UNIVERSITÀ
DEGLI STUDI DI TRIESTE

dipartimento
di ingegneria
e architettura

Serial Communication

A.Carini – Microcontrollers

Serial communication

- It is the simplest and cheapest way to realize a data exchange between a µC or DSP and an external peripheral.
- It allows transmission and reception, also simultaneous, of data *one bit at a time* and requires a minimal physical support (3-4 lines, also just 2 lines).
- In contrast, the parallel transmission requires a bus with as many lines at the bits parallel transferred plus some control lines, but allows faster speed.
- The serial communication is much used and is at the basis of many successful communication *protocols,* as Controller Area Network (CAN) or Inter Integrated Circuit (I2C).
- All µCs and most DSPs integrate peripherals called *USART, Universal Synchronous Asynchronous Receiver Transmitter,* for implementing serial communications.
- Many have more sophisticated peripherals capable to simplify the implementation of CAN, I2C, and SPI protocols.

Serial communication

- There are two possible mode of operation: *synchronous* and *asynchronous.*
- The *synchronous* mode allows faster speed but requires a more complex physical support.
- The *asynchronous* mode requires a minimal complexity of the connection (just 2 lines for unidirectional communications) but allows only slower speeds.
- The peripherals of µCs allow both modes, but the asynchronous mode is usually preferred when there is no need for high speed.
- The synchronous mode is usually implemented with one of the following communication buses:
  - *I2C (Inter-Integrated Circuit)* introduced in mid 80's by Philips (now NXP).
  - *SPI (Serial Protocol Interface)* developed in 1979 by Motorola (later become Freescale, NXP)
- Both protocols have a certain complexity, managed with dedicated peripherals.

Asynchronous serial communication

- It is implemented *without* a clock synchronization between the transmitter *Tx* and the receiver *Rx.*
- The receiver will extract synchronism from the data line.
- Each transmission is preceded by a *start bit* and is concluded by one or more *stop bit.*
- Initially conceived for isolated transmission, with long pauses between transmissions, thanks to its simplicity it is also used for continuous transmissions.

Asynchronous serial communication

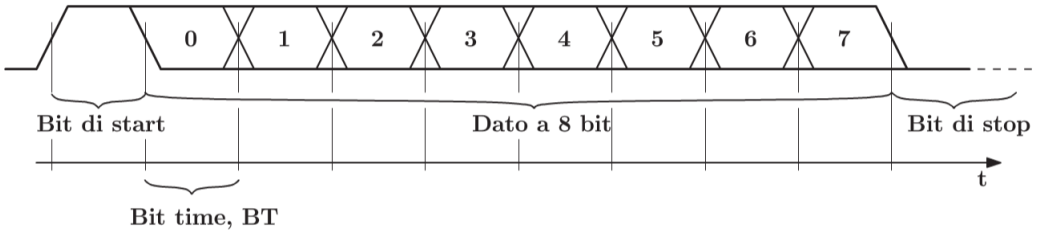


Figura 13.1: Diagramma temporale di una trasmissione seriale asincrona. La figura assume che lo stato inattivo della linea sia quello *basso*. Sono inoltre evidenziati gli intervalli temporali, o bit time, assegnati a ciascuna cifra del messaggio trasmesso.

- The same time interval, called *bit time,* is assigned to each transmitted bit.
- We should distinguish between *logic states*, *active* and *inactive,* and *logic levels, low* and *high*. The association between state and level can change.
- The start bit is *always* signaled by a transition of the line between inactive (here low) to active state.
- The stop bit has always the polarity of the inactive state.

Asynchronous serial communication

- The channel configuration that must be shared between *Tx* and *Rx* is based on few typical parameters:
  - Number of start bits: 1
  - Number of data bits: from 5 till 8
  - Number of *parity* bits: 0 or 1.
  - Number of stop bits: 1, 1.5, 2;
  - Communication speed: 1200, 2400, 4800, 9600, 19200 bps
- These parameters are configured acting on some registers of the USART peripheral, here acting as *Universal Asynchronous Receiver Transmitter UART.*
- The optional parity bit allows to check communication errors: it is 0 if the sum of data bits is even, otherwise is 1 (but is also possible the opposite convention).
- The “additional” bits reduce the transmission speed. In the best case, 8+2 bits, the data rate is reduced by 20% from the start and stop bit.

Synchronization extraction from data line

- The synchronization procedure provides that the receiver start to sample the data line after its first transition from inactive to active.
- Ideally, the receiver should sample bits in the middle of their temporal interval, which is BT=1/*bps*, with *bps* also called *baud rate*. Thus, at 1.5 BT, 2.5 BT, …
- Rx and TX know the communication speed that has been set, but each has his own frequency, which introduces an uncertainty on the sampling times.
- The problem is solved by sampling the data line with a period lower than bit time, typically BT/16.

Synchronization extraction from data line

- The synchronization procedure starts at first transition from inactive to active.
- After 6 sampling periods, three consecutive samples are read to check start.
- From this moment on, every 16 periods 3 samples are read.

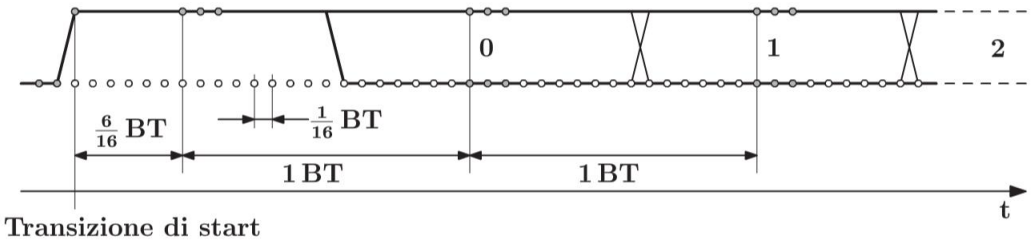


Figura 13.2: Campionamento della linea dati da parte del ricevitore. Il valore logico assegnato al bit corrente è deciso con logica di maggioranza all'interno del blocco di 3 campioni acquisiti. Questo abbassa la probabilità di errore.

Synchronization extraction from data line

- We can evaluate the maximum allowed clock difference between Tx and Rx.
- The most critical bit for sync errors is the last data bit.
- Taking into account the uncertainty in detecting the start, at most BT/16, a sync error could occur if, in a time equal to 9.5 BT, the receiver accumulates a drift of ±(0.5 -1/16) BT, which means:

$$\left[\frac{\Delta f}{f_{Tx}}\right]_{MAX} = \frac{|f_{Tx} - f_{Rx}|}{f_{Tx}} \leq \frac{0.4375}{9.5} = 0.046,$$

- A relative frequency precision of ± 2.25 % is sufficient to avoid errors. Such precision is easily achievable with quartz oscillators.

Synchronous serial communication

- The transmitter acts as master and sends its clock to the receiver on a dedicated line.
- Often further control lines are present, which allow to send a *frame sync,* each word or frame.
- With this serial, the lines can be used for 100% of time to transmit data and the transmission speed can be increased in comparison with asynchronous systems.

Example

- In the picture, the clock falling edge signals “data is stable and can be sampled”.
- The frame sync indicates the beginning of a new word.

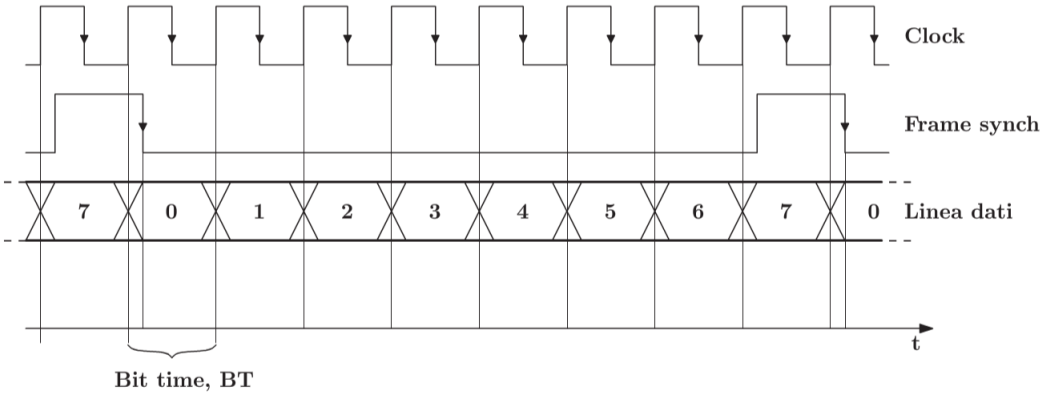


Figura 13.3: Diagramma temporale di una comunicazione seriale sincrona. Il fronte di discesa del clock indica al ricevitore quando il dato è stabile e può essere campionato. Nello stesso modo, il segnale di sincronismo di frame indica al ricevitore l’inizio di una nuova parola.

Synchronous serial communication

- USRT peripherals allow to configure many of the transmission parameters:
  - The *clock polarity*, i.e., the clock edge that signals the sampling time
  - The *data polarity,* the logical association to high and low voltages.
  - The bit transmission order, *MSB first* or *LSB first*.
  - The data size, 8, 16, ….
- Also the frame sync can be configured in many modes.
  - The active phase can last 1 BT or 1 word.
  - It is possible to transmit sync every two words.
- In many USRT it is possible to flexibly set the clock frequency, since there is a programmable clock divider.
- In other cases the transmission frequencies are predefined.
- In rare cases, clock can be generated by an external circuit.

Physical layer standards for serial communications

- Serial communication, both synchronous and asynchronous, is often implemented according to one the following physical layer standards:
  - RS232, in one of the many variants;
  - RS422/v11 or v12;
  - RS485;
  - Current loop at 20mA.
- They define the number of conductors, the voltage levels, the line drive.
- In case of closed systems, i.e., in systems where the serial communication is used for connecting internal components, it is possible to not meet any particular standard.

RS232

- Introduced in 1962 and modified many times.
- Was designed to connect a DTE (Data Terminal Equipment) and a modem.
- The standard defines 54 lines, establishing their functions.
- An RS232 connector has 25 poles, or in many cases, 9 poles.
- Only 3 lines are needed for asynchronous communications (Rx, Tx, 0V ref).
- The voltage levels established by the standard are +12 V (logic 0), -12 V (logic 1). But many devices works also with +10 V, -10 V.
- Communication with RS232 standard often requires a *driver,* a device capable to drive the lines, adapting the µC output voltages (usually ≤5V) to the line voltage, and providing a sufficient current to sustain a cable also longer than 1m.
- The communication is *full duplex,* since we can have a simultaneous Tx and Rx.

RS232

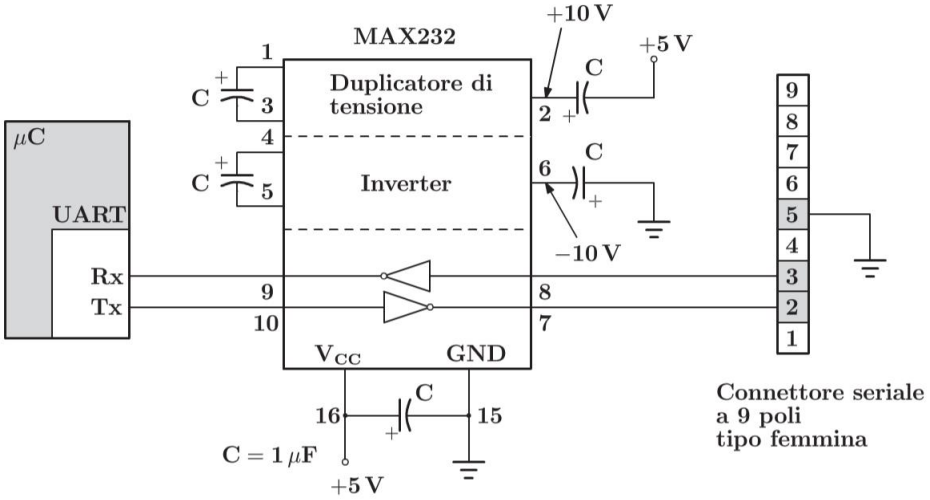


Figura 13.4: Esempio di collegamento seriale asincrono, secondo lo standard RS232, tra un microcontrollore e un dispositivo esterno, come ad esempio un PC.

RS422 and RS485

- These standard were introduced to guarantee a good quality of serial communication also in industrial environments, characterized by a relatively high electromagnetic pollution.
- Both standards foresee data transmission in *differential mode* with two conductors normally intertwined to minimize possible interferences by the electromagnetic field radiated by other conductors in proximity.

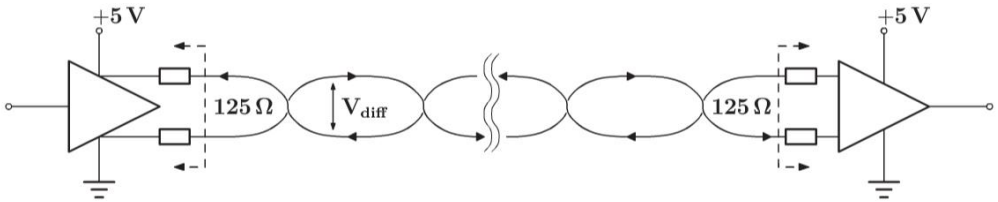


Figura 13.5: Esempio di collegamento seriale asincrono secondo gli standard RS422 o RS485. Si notino l'effetto dell'intreccio dei fili sull'orientazione delle spire e la tensione differenziale che trasporta l'informazione. Trasmettitore e ricevitore offrono impedenze adattate alla linea di trasmissione (125 Ω).

RS422 and RS485

- The differential mode transmission associates the information to the difference between the voltage levels of the two lines.
- The differential transmission is insensitive to the common mode components, i.e., to a common voltage applied to both conductors. Capacitive couplings between the two lines and conductors in proximity give rise to common mode voltages that alter the average level of the two lines.
- The two lines are intertwined to allow cancellation of voltages induced by variable magnetic fields, since two close loops will have opposite induced voltages. With this trick, the transmission is robust against inductive couplings.
- The better immunity to noise allows us to sustain relatively fast transmission speed (10 Mbs for distances < 15m).
- In contrast to RS422, in RS485 it is possible to have multiple-transmitters on the same line, with each Tx in high impedance when inactive.

Digital transmission with current loop

- The connection between two devices can also exploit the circulation of a current to transmit information.
- This channel, called *current loop,* associates a current, typically of 20mA, to a logic 1, and the absence of current to a logic 0.
- The transmission method is much used in industry, since it is less sensitive to electromagnetic noise. The current is imposed by the driver and is independent from the average or differential voltage level of the two conductors.
- To realize the communication, often opto-isolated drivers are used, which keep galvanically isolated the transmitter and receiver up to kV voltage differences.

Digital transmission with current loop

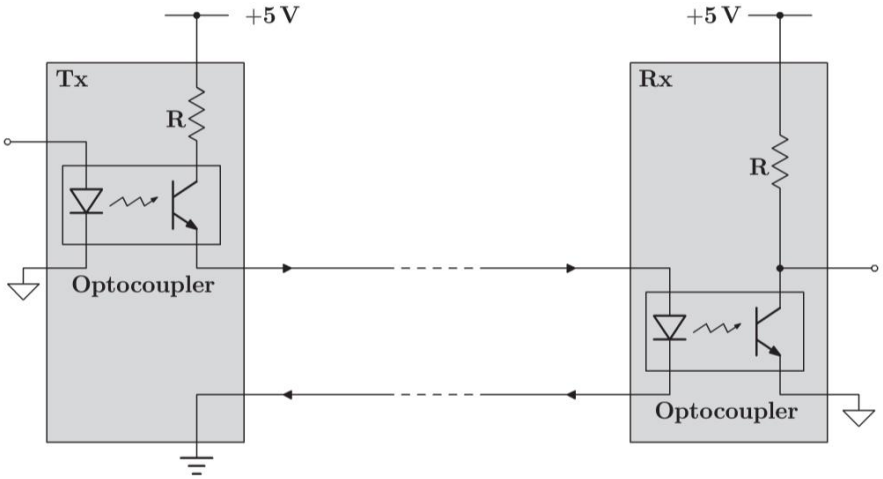


Figura 13.6: Esempio di collegamento seriale asincrono in loop di corrente a 20 mA. Il driver sfrutta un optoisolatore, che funge da semplice generatore di corrente.

Bus I2C, Inter Integrated Circuit

- Is a bus standard largely used at industrial level.
- It allows the *synchronous* serial interconnection of a suitable µC (with I2C peripheral) and a large variety of compatible digital integrated circuits.
- The µC acts as *bus master,* with the bus composed by two lines, one for data, DA, and one for clock, CL.
- There is no frame sync.

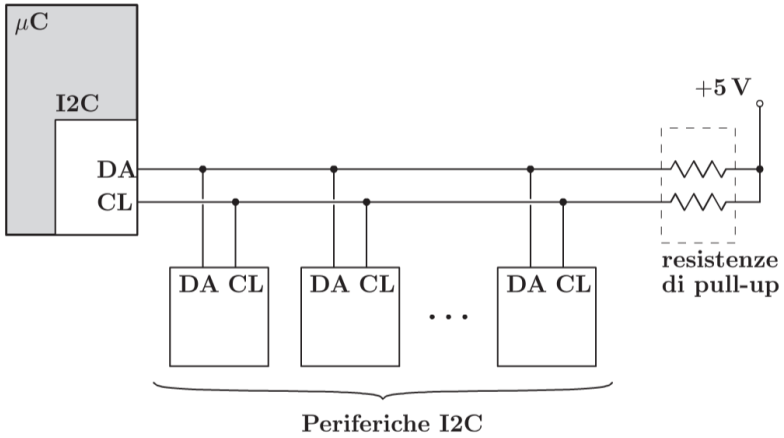


Figura 13.7: Configurazione tipica del bus di comunicazione I2C.

Bus I2C, Inter Integrated Circuit

- In I2C, different operation modes are possible, which differ for the allowed transmission speed.
- In *normal mode*, we have data rates between 0 and 100 kbps; in *fast mode* (introduced later) we arrive up to 400 kbps, now we have also a *ultra fast mode,* up to 5 Mbps.
- In case many devices with different speed are connected to the same bus, the bus shall be operated at the data rate of the slowest device.
- The protocol transmits messages composed by a 9 bit packet.
- Each message begins with the µC generation of a *start* signal and ends with a *stop* signal from the microcontroller.
- The start and stop conditions are characterized by data line that moves towards low (start) or high (stop) *during the clock high phase.* Any other bit is set during the low clock phase.

Example

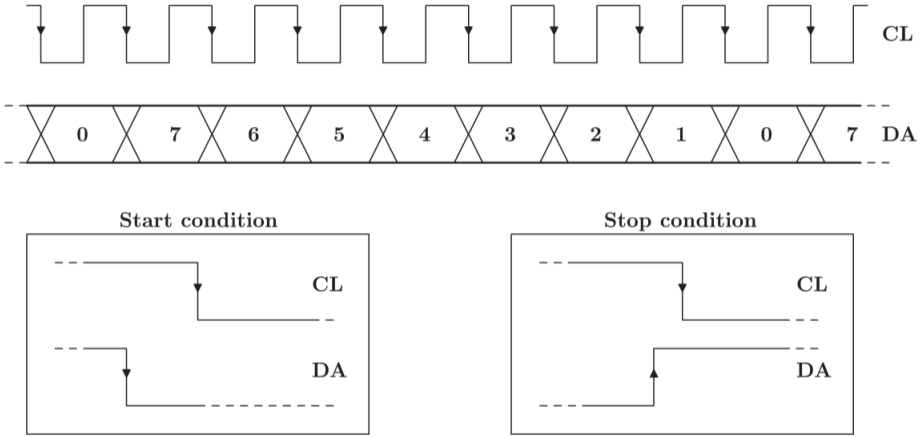


Figura 13.8: Modalità di trasmissione di un dato sul bus I2C. Nei riquadri, le condizioni di "start" e "stop".

Bus I2C, Inter Integrated Circuit

- The first 8 bit of each packet are written by the transmit unit, the ninth bit is written by the destination unit. Each byte start with the MSB.
- The ninth bit is an *acknowledgement* bit. The receiver must acknowledge setting low the data line at bit 9.
- The µC connects with a peripheral by transmitting as first byte after *start* the address of the peripheral (7 bit) and a bit establishing the direction of transmission (low from µC to peripheral, high for vice versa).
- When a peripheral reads his address, it responds setting low the data line at ninth bit. The µC reading the low value knows he can start transmission or reception because he knows the peripheral is ready.
- In case of complex peripherals, the µC can send more than one byte for configuration purposes, before the communication begins.

Bus I2C, Inter Integrated Circuit

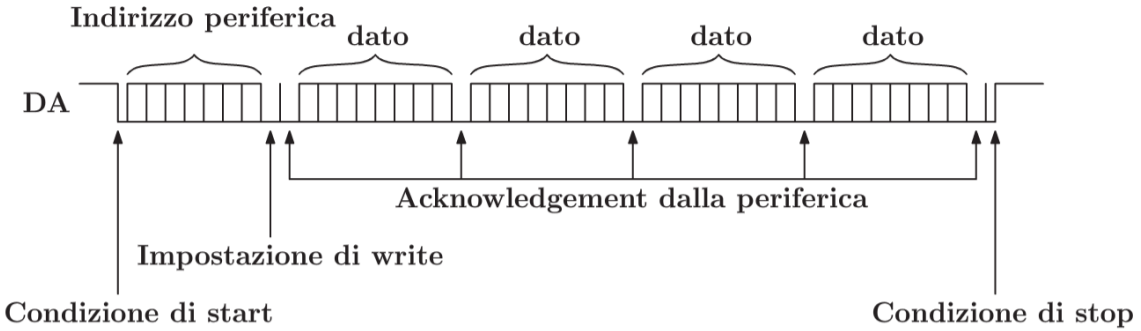


Figura 13.9: Trasmissione di dati dal microcontrollore ad una periferica nel bus I2C. Il microcontrollore avvia e conclude la trasmissione: il primo byte indirizza la periferica e imposta la condizione di write, tenendo la linea bassa nell'ottavo bit time.

Bus I2C, Inter Integrated Circuit

- Reading data from a peripheral, the µC can stop communication by not giving acknowledgment after a byte received.

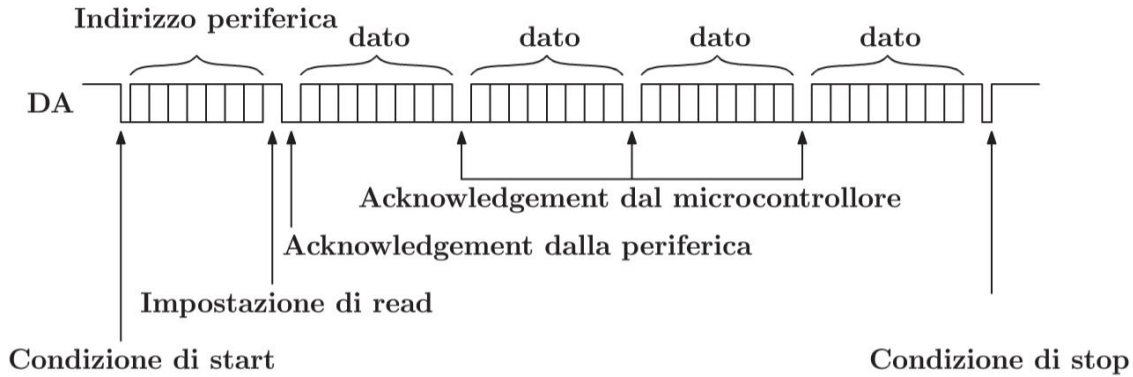


Figura 13.10: Trasmissione di dati da una periferica al microcontrollore nel bus I2C. Il microcontrollore arresta la trasmissione dei dati da parte della periferica lasciando alta la linea dati nel bit time riservato all'acknowledgement, dopo il quarto byte ricevuto.

Bus I2C, Inter Integrated Circuit

- In commerce are available many kind of devices with I2C interface.
- You can find demultiplexers, temperature and pressure sensors, A/D and D/A converters, etc.
- All allow the user to define the peripheral address by setting the level of some dedicated pins.
- In general just the lowest 3 can be set, since rarely more than 8 peripheral of the same type are needed.

Bus SPI

- Exploits a synchronous serial communication.
- The number of lines is at least 5:
  - A clock (SLCK) sent by master to all devices,
  - A selection line (SSn) for each slave device;
  - A MOSI (master out slave in) data line
  - A MISO (master in slave out) data line
  - A reference voltage line (0V).
- The protocol is very simple, the master starts communication by activating the slave selection line, then start sending the information bit to MOSI, and sample the MISO line. Communication is full-duplex.
- There are 4 communication modes, that differ from the clock edge where the MOSI line change state and for the inactive clock level. These are controlled by two parameters: CPOL, Clock polarity, and CPHA, Clock phase.

Bus SPI

- The master-slave couple share the same settings of clock frequency, CPOL, CPHA, but the configuration can be changed according to the selected slave.

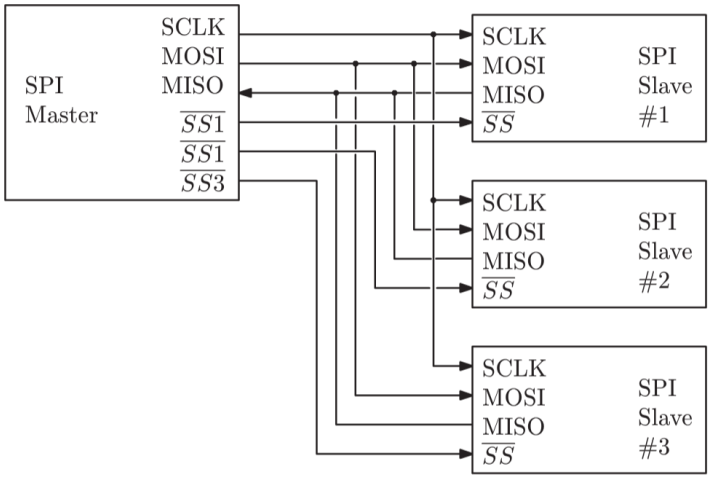


Figura 13.11: Connessione su bus SPI di un microcontrollore (master) e 3 unità periferiche slave. Nel caso più semplice, il master dispone di una linea di selezione $\overline{SSn}$, attiva a livello basso, per ciascuno slave.

CAN Bus

- The bus CAN (Controller Area Network) is an *asynchronous* serial standard for industrial environments, also for real-time applications.
- Developed by Bosch in 1986 under request of Mercedes, the CAN bus is characterized by high transmission speeds, low costs, and numerous mechanisms for error detection and correction.
- The CAN bus has a particular structure of data frames, called *messages.*
- It adopts a specific mode for accessing bus and arbitrating contentions.
- At physical level, it follows the RS485 standard, with an intertwined pair having impedance of 120Ω.
- Data transmission speed depends on the connection length.

CAN Bus

Tabella 13.1: Velocità del bus CAN in ragione della lunghezza di collegamento.

Lunghezza del bus [m]	Velocità di trasmissione $Mbit/s$
40	1
100	0.5
200	0.25
500	0.125
6000	0.01

CAN Bus messages

- CAN Messages do not have sender and receiver, but are *labelled* on the basis of their content.
- The various units interfaced with the bus receive only and all the messages whose content is relevant for them.
- The ID of the content of a message shall be unique for all the network.
- Without addresses, the CAN bus is very flexible and allow *plug and play* addition of new nodes.
- Moreover, there is *no bus master*.
- Since a message can start anywhere, the network is *multicast.*
- When two or more unit contend the bus for message transmission, the *assignment* is *automatic* on the basis of the *implicit priority* of any *ID*.
- Low binary values prevail on the higher ones according to a *wired-and* logic, i.e., low levels are *dominant* and high level are *recessive*.

CAN Bus messages

- The message with the lower ID prevails, and *looser* units stop:

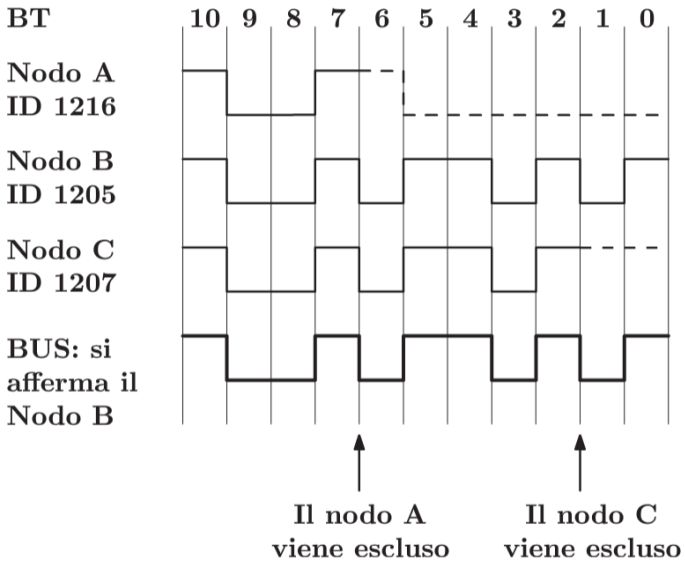


Figura 13.12: Meccanismo automatico di risoluzione dei conflitti nel CAN bus. Il nodo B si afferma in modo automatico, perché il suo messaggio ha un identificatore di valore più basso.

CAN Bus messages

- There are 4 possible messages:
  - Data frame;
  - Remote frame;
  - Overload frame;
  - Error frame.
- The first two messages have different formats in case of a *standard* (11 bit ID) or *extended* (29 bit ID) CAN bus.

- Let us consider an example of message of according to CAN 2.0A.

CAN Bus messages

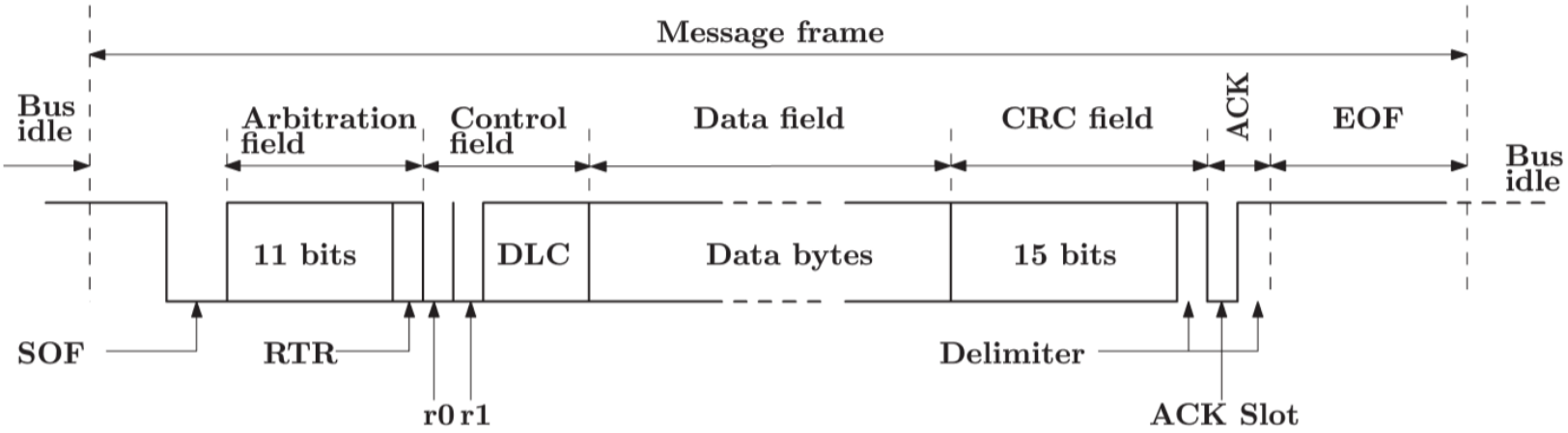


Figura 13.13: Formato di un messaggio secondo lo standard CAN 2.0A.

CAN Bus messages

- The *start of frame* is a dominant bit, a zero that follows an *idle* phase of bus, in which the bus is in *high* state.
- The message opens with the ID field of 11 bit, and is followed by a *Remote Transmission Request, RTR.*
  - *RTR = 0* for a *data frame,* i.e., if we are transmitting data.
  - *RTR = 1* for a *remote frame,* i.e., if we are requesting data.
- The next six bit are a control field. The first two are reserved (always low), while the next four indicate the number of bytes (from 0 to 8) that will be transmitted.
- The CRC field of 15 bits is reserved for *cyclic redundancy check code*, for error detection and correction.
- After a high bit, a two bit *acknowledgement* follows: in the first all peripherals that correctly received the message take the bus to 0; the second is high.
- EOF consists in 7 high bits.

CAN Bus messages

- The bus is considered free only after three high bits.
- After the EOF, the units can send on the bus an *overload frame.*
- The overload frame consists in a low bit, signaling “the node cannot receive other data because is busy processing the previous message”. The other units respond with a low bit on bus.

CAN Bus error conditions

- Each unit of bus detects an error any time there is a violation of the protocol transmission rules, at bit level or frame level.
- There are many rules that make this protocol robust.
- A first rule is that of *bit stuffing*. Any transmitter cannot send more than 5 bits of the same value. After 5 equal bits, one of complementary value is added. If a receiver read more than 5 equal bits, it is sure of an error.
- The transmitter continuously monitors the values he writes and, if he finds a value different from the intended one, he signals an “error” condition.
- A *checksum error* is generated by a receiver that finds a CRC error on the received data.
- A *frame error* is detected when the size of the fields are violated.
- An *acknowledgement error* is generated if no one acknowledge a message.

CAN Bus error conditions

- Any detected error causes the transmission of a single *error frame* from the node that detected it, and a consequent response form all other nodes.
- The *error frame* is compose by 6 equal bits, dominant (low) for the node that detected the error, recessive (high) for all other nodes.
- In presence of an error frame, the last message will be discarded (and all units will be protected).
- As a further protection mechanism, each unit counts the errors.
- A counter is incremented at each error, and decremented at each correctly received frame:
  - If the counter exceeds 127, the unit signal *malfunction.*
  - If the counter exceeds 255, the unit detaches from the bus, protecting the others.

Peripherals available on µCs and DSPs

- All µCs and many DSPs include peripherals for asynchronous (UART), and/or synchronous (USRT, SPI) communications.
- Often I2C or CAN bus peripherals are also available. CAN bus controllers are generally of the type *basic*.
- More and more often, µCs can connect to an Ethernet network.
- Programming peripherals for I2C or CAN bus, or Ethernet network, is complex. The problem is mitigated by the availability of specific libraries, provided by the same chip manufacturers.

Wireless communications

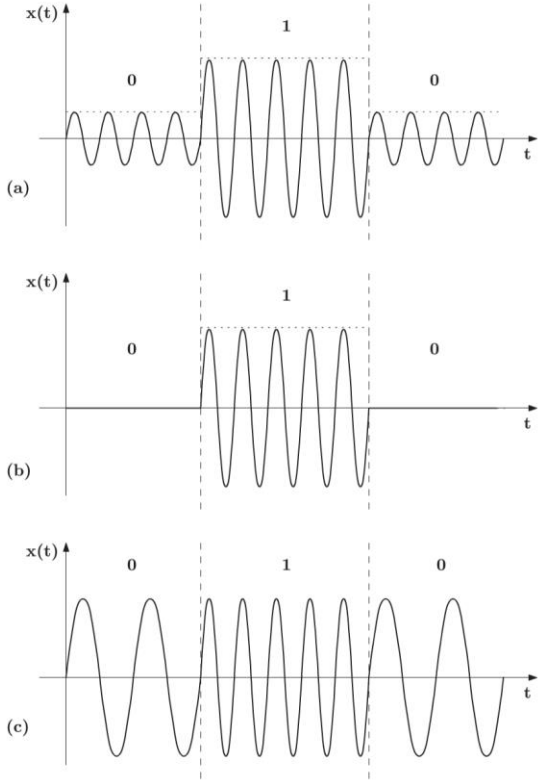
- The demand of wireless data transmission systems has increased in the last years, both in consumer and industrial electronics.
- In consumer electronics, we have high production volumes and low costs, which need simple solutions. As possible applications we have:
  - Smart domestic appliances,
  - Electricity metering and control equipment,
  - Wireless domestic alarm systems,
  - Home automation systems
  - Wearable electronic devices, …
- Industrial applications are instead characterized by higher complexity and higher reliability and security constraints.
- The availability of processing power and transmission devices *on the same chip* are very interesting in these fields.

Example

Tabella 13.2: Microcontrollori con periferiche di trasmissione wireless.

Costruttore	Dispositivo	TX/RX	Frequenza di lavoro [MHz]	Modulazione/ protocollo supportati
ATMEL	AT86RF401	TX	264 – 456	OOK
ATMEL	ATMega2564RFR2	TX e RX	2400	Zigbee
Microchip	rfPIC12F675K	TX	290 – 350	ASK, FSK
TI	CC430F51	TX e RX	300 – 348	ASK, OOK, FSK
NXP	JN516X	TX e RX	2400	Zigbee

Modulations



Amplitude Shift Keying

On/Off Keying

Frequency Shift Keying

Figura 13.14: Segnali modulati per: (a) modulazione digitale di ampiezza, ASK, (b) la sua variante più semplice, OOK, e (c) modulazione digitale di frequenza, FSK.

Wireless Transmission Standards

- Currently, those most used are the following:
  - IEEE 802.11, WiFi
  - IEEE 802.15.1, Bluetooth
  - IEEE 802.15.4, ZigBee

- ZigBee was recently introduced for applications with low power consumption (e.g., wireless sensor networks).
- It allows short range communication (30m indoor, 100m outdoor) and works at 2.4GHz (guaranteeing good penetration in walls, etc.)
- It has a slow transmission speed (250 kbit/s) suitable for sporadic communications of small length.
- Typical applications: Home automation, wireless sensor networks, but also industrial control and automation.

See:

- Simone Buso, «Introduzione alle applicazioni industriali di Microcontrollori e DSP» Società editrice Esculapio, 2018
  - Chapter 13