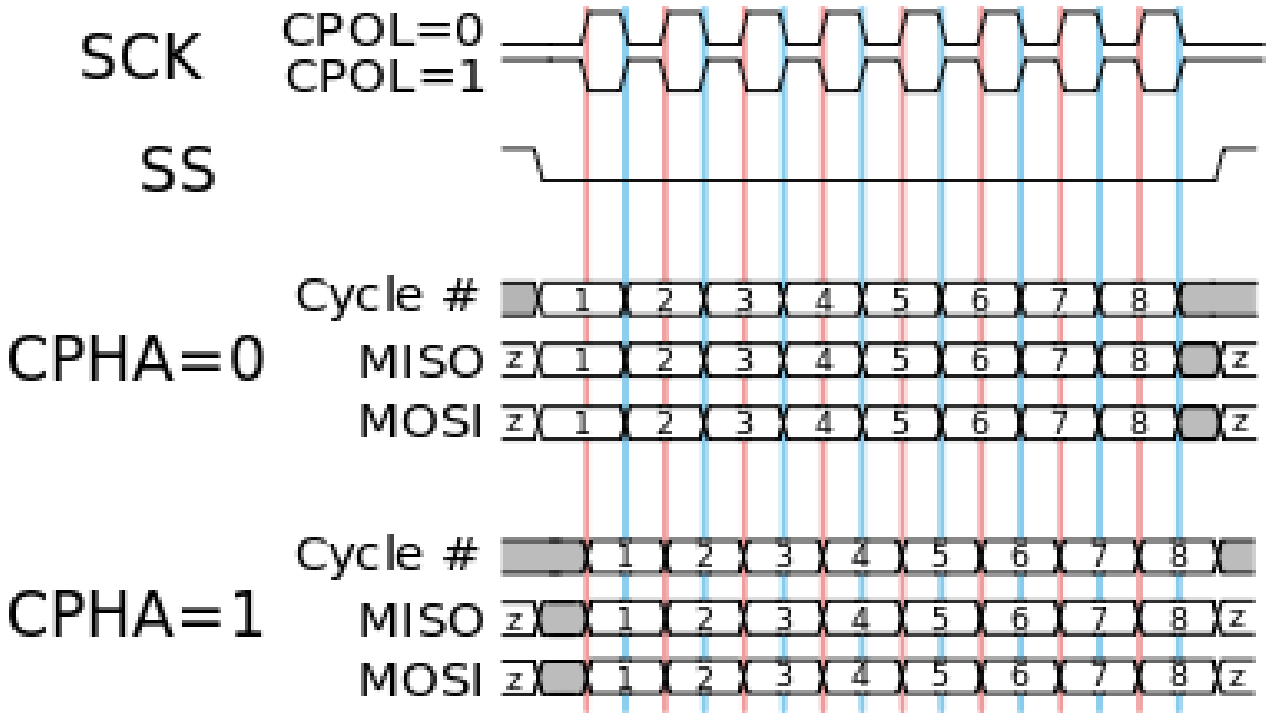
El protocol SPI tindrà un únic màster, les senyals són unidireccionals i hi haurà una senyal SS per cada esclau.



UNIVERSITAT DE BARCELONA

SPI - Paràmetres

SPI és un estandard que només està definit a nivell de bit, no diu res de com s'han d'organitzar les trames, etc. Només te dos paràmetres: clock polarity (comença a nivell alt o baix el clock) i clock phase (escritura en flanc de pujada o baixada).



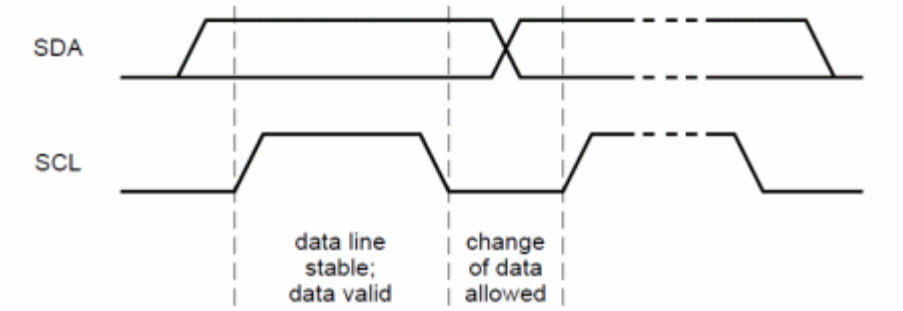
I²C – Inter-Integrated Circuit

Velocitat habituals de 100 – 400 kbps. Tot i així, és molt susceptible a interferències i fàcilment pot tenir errors de transmissió.

Línies del bus:
- Serial Clock Line (SCL): Senyal de rellotge bidireccional.
- Serial Data Line (SDA): Senyal de dades bidireccional.
- Ground: massa, referencia elèctrica dels senyals.

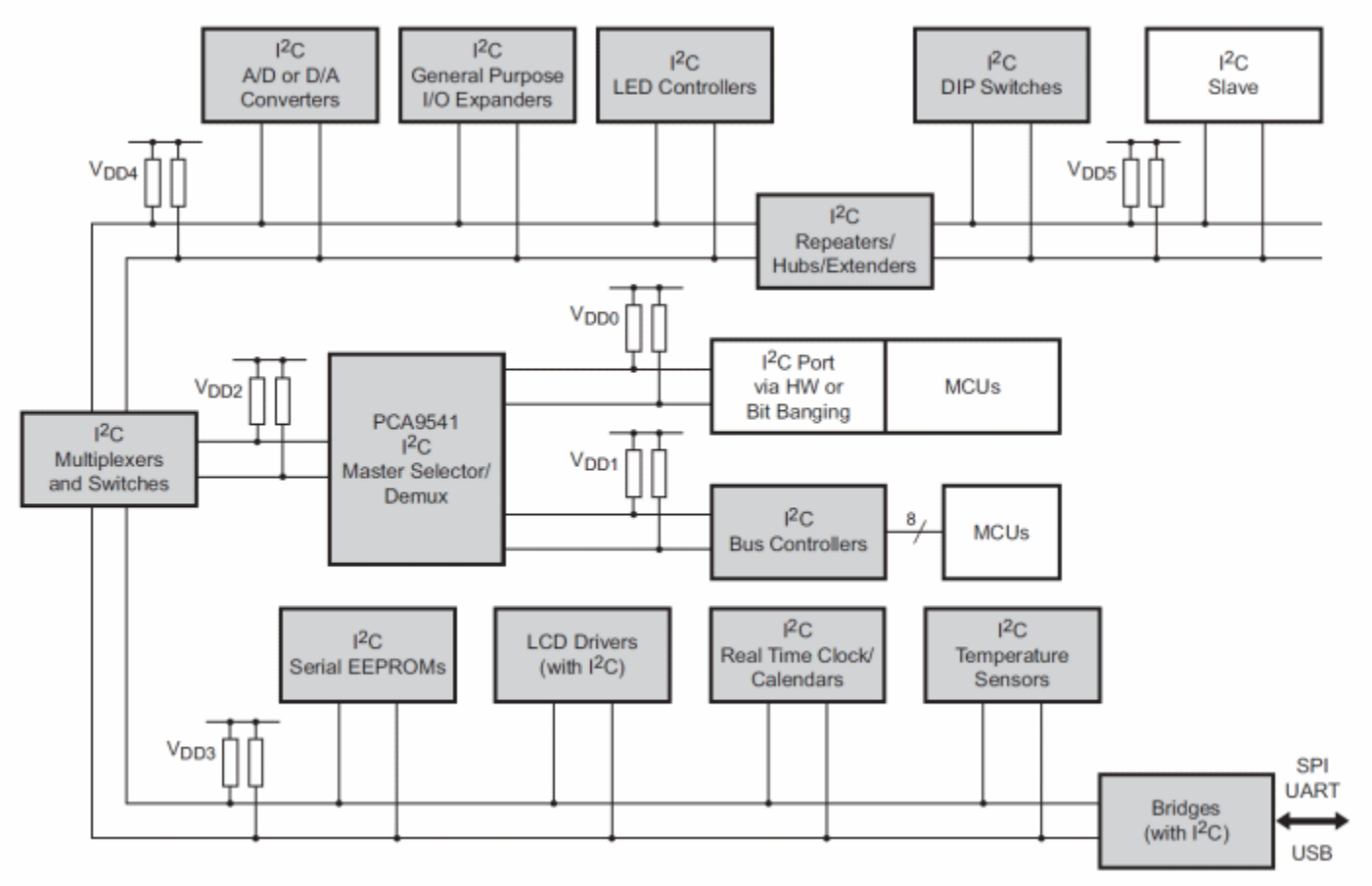
Les senyals són open-drain (s'escriu només el '0') i requereixen de resistències pull-up.

És un estandard complet: defineix la capa física, les trames, la gestió de la xarxa, com tenir diferents masters i com passar d'una a un altre, etc.



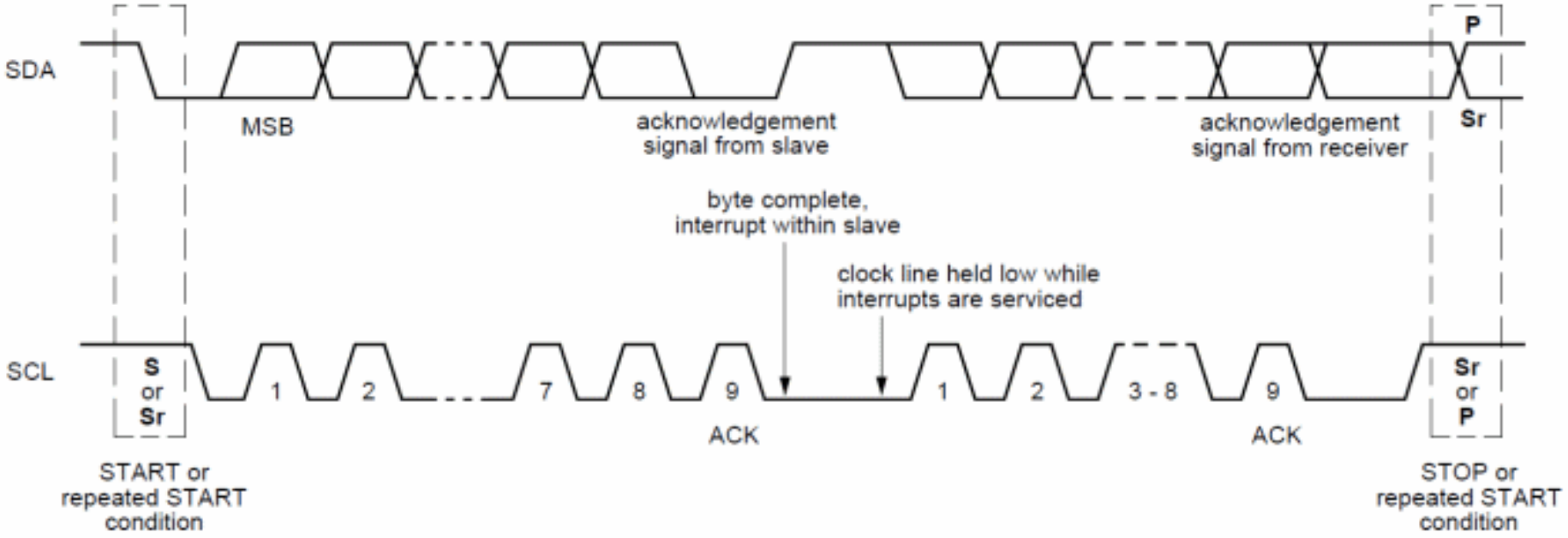
UNIVERSITAT DE BARCELONA

I²C – Inter-Integrated Circuit



I²C – Frame

START -> 8 bits del master (7 adreça + R/W) -> ACK del slave (o error de com) -> 8 bits dades (MSB) + ACK (n vegades) -> STOP

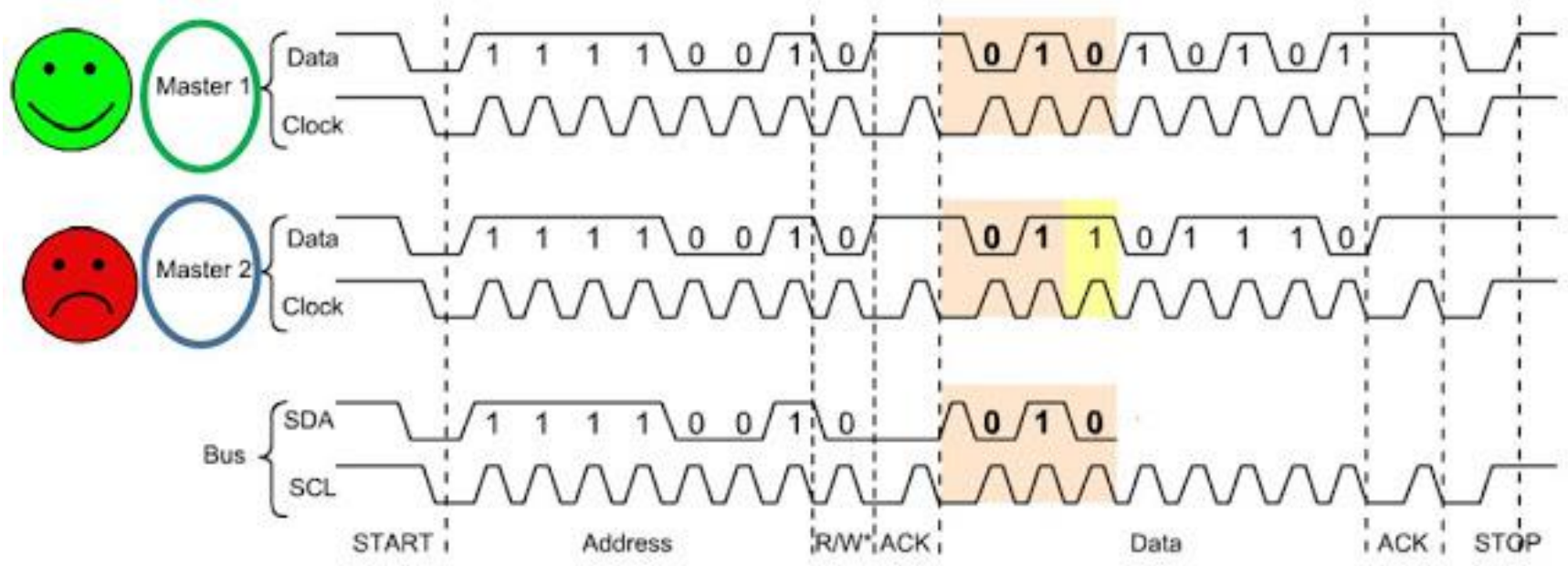


En el ACK, el master controla el clock (sempre ho fa), però el slave escriu el '0'

Després del ACK, el slave pot mantenir la data a '0' si necessita temps abans de continuar.

I²C – Multi-master

El protocol permet múltiples masters i defineix com és passa d'un a un altre sense perdre informació ni reintents.
Qualsevol master poden iniciar una transferència quan el bus està lliure (no hi ha clock ni en mig d'una transmissió) baixan iniciant un START. El primer que ho fa és el master. Per això, els masters han de mirar la línia per saber si realment ho són o no. En cas de conflicte, el 0 domina i el master que ha escrit un 1 ha de cedir.

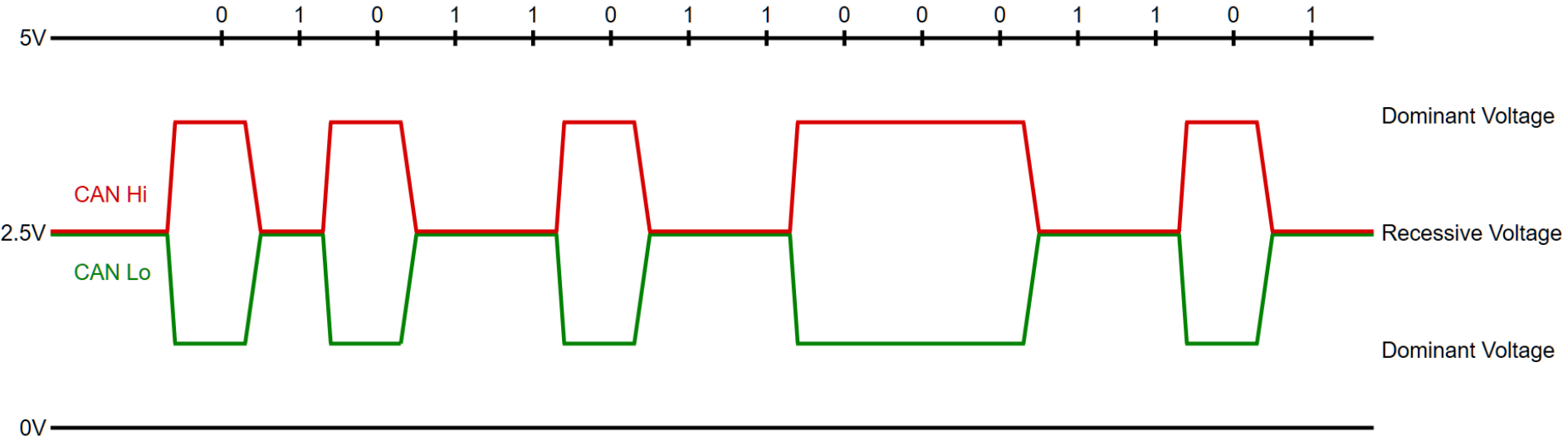


CAN

Estàndard de comunicacions molt usat en automoció. És un protocol complet que defineix des de la capa física fins a les trames i permet una comunicació basada en prioritat de missatges que a afectes pràctics permet diferents masters segons prioritat del missatge.

Línies del bus:

- CAN_H i CAN_L: senyals elèctriques diferencials, amb un nivell dominant (0) i un recessiu (1). Requereixen d'un transceiver extern al MCU normalment.



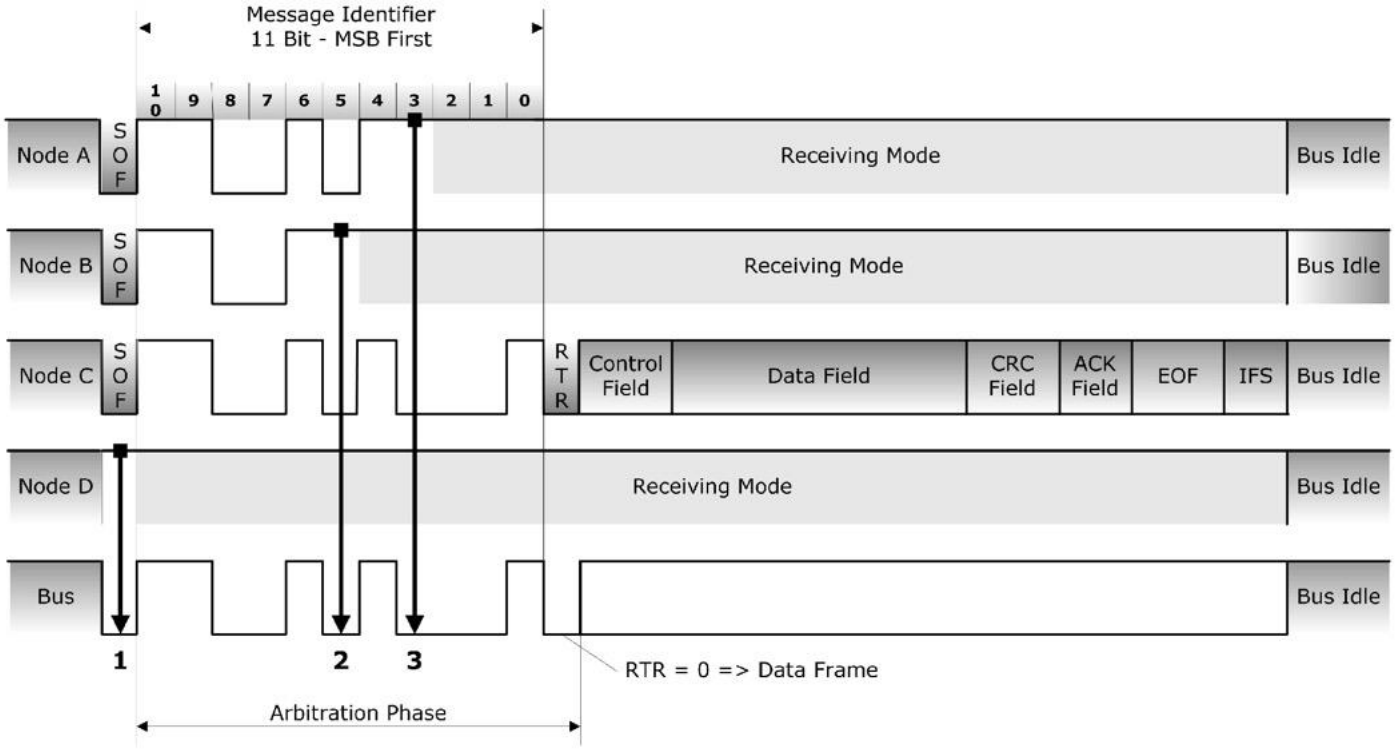
CAN

Al inici dels frames de CAN hi ha el identificador del emissor del paquet. Similar al i2c, en cas de conflicte el valor dominant guanya i el emissor amb ID més baix (major prioritat) manté el control del bus i el de valor més alt perd el control.

ID: 0x65F

ID: 0x67F

ID: 0x651



CAN

Les trames estan també definides:

- ID del missatge (IDx)
- Sol·licitud d'enviament de missatge (Requ. Remote)
- Camp de longitud del missatge (DLx)
- Camp de dades (longitud DLx bytes)
- CRC per control d'errors (CRCx)
- ACK recessiu per l'emissor i dominant pel destinatari