DIGITAL COMMUNICATION TECHNIQUES

- **Introduction and general issues**
- **Parallel bus communication (IEEE 488)**
- **Serial bus communications (RS232 … and USART drivers)**
- **Field bus (Fip, CanBus)**

DIGITAL COMMUNICATION TECHNIQUES

- **Encoding = bijective correspondence between transmitted info and symbols**
- **The encoding can be arbitrary but must be followed by both the communicating devices otherwise no comprehension**
- **Symbols are sent through signals associated to values ranges (within the calculator) or subject to modulation (in the used medium)**
- **Transmission media: wired twists and/or shielded, Wi-Fi (pay attention to the environmental disturbs), coaxial cables, optical fibers (pay attention to costs, maintenance, installation and required competences).**
- **Characteristics:**
  - ***bandwidth*** **(100 KHz - 1 GHz) $\Rightarrow$ max. transmission velocity**
  - ***attenuation*** **(0.4-1.5 dB/Km) $\Rightarrow$ max. distance without repeaters ($e^{-\alpha l}$)**
  - ***propagation velocity*** **(15000-150000 Km/s)**
  - ***characteristic impedance*** **(50-100 Ohm)**
  - ***noise immunity*** **(S/N, 100 dB)**

DIGITAL COMMUNICATION TECHNIQUES

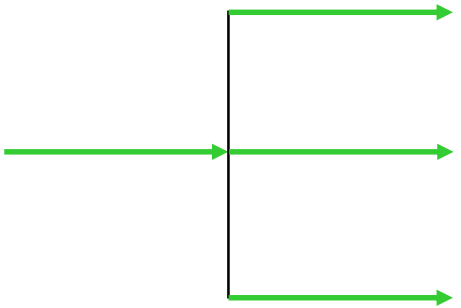
- **THE GENERAL PROBLEMS OF THE DIGITAL COMMUNICATIONS**

  - **Transfer direction (simplex, half and full duplex)**
  - **Organize the transmission content (bits, characters and messages)**
  - **Data synchronization and recognition (auxiliary signals)**
  - **Error avoidance**
  - **Error recognition and correction**
  - **Spatial and temporal coordination of interlocutors through *protocols* (who can transmit and how) and *routing* (shortest vs most efficient path policies)**

DIGITAL COMMUNICATION TECHNIQUES

- **TRANSFER DIRECTION**

– **simplex**, only one single-direction channel that allows to transfer from a sender to many (multicast) or all (broadcast) destinations.



– **half duplex**: 1 bi-directional shared channel or 2 single-direction opposed channels of which only one can send or receive (for instance the walkie talkie). Simple approach typical in local networks.

– **full duplex**: transmission and reception can occur simultaneously on the same channel (for instance telephone). The communication can consist of 2 simplex opposed channels or of 1 bi-directional channel.

DIGITAL COMMUNICATION TECHNIQUES

- **TRANSMISSION CONTENT**

– Hierarchical structure of information (bits, characters and messages)

– Characters are made up by bits (7, 8 or 9)

– ASCII or different format encoding to make the transmission content more compact and improve the robustness (immunity) to the noise

– Characters are organized in messages to

»Recover in a easier way the information content, by simplifying error detection and recognition

»Synchronize interlocutors

»Allow concurrent communications on the same transmission channel

– Characters length influenced by the error management techniques (duplications and redundancy)

DIGITAL COMMUNICATION TECHNIQUES

- **SYNCHRONIZATION AND RECOGNITION**

– Synchronization = to establish what time windows can be used to exchange information and what symbols are used to represent them

– Auxiliary signals

– If parallel communications $\Rightarrow$ one or more addictive lines

– If serial communications $\Rightarrow$ data and synchronization info are brought by the same line (for instance in synchronous communication)

– Bit, characters and message structure $\Rightarrow$ synchronization signals must be organized in a similar structure to recognize those bits that define a character and the bits that define a message (for instance in the asynchronous communication start and stop bits, SOM and EOM, …)

DIGITAL COMMUNICATION TECHNIQUES

- **ERROR PREVENTION**

– Disturbing phenomena restraint (filtering, differential amplification with high CMRR)

– Preventing from disturbs (shielding, sheaths, twisted pairs)

– “Robust” signals (high S/N)

- **ERROR RECOGNITION AND CORRECTION**

– Errors can never completely eliminated $\Rightarrow$ error rate $\neq 0$ that must be accepted and foreseen in the specs

– Errors rate can be even high but it is important that errors are detected and when possible original data recreated

»Not detected errors (if possible with a low rate)

»Detected errors with relative information discarding

»Recognized errors with information recovery

DIGITAL COMMUNICATION TECHNIQUES

- **PARALLEL COMMUNICATIONS**
- Groups of bits simultaneously transmitted

– Short distances (mechanical stress and noise)

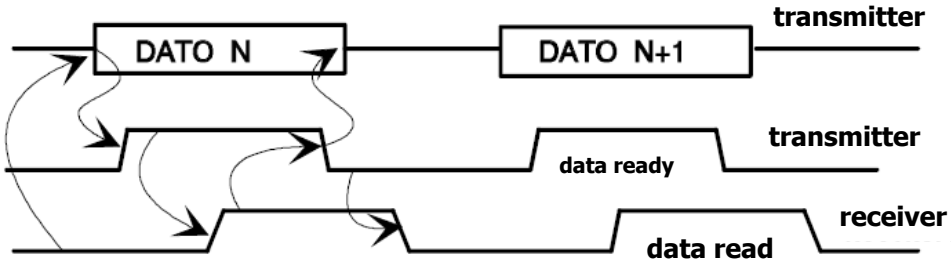
– System bus (with separated data and addresses) vs communication bus (data and addresses together)

– Auxiliary signals to synchronize the transfers (asynchronous bus). For instance:

  » ReadReq (memory read requirement ⇒ the address is put on the bus)

  » DataRdy (the data are ready and on the bus: the signal is managed by the sender)

  » Ack (ok reception of ReadReq and DataRdy and of the data)



– The slowest node defines the velocity

– Existent standards for interfacing: IEEE 488 and Centronics

DIGITAL COMMUNICATION TECHNIQUES

- **OBSERVATION: why do not use synchronous buses?**

In a synchronous bus the clock signal is included in the control lines

The use of a synchronous bus requires a fixed protocol for the communication that is a sequence of steps executed at a certain clock cycle (CC) sequence

For instance: in the first CC a "read" command and the relative address are sent by a device to the memory. The memory after few cycles (for example 5) provides the requested data on the bus, then the device reads in three CC and elaborates…

This can be implemented with a small finite state machine $\Rightarrow$ very few digital logic required, very fast response and small interfacing, but …

… all the devices must work at the same frequency (this is not said)

… the buses *must* be kept physically short to avoid the misalignment with respect other devices (*clock skew*)

DIGITAL COMMUNICATION TECHNIQUES

- **PARALLEL COMMUNICATIONS: IEEE 488 STANDARD BUS**

GPIB = General Purpose Interface Bus

At the beginning (’75) proposed by HP to carry out a network between calculation systems and instrumentations by different manufacturers (named HPIB)

Successively it was adopted as a standard by IEEE and redefined

The standard allows to connect up to 15 devices on the same bus

16 lines of which 8 of data, 3 for the *handshake* and 5 for interfacing management

The devices can play 3 possible roles: “*talker*”, “*listener*” and “*controller*”: thus it is possible to specify type and functions of the connected instrumentation

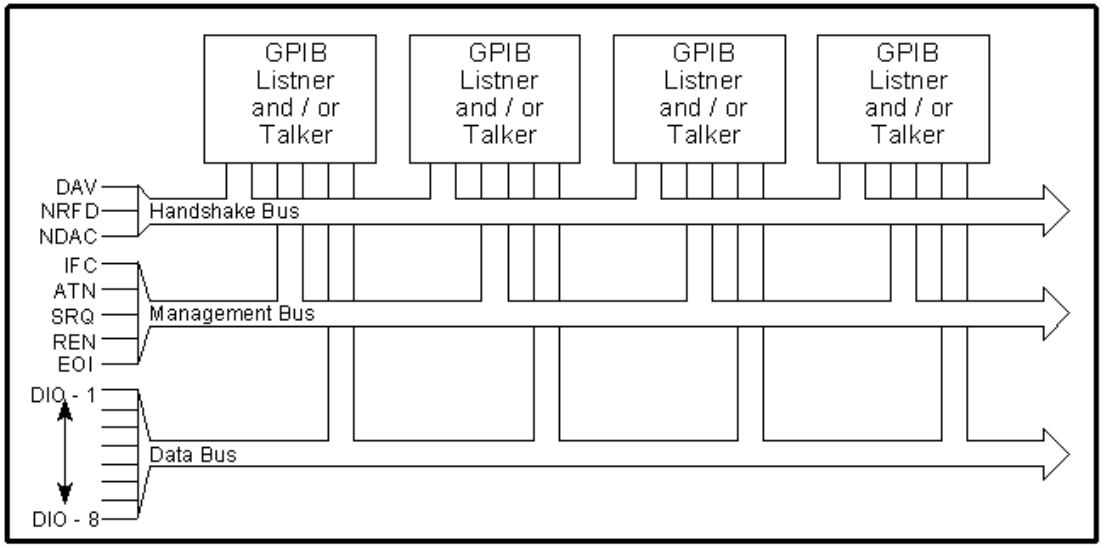
A device is programmed to be the System Controller at the beginning or the Active Controller during the normal working cycle and among its main functions can send a reset (IFC - interface clear) to all the devices or enable them to transmit (REN – remote enable)

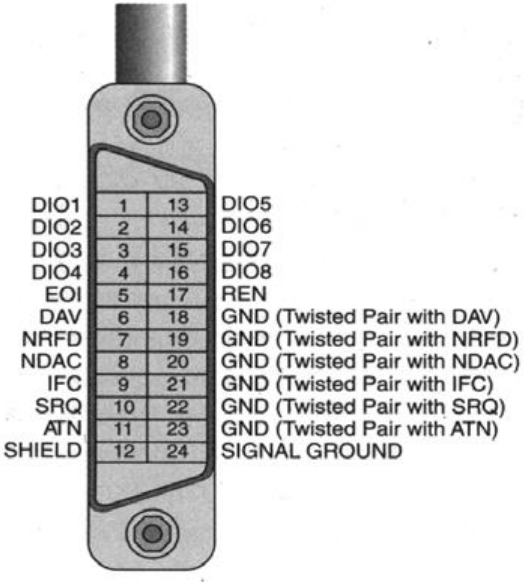
Transmission velocity = MB/sec

Max. length of the bus 20 m; max. distance between devices 4 m.

DIGITAL COMMUNICATION TECHNIQUES

- **PARALLEL COMUNICATIONS: IEEE 488 STANDARD BUS**

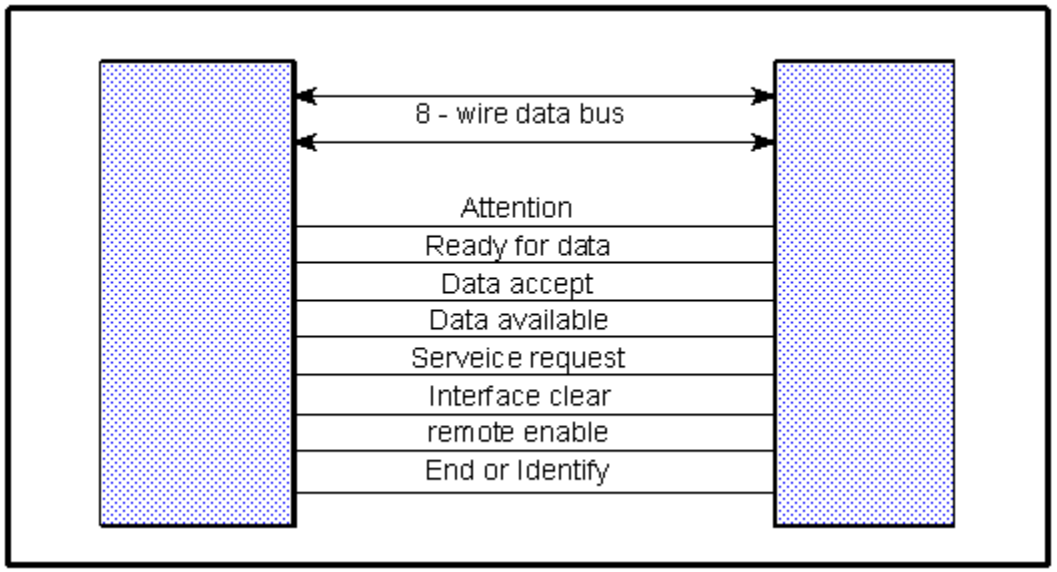




- Several "controllers" can be present but only 1 at a time will be Active (In Charge)
- The control is passed among the "Active Controllers" through a token mechanism (only the System Controller is established on the basis of the given system configuration)
- A "listener" device receives data from the bus only when the "Controller" enables it
- Similarly a "talker" device transmits only when enabled
- The "Controller" can set a "multicast" transmission mode between a "talker" device and a "listeners" group

DIGITAL COMMUNICATION TECHNIQUES

- **IEEE 488: lines to interface management**



- ATN (attention) is asserted by the "Controller" to indicate the placement of an address/control on the bus. The "Controller" disables it to allow the "talker" to transmit then re-enables it to continue in the transmission management
- EOI (end or identify) two possible uses: if enabled by the "talker" it means end of transmission; if enabled by the "controller", together with it ATN, it stands for a (parallel poll) to all the devices
- IFC (Interface Clear) enabled only by the "System Controller" to bring all the devices in a known initial state
- REN (Remote Enable) enabled only by the "System Controller" to arrange the devices in a remote control mode (command reception) that disables the local control from front panel (if present). The control will become complete only when those devices will be addressed as "listener"
- SRQ (Service Request) = a kind of interrupt by a whatever device that asks the "Controller" for a certain action. The "Controller" identifies the emitting device through a "serial poll". When polled, the device releases the SRQ signal

DIGITAL COMMUNICATION TECHNIQUES

- **IEEE 488: handshake signals**

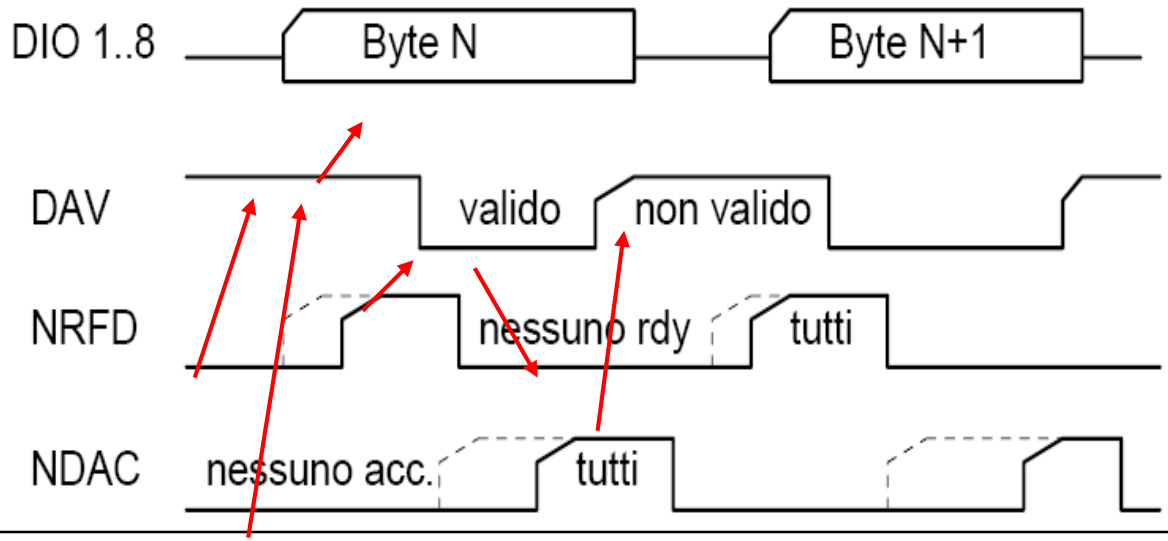
TTL inverse logic signals

NRFD (Not Ready For Data): set high from receivers (low active) if (all) ready to accept data. Open collector mode

DIO: data lines

DAV (Data Valid): active low only after the data have been put on the bus and the NRFD signal has been found high

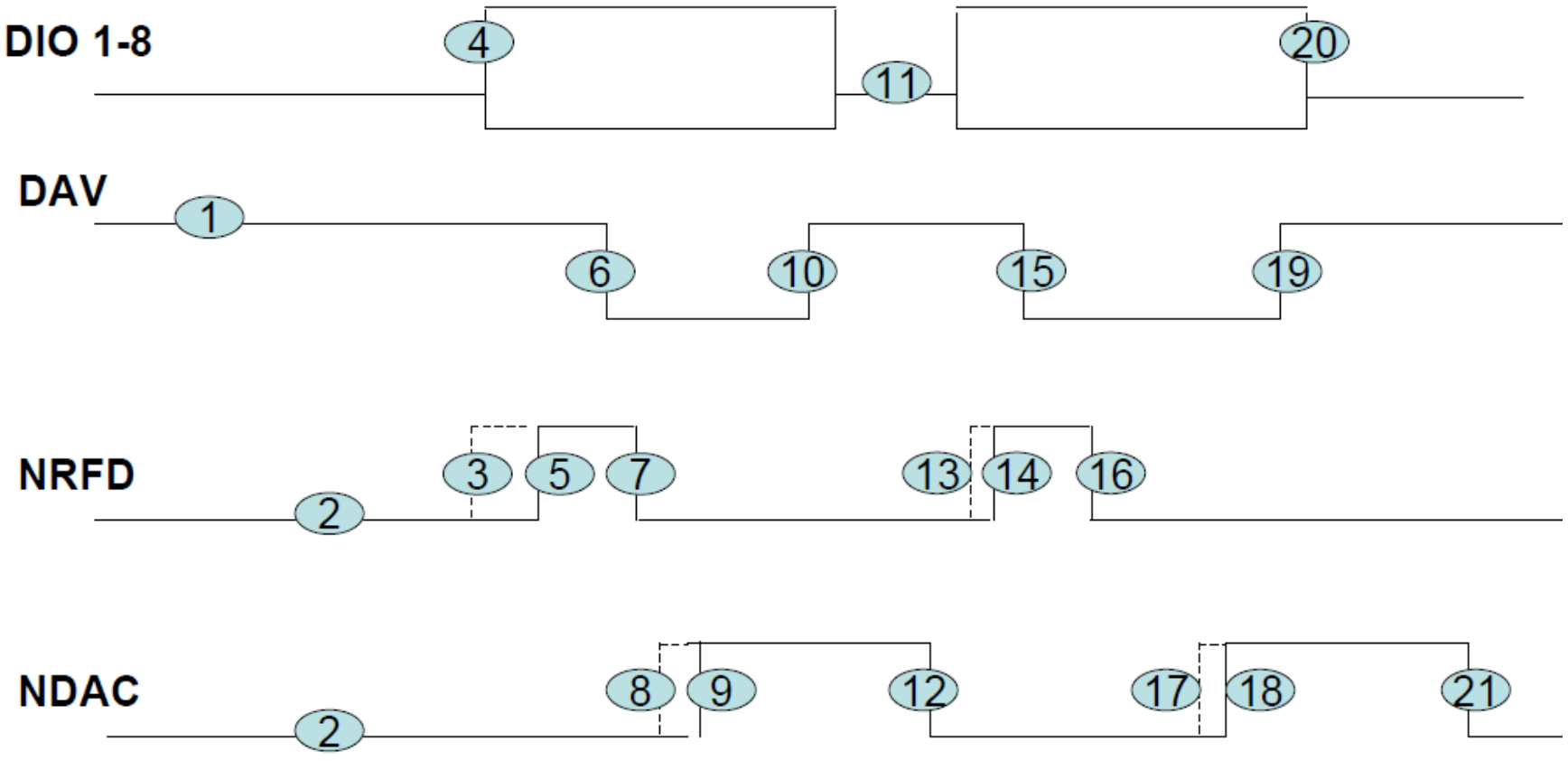
NDAC (Not Data Accepted): set low by receivers until a new byte has not been acquired. If high, DAV will be disabled and DIO lines changed with a new value. Open collector



Handshake

1. DAV high if the sender ("controller"/"talker") wants to transmit. If NRFD and NDAC low, data are written in the bus DIO. Only when NRFD high, DAV will be set low by the sender
2. Every receiver keeps NRFD low to signal to be "busy"; when the last one will have completed the reception, NDAC will be set high and the sender will set DAV high to transmit the next byte.
3. If, when set high DAV, the sender finds NDAC and NRFD high an error happens. Moreover if a device cannot perform correctly the handshake and releases the NDAC or the NRFD a error happens.
4. The error is managed by the operative system as a time-out error

DIGITAL COMMUNICATION TECHNIQUES



Courtesy prof. Laracca – università di Cassino

DIGITAL COMMUNICATION TECHNIQUES

- **DEVICE ADDRESSING (when ATN true)**

The 5 low bits constitute a primary address (MSb always 0, parity check)

Bit 5 indicates Listener address

Bit 6 indicates Talker address

0	T	L	4	3	2	1	0

UNT = Disable talkers = 3F; UNL = Disable listeners = BF

A secondary address can be conceived to address sub-units of a generic device. Normal addressing followed by a secondary address with bit 5-6 set to 11

Commands

The following commands control the bus operations and can be grouped in 4 classes:

Universal/Multiline (to all the peripherals with handshake): DCL, LLO, SPE, SPD, PPU, UNT, UNL

Uniline (to all periph. without handshake): IFC (Interf. Clear), REN (Remote En.), ATN (Attent.)

Addressed: GET, SDC, GTL, PPL, TCT

Secondary: PPE, PPD

DIGITAL COMMUNICATION TECHNIQUES

Multiline Command	*Mnemonic*	*Action*
Device Clear	**DCL**	**All the instruments are brought in a predefined state by the manufacturer, that depend on the instrument and specified in the datasheet; the device must be in remote mode or previously addressed.**
Local Lockout	**LLO**	**Disable the control "return-to-local" for the device (in practice disable the instrument front panel). The device can be addressed in local or remote mode. To enable again the local "return-to-local" control the physical line REN must be brought to the logic value 0: in such a way the device is set for local controlling (enabling from the front panel).**
Serial Poll Enable	**SPE**	**A Serial Polling of all the devices that can be addressed as Talkers is done; each one provides a status-byte, that represents with a suitable encoding the situation (service request, available data, errors, …). Used to debug.**
Serial Poll Disable	**SPD**	**Terminates the Serial Poll mode while resets the devices in Talker mode, so they provide device-dependent information or other status generic information.**
Parallel Poll Unconfigure	**PPU**	**Terminates the peripheral parallel poll for diagnostic that is similar to the serial one but faster (each peripheral returns 1 bit) although with less information**
Untalk	**UNT**	**Disable the current Talker device. A similar result can be obtained by selecting another device as Talker.**
Unlisten	**UNL**	**Disable all the Listener devices since it is not possible to disable a single Listener; this command can be conveniently employed to disable all the listener devices, before enabling all those devices that we want to set as listener.**

DIGITAL COMMUNICATION TECHNIQUES

Addressed Command	*Mnemonic*	*Action*
Group Execute Trigger	**GET**	**All the Listener devices receive the command to start the action previously defined for each of them. The GET command allows to simultaneously synchronize many devices.**
Selected Device Clear	**SDC**	**All the devices addressed as Listeners are brought back to a device-dependent pre-defined state. It is equivalent to the DCL command after the destination has been addressed**
Go To Local	**GTL**	**All the devices addressed as Listeners are brought back in local control mode (exit from remote mode). The device will come back to the remote function if addressed as Listener with REN line 1.**
Parallel Poll Configure	**PPC**	**All the devices addressed as Listeners are configured as the command following PPC, waiting for a parallel poll.**
Take Control Talker	**TCT**	**The "Controller In Charge" enables the current Talker device so as it assumes the role of Bus (Active) Controller.**

Secondary Command	*Mnemonic*	*Action*
Parallel Poll Enable	**PPE**	**Configures all the devices that received the PPC command so as they can answer to the Parallel Poll on a particular data line (DIO), with a certain pre-defined value.**
Parallel Poll Disable	**PPD**	**Disable the devices that received the PPC command so as they do not answer to the Parallel Poll (end of Parallel Poll).**

DIGITAL COMMUNICATION TECHNIQUES

Serial Polling

A service request (SRQ) allows to the controller to receive info on the device state, but it allows also to the controller to understand if one or more devices asks for service.

Procedure: the ATN signal is activated (bus in command mode), the command SPE is sent, all the devices are sequentially checked, each of them returns its status byte, then the command SPD is sent eventually together with UNT.

Parallel Polling

The same as before concerning the status of a device. The Controller activates ATN, then sends the PPC command to each device, assigns to each one an I/O line for answering through the PPE command and sends a EOI (data request). Then it reads lines so identifying the device that requests the service and asks for a status byte to understand the requirement to fulfill.

Sometimes shared lines to provide logic AND/OR of the status of the sharing devices

The answer from a device must arrive within 200 nsec. The controller must read this answer within 2 μsec after the 200 nsec.

Eventual disable with con PPD and end with PPU

DIGITAL COMMUNICATION TECHNIQUES

SERIAL POLL

Advantages:

The controller achieves more information on the interrupt generation cause.

Drawbacks:

The time for a poll cycle augments with the number of connected devices.

PARALLEL POLL

Advantages :

The controller can poll up to 8 devices at the same time.

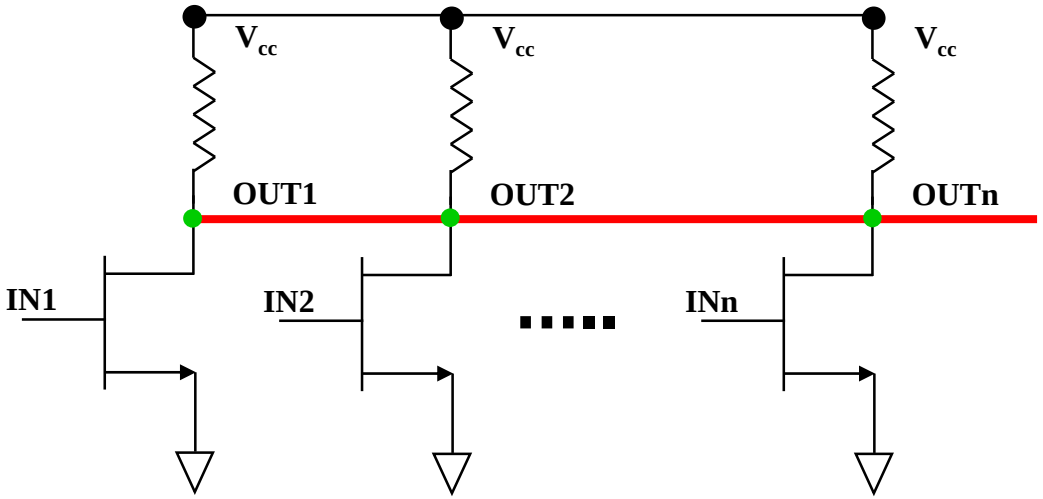
Drawbacks:

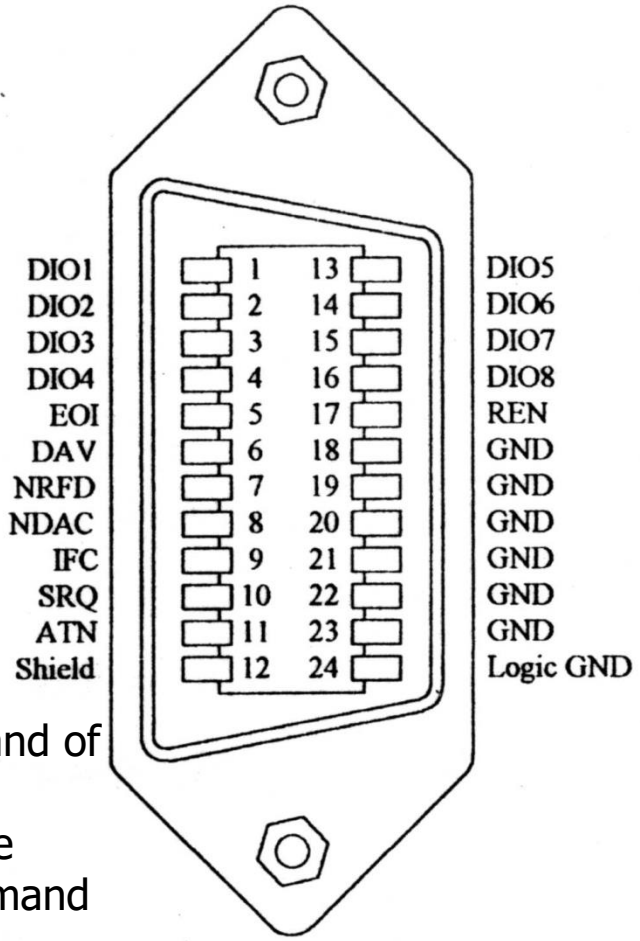
The status byte must be read to identify the interrupt cause.

Only 8 devices can be polled in parallel.

DIGITAL COMMUNICATION TECHNIQUES

- **PARALLEL COMMUNICATIONS:** collector wired or connection



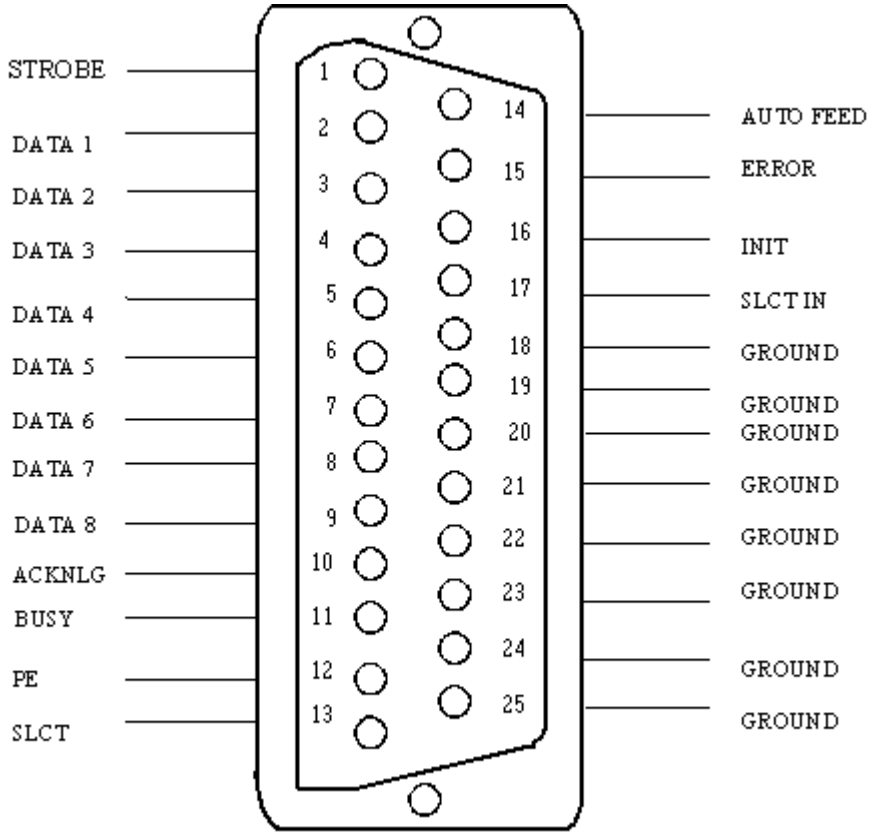


Automatic adjustment to the slowest device

IEEE 488.1 defines the standard from the electric, mechanical and of protocol point of view. The 488.2 version further enriches the standard by specifying data format, error management, possible controller functions, type of status information and typical command to be assigned to instruments

DIGITAL COMMUNICATION TECHNIQUES

- **PARALLEL COMMUNICATIONS :** Centronics interface



- Standard interfacing between PC and old printers
- 8 data lines and 9 control ones
- Output towards PC on a Cannon 25 poles jack
- Bi-directional half-duplex connection

DIGITAL COMMUNICATION TECHNIQUES

•SERIAL COMMUNICATIONS

A single signal bringing a sequence of bits one at a time

Short distances but greater than parallel communications (up to hundreds of meters)

Transmission velocity expressed in symbol or bit per second (baud rate):

- 75÷600 low standard velocities no more used
- 1200÷9600 low standard velocities still used (PLC, μP)
- 19200÷56400 average standard velocities
- 100÷1000 Kbit/sec average standard velocities for field bus
- 10÷1000 Mbit/sec high velocities for local LANS

Characterizing issues:

- bit, character, message synchronization techniques
- signal values, codification and eventual modulation
- error management

Existent standards: RS232 standard and successive versions and USB

DIGITAL COMMUNICATION TECHNIQUES

•SERIAL COMMUNICATIONS: bit synchronization

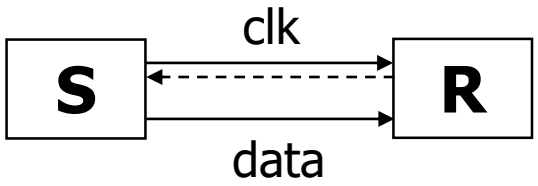
In what "time window" the receiver can sample and thus can read the voltage levels of the received signal

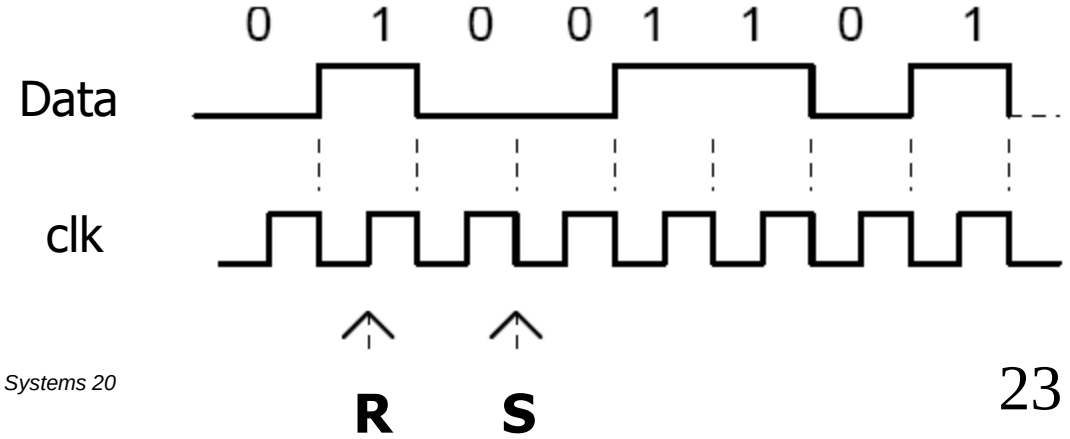
3 techniques: *isochronous, asynchronous* (mainly used with different manufacturers equipment) and *synchronous* (especially in field buses and local networks)

THE ISOCHRONOUS TECHNIQUE

A single clock signal shared between sender and recipients

The commutation edge on which the sender writes data is established conventionally (for instance falling edge) of course on the other edge the recipient will sample the line





DIGITAL COMMUNICATION TECHNIQUES

•SERIAL COMMUNICATIONS: THE ISOCHRONOUS TECHNIQUE

Synchronization characters are used

Precision in the pulses duration not required (edges determine the sample time)

If clock is generated by sender clock and data are in phase since they travel together

If clock is generated by the recipient it is not in phase with data. The recipient will receive data with a delay twice the time needed to propagate the clock toward the sender and then back $\Rightarrow$ the clock should be sized at least = to this time.

Limitations in velocity $\Rightarrow$ short connections (also to avoid to collect noise)

RS232 mode: a typical example

In another version a unique conditioned clock is used:

- a open collector signal between sender and receiver
- the receiver keeps the signal low until not ready, then it raises up
- when the sender detects the signal high starts to transmit and keeps the signal low
- Philips I^2C bus

DIGITAL COMMUNICATION TECHNIQUES

•SERIAL COMMUNICATIONS: THE ASYNCHRONOUS TECHNIQUE

Sender and receiver with their own independent clock regardless one or the other

Similar stable clocks (tolerance 1…3%) with standard values (300, 600, 1200, 2400, 4800, 9600 Hz)

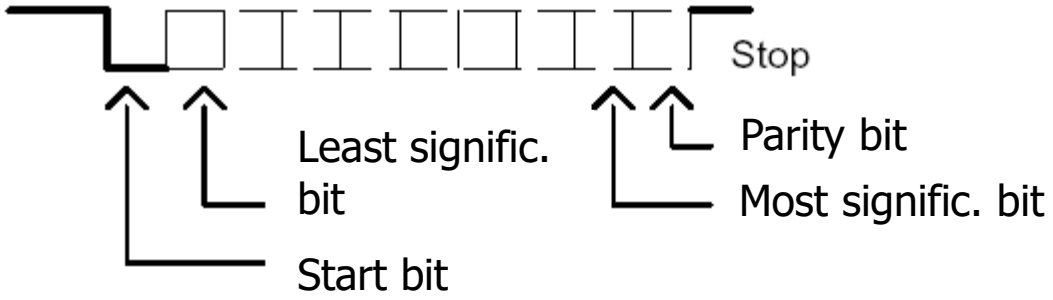
Byte synchronization through start (0) and stop (1) bits with duration 1, 1.5, 2 bit

Message synchronization through silence intervals (1) between a character and the next

Allowed phase shift ± half bit window time

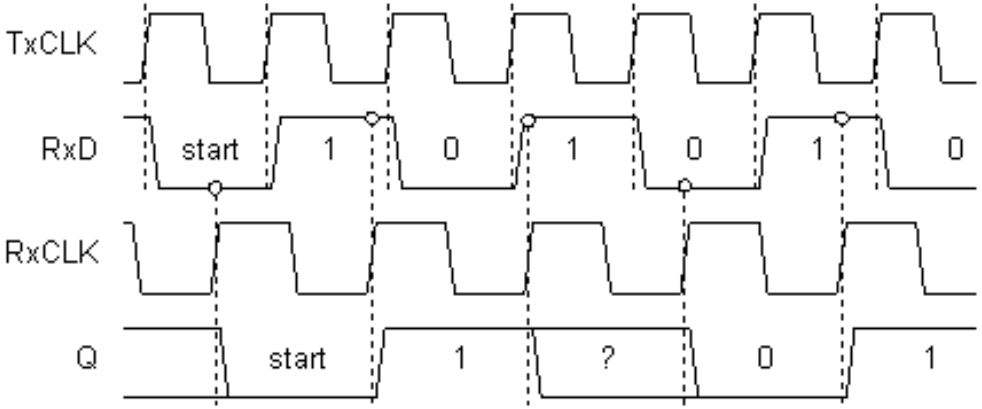
Each byte is synchronized when the first bit is received

Velocity in the order of Kbit/sec (mouse, printers, PLC)



Asynchronous transmission with 8 bit datum and parity

DIGITAL COMMUNICATION TECHNIQUES



TxCLK = sender internal clock used to synchronize serial data (RxD)

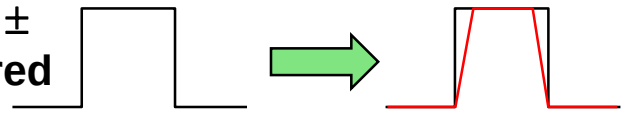
RxCLK = recipient internal clock, supposed slightly slower than TxCLK

Q = RxD bits as they arrive internally to the recipient

The two clocks feature a progressive phase shift that causes uncertainty on the value of the II bit (0) and the loss of the III bit (1). Conversely, if the sampling would be done very close to the beginning of each bit time window and if the sender clock frequency would be slightly minor than the recipient's one, the recipient could sample the same bit twice.

Therefore, the recipient clock must be re-synchronized at every start bit. The active edges (i. e. raising) will be located as much as possible in the center of the window time, so as to sample the RxD line as far as possible from those instants where the commutation can take place. This allows to tolerate slight differences between sender and recipient clocks.

The acceptable phase shift on 10 bit (8 data + start and stop) = ± 5% . This value diminishes when the edges cannot be considered as instantaneous (high velocities and limitation in the band).



DIGITAL COMMUNICATION TECHNIQUES

•SERIAL COMMUNICATIONS: THE SYNCHRONOUS TECHNIQUE

The sender features a free clock similar (but not equal) to the recipient: to cancel the phase shift a suitable PLL (phase lock loop) mechanism is provided.

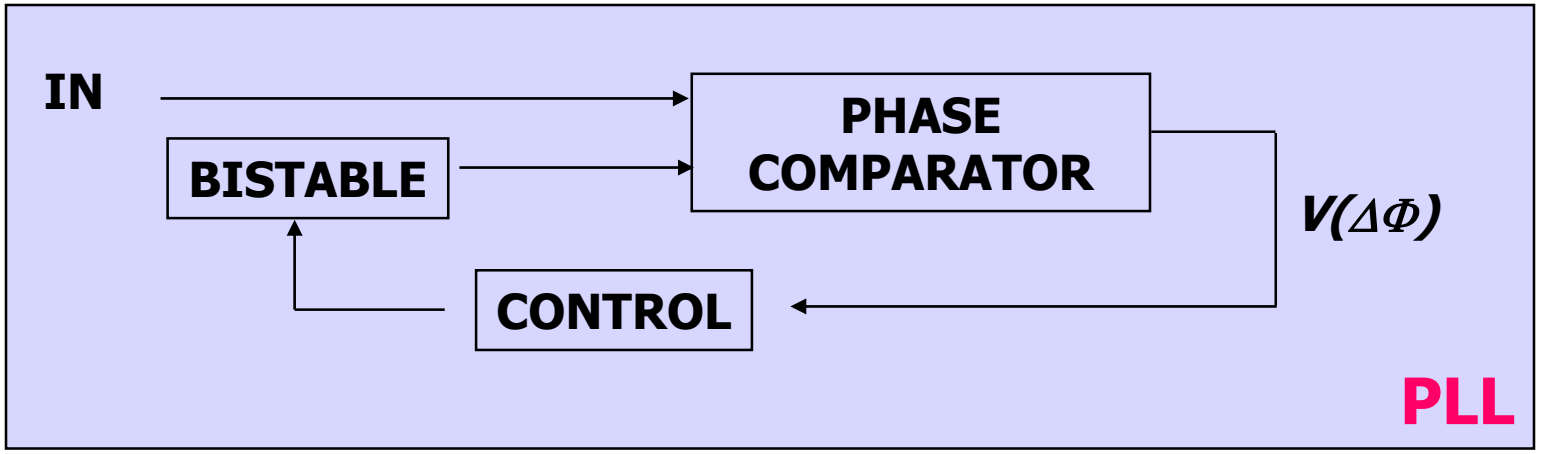
Synchronization bits are transmitted at first (from 3 to 5)

Suitable characters must be chosen to favor the phase coupling (AAAA ko, A6A6 ok)

Avoid long "1s" or "0s" sequences that may vanish the phase coupling

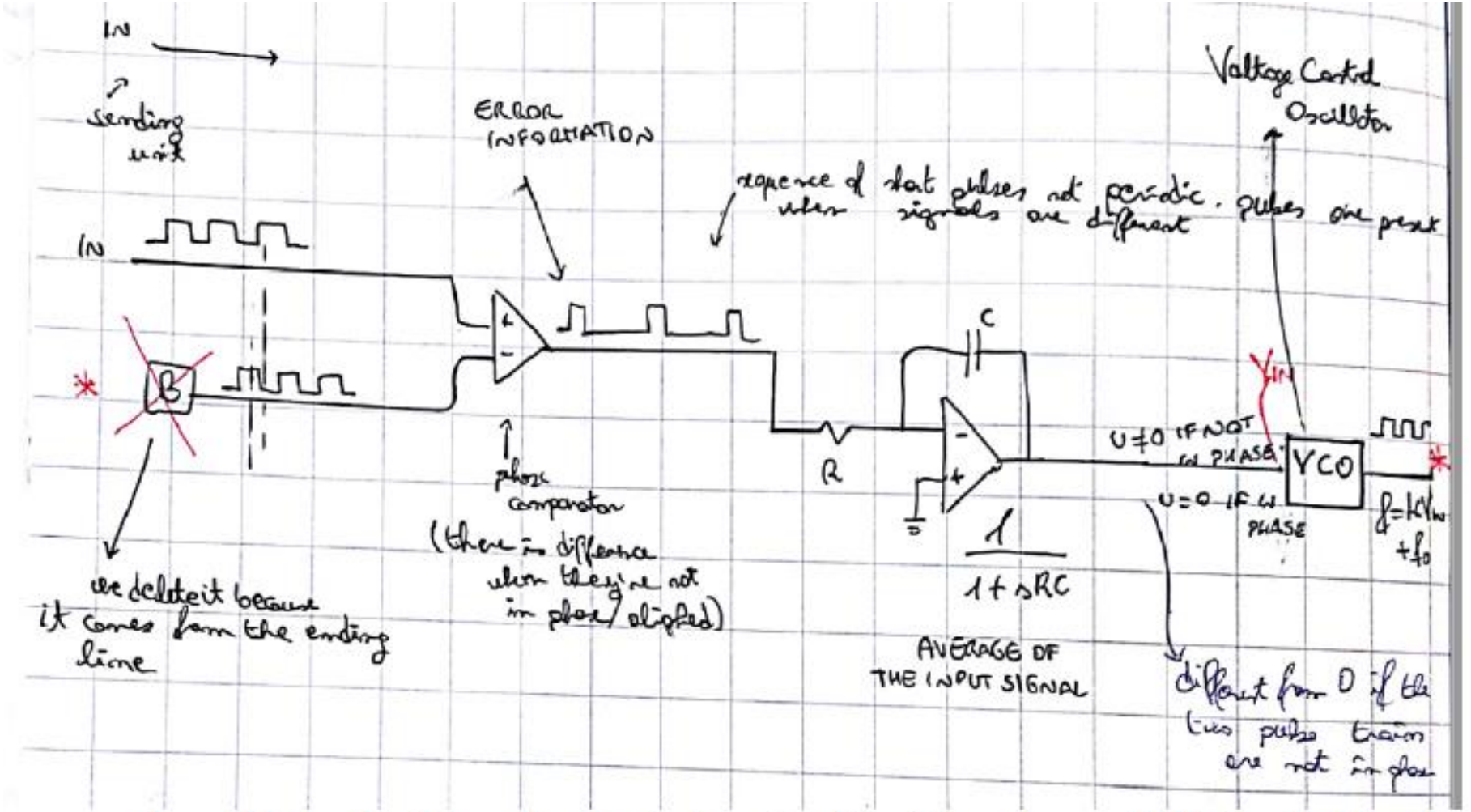
More complexity but more efficiency (no start and stop bits) and velocities

The PLL mechanism compensates eventual jitters on the clock



DIGITAL COMMUNICATION TECHNIQUES

- **PLL**



DIGITAL COMMUNICATION TECHNIQUES

BASE BAND ENCODING

NRZ (Not Reduced Zero): **The signal level is kept constant throughout the bit time. It may cause synchronization problems if long sequences of equal bits should be sent.**

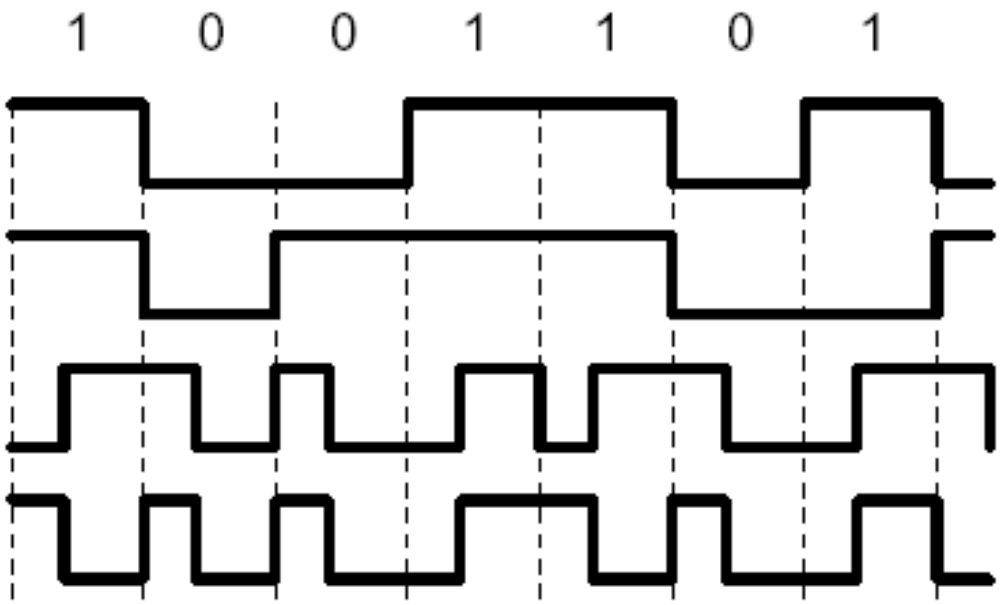
- ***RZ (Reduced Zero):*** **The signal level is kept high only for a bit time fraction. Thus, the synchronization problems in case of long sequences of "1s" (high) is solved (but not in case of "0s" (low).**

- ***MANCHESTER:*** **The basic idea is to divide the bit time in two portions equally long. Each bit will be sent as the sequence of two NRZ coded bits each in half of the time window. I. e. the bit 1 is transmitted as a 0 bit and 1 bit while the bit 0 is transmitted as a 1 bit + a 0 bit. This halves the transmission efficiency that is per each bit to transmit, indeed two are sent. In practice a bandwidth is required twice as much as it would be necessary. It is pretty used in local Ethernet and Token-Ring networks.**

- ***NRZI (Not Reduced Zero Inverted):*** **It solves the long "1s" sequences problem. The value 0 is coded maintaining the signal level throughout the period. This level corresponds to the one of the last transmitted bit. On the other hand, the "1" causes a level transition (from low to high or vice-versa) compared to the value of the last transmitted bit. Conversely if the transition level is due to a "0" it solves the problem caused by the transmission of long sequences of "0s". Used in FDDI networks.**

- ***Differential MANCHESTER:*** **It joins Manchester with NRZI encoding.**

DIGITAL COMMUNICATION TECHNIQUES



BASE BAND BINARY ENCODING

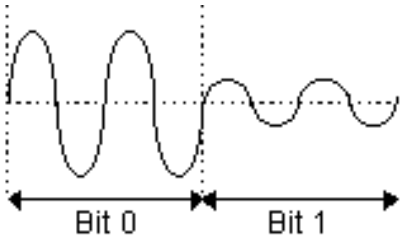
DIGITAL COMMUNICATION TECHNIQUES

•SINE SIGNALS ENCODING

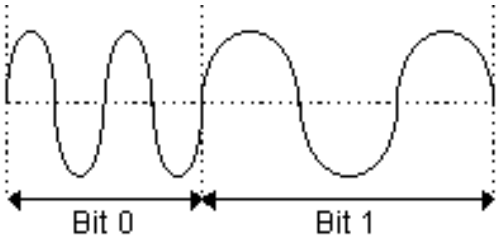
Width, frequency, phase possible modulation

Used when communicating with modems on telephone lines, radio link or coaxial cables

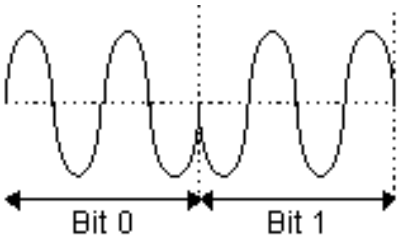
The "base band" signal is provided to the modem which carries out the modulation



width modulation



frequency modulation



phase modulation

•POSSIBLE SIGNAL RANGES

TTL level (0V ÷ 5V/3.3V)

bipolar voltages (-6V ÷ 6V): RS 232

Differential voltages: RS 485

"current loop" 0-20 mA

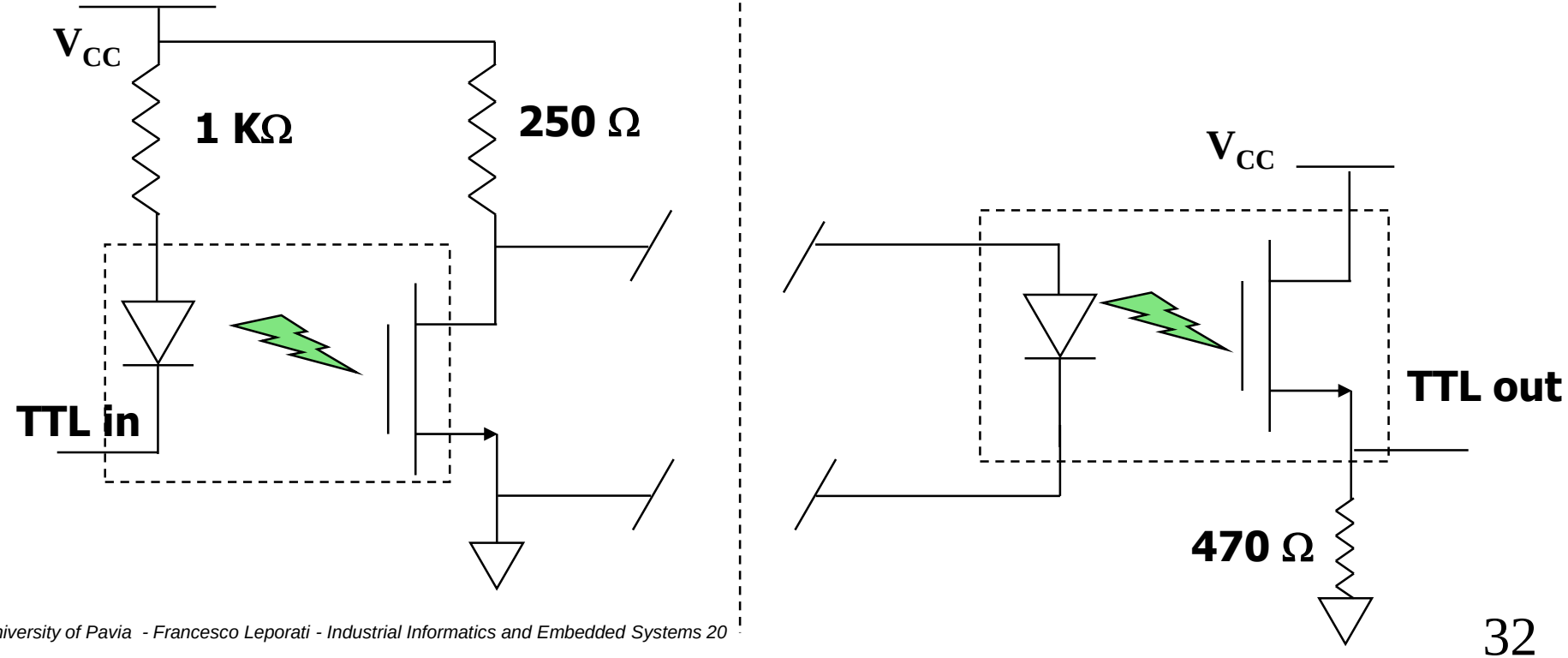
DIGITAL COMMUNICATION TECHNIQUES

- **Current loop 0-20 mA**

low baud rates (2400)

High noise immunity, miles allowed length (ok for industrial environment)

The transmitted signal is current (20 mA) not voltage



DIGITAL COMMUNICATION TECHNIQUES

- **Errors**

Recognition

– Error on a character: parity bits or autocorrecting Hamming code (corrects errors on M bits while recognizes them on N with N<M)

– Errors on messages: additional characters are present made up by Cyclic Redundancy Check evaluated on 16 bits and sent as 2 characters or checksum or longitudinal parity

Correction

– Hamming codes and CRC require hw complexity (i. e. devoted circuits) and redundancy. Used only in critical applications

– Re-transmission

– The message length depends on the foreseen error rate: if a re-transmission is required also this one could be eventually affected by the error thus the problem will be not solved and the communication cost will be very high

DIGITAL COMMUNICATION TECHNIQUES

The Cyclic Redundancy Check associates bit streams to polynomial with 0 and 1 coefficients. A K bit string is read as a polynomial with K coefficients with degree <= K-1. For instance:

$$101011 \rightarrow x^5+x^3+x+x^0$$

The used arithmetic is *module 2*, without carries and borrows. Transmitter and receivers must agree on a G(X) common polynomial, said *generating polynomial*. The idea is to integrate the sequence with a suitable *checksum* making the final polynomial divisible by G(x). For instance:

- add r bit set to 0 to each bit frame so having a m+r bit sequence that corresponds to the polynomial $x^r \times M(x)$, with r rank of G(x), M(x) is the polynomial corresponding to the original frame and m its relative bit number. We will have 101011**000.....0** (r zeros)
- $x^r \times M(x)$ must be divided (module 2) by G(x): $x^r \times M(x)=G(x) \times Q(x)+R(x)$ the coefficients in R(x) (remainder) will constitute the bit sequence of the checksum.
- Now, the remainder R(x) is added (module 2) to $x^r \times M(x)$ achieving the new bit frame that must be transmitted provided also with the right checksum.

The generator of cyclic codes cannot be a whatever polynomial. Generally speaking it is possible to obtain the same R(x) starting from different M(x) dividend polynomials. All those polynomials that give the same R(x) when divided by the same G(x) generator are defined as *congruous module G(x).* This approach does not detect those errors that transforms the transmitted bit sequence in a different one that however replicate a congruous polynomial. In any case it is a very efficient method for error recognition.

DIGITAL COMMUNICATION TECHNIQUES

•RS 232 C STANDARD INTERFACE (1969)

Cannon connector with 25 or 9 poles

Allowed length from few until tenths of meters

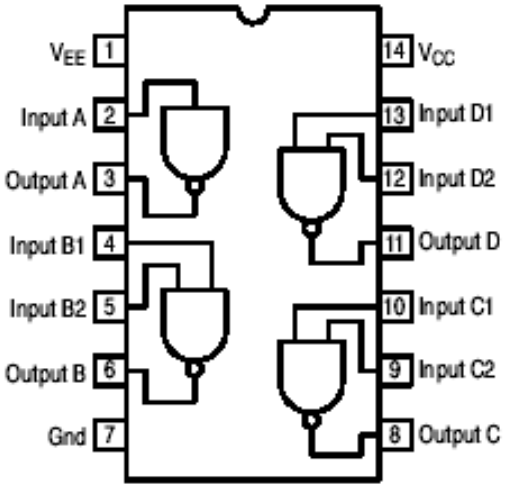
Negative logic voltage levels: "0" (3V ÷ 25V) / "1" (-3V ÷ -25V). Usually +12 V e -12 V are used. Wide noise immunity range but not ground reference

TTL signals must be converted in voltage levels compliant with RS232 values: line drivers and receivers must be employed

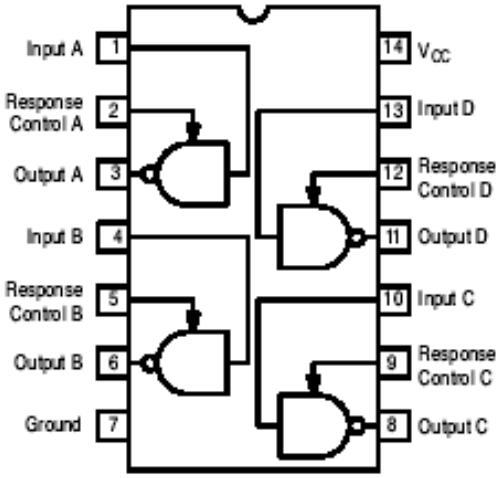
Line driver

1488

PIN CONNECTIONS



PIN CONNECTIONS



Line receiver

1489

DIGITAL COMMUNICATION TECHNIQUES

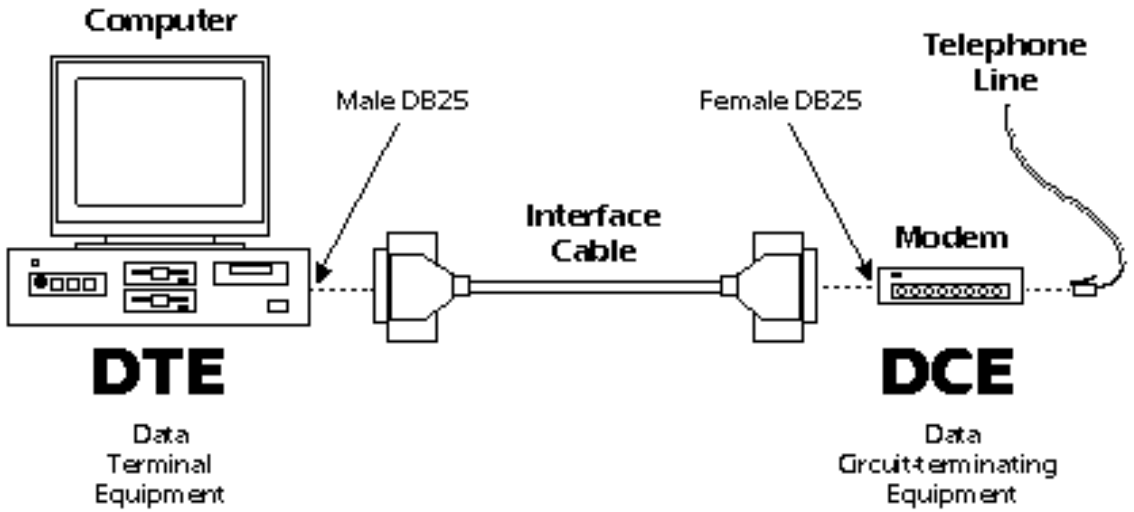
- **RS 232 C STANDARD INTERFACE (1969)**

Proposed for the first time in ’60s by Electronic Industry Association (EIA) and then upgraded several times (last in ‘97) but still used in industrial manufacturing

The connection model is among so-called Data Terminal Equipments (DTE) (typically computers connected through modems) and DCE - Data Communication Equipments (peripherals connected through modems)

Half duplex and full duplex communications

Null modem version as direct connection between DTE devices



DIGITAL COMMUNICATION TECHNIQUES

- **RS 232 C PINOUT**

1-7: ***Signal ground (AB)***

2: ***Transmitted data (TxD)***: data output

3: ***Received data (RxD)***: data input

4: ***Request-to-send (RTS)***: DTE ready to transmit (out in DTE and in DCE)

5: ***Clear-to-send (CTS)***: DCE ready to receive (out in DCE and in DTE)

6: ***DCE ready (DSR)***: modem ready to start communication

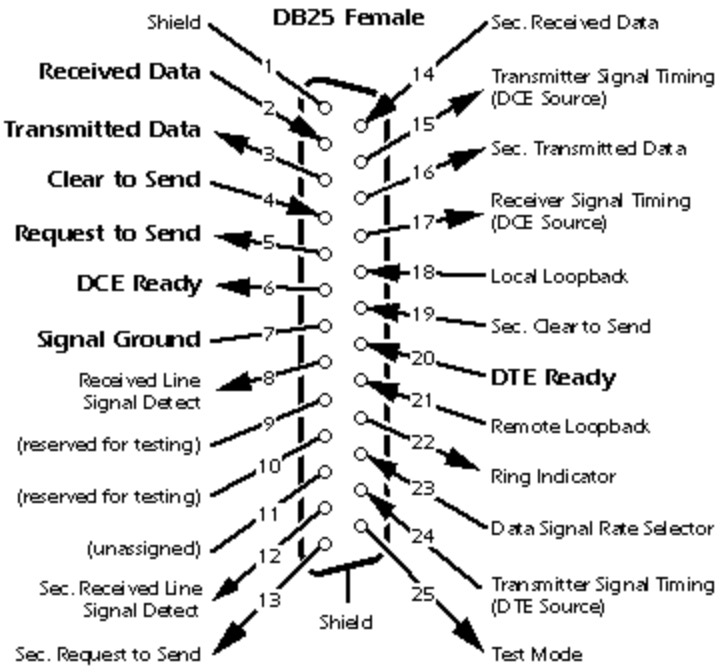
8: ***Received line signal detector (CD)***: a remote transmitting modem is detected

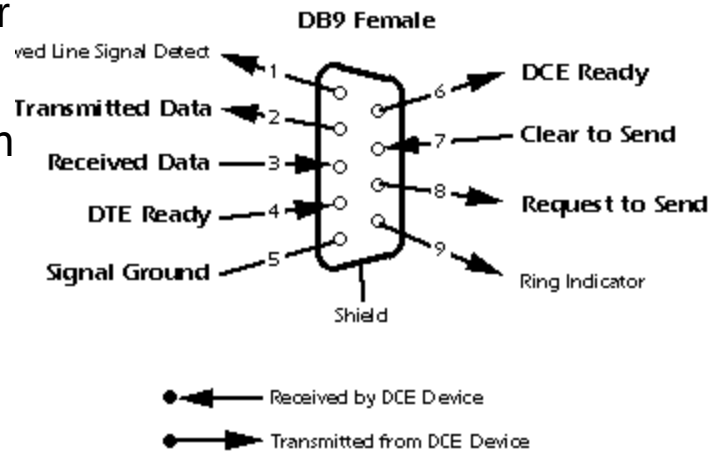
15: ***TXC*** (Transmit Clock): clock generated by the transmitter (in case of isochronous synchronization)

17: ***RXC*** (Receive Clock): clock generated by the receiver (in case of isochronous synchronization)

20: ***DTE ready (DTR)***: DTE ready to receive

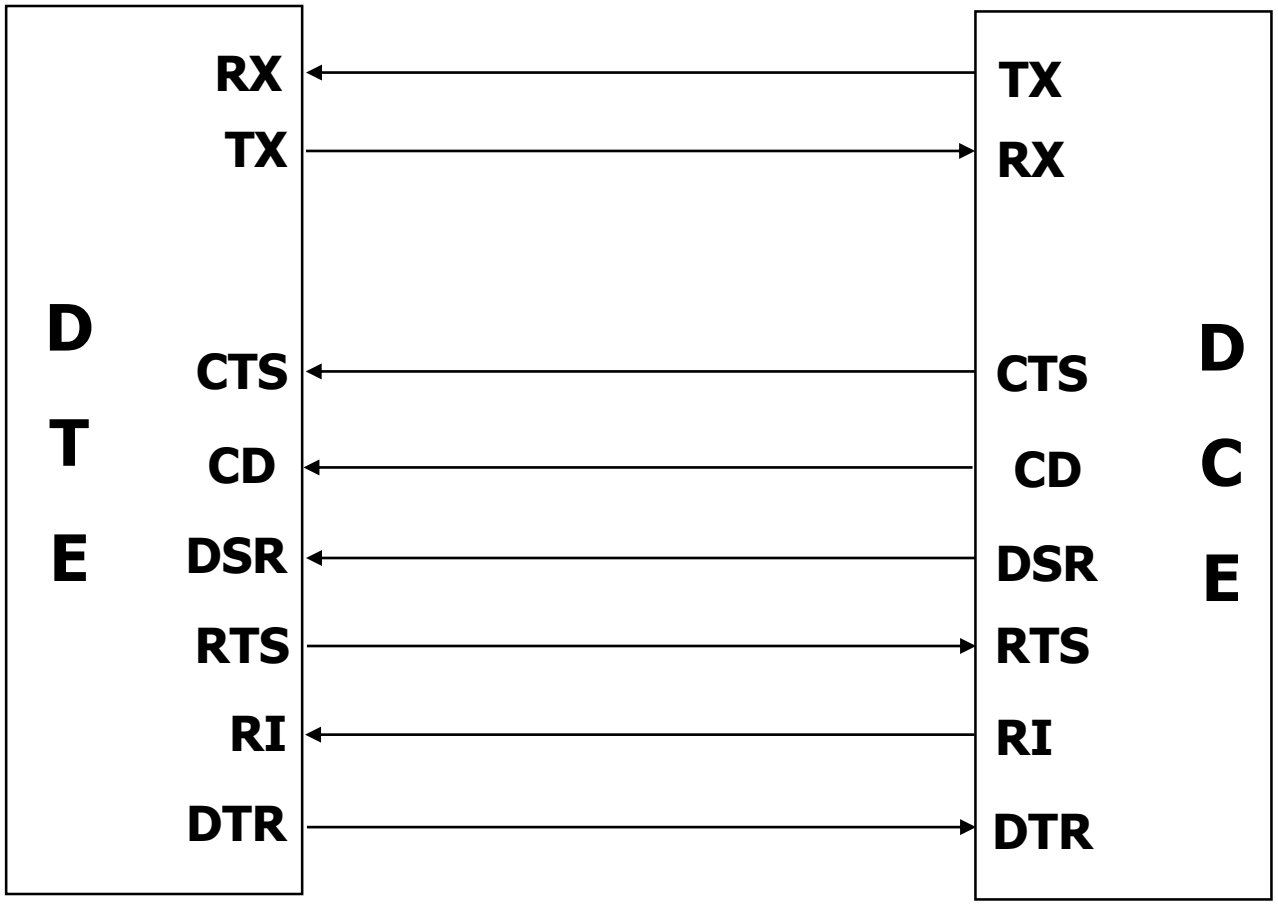
22: ***Ring Indicator (RI)***: modem alerts DTE about a call





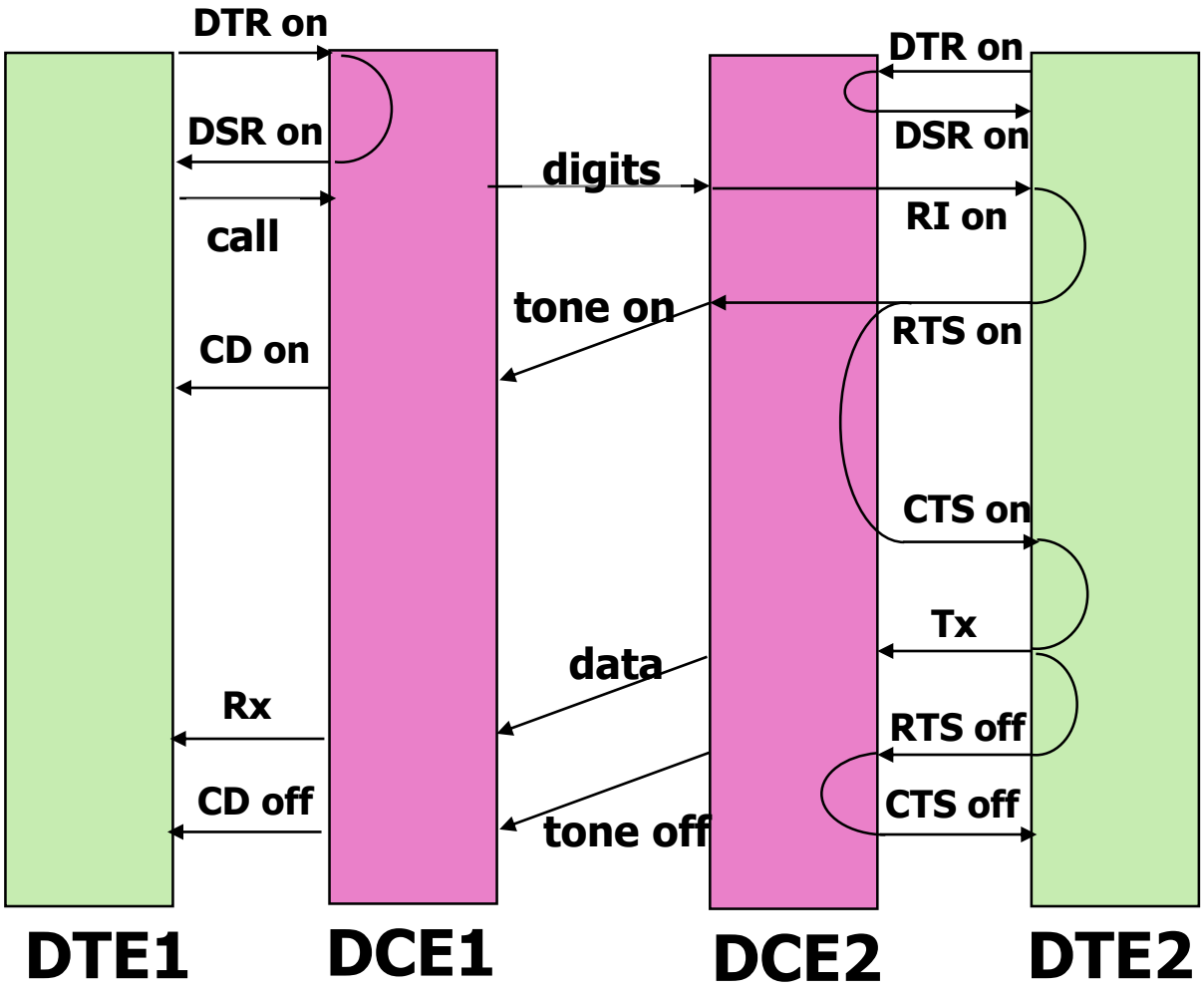
DIGITAL COMMUNICATION TECHNIQUES

- **DTE-DCE connection**



DIGITAL COMMUNICATION TECHNIQUES

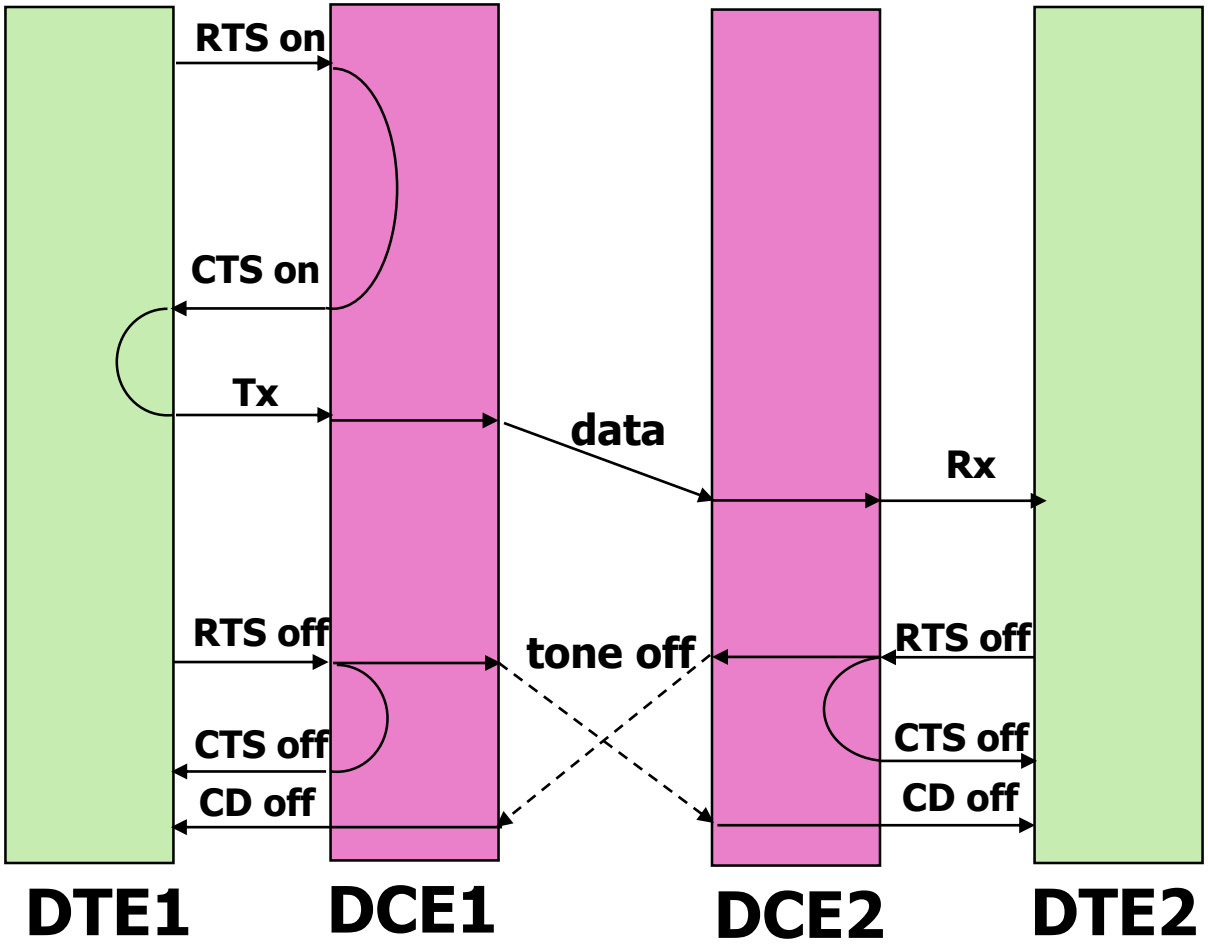
- **Setup call**



- **Half duplex mode**
- **DTE2 available**
- **DTE1 calls**
- **DCE2 alerts DTE2**
- **DTE2 confirms (handshake RTS-CTS)**

DIGITAL COMMUNICATION TECHNIQUES

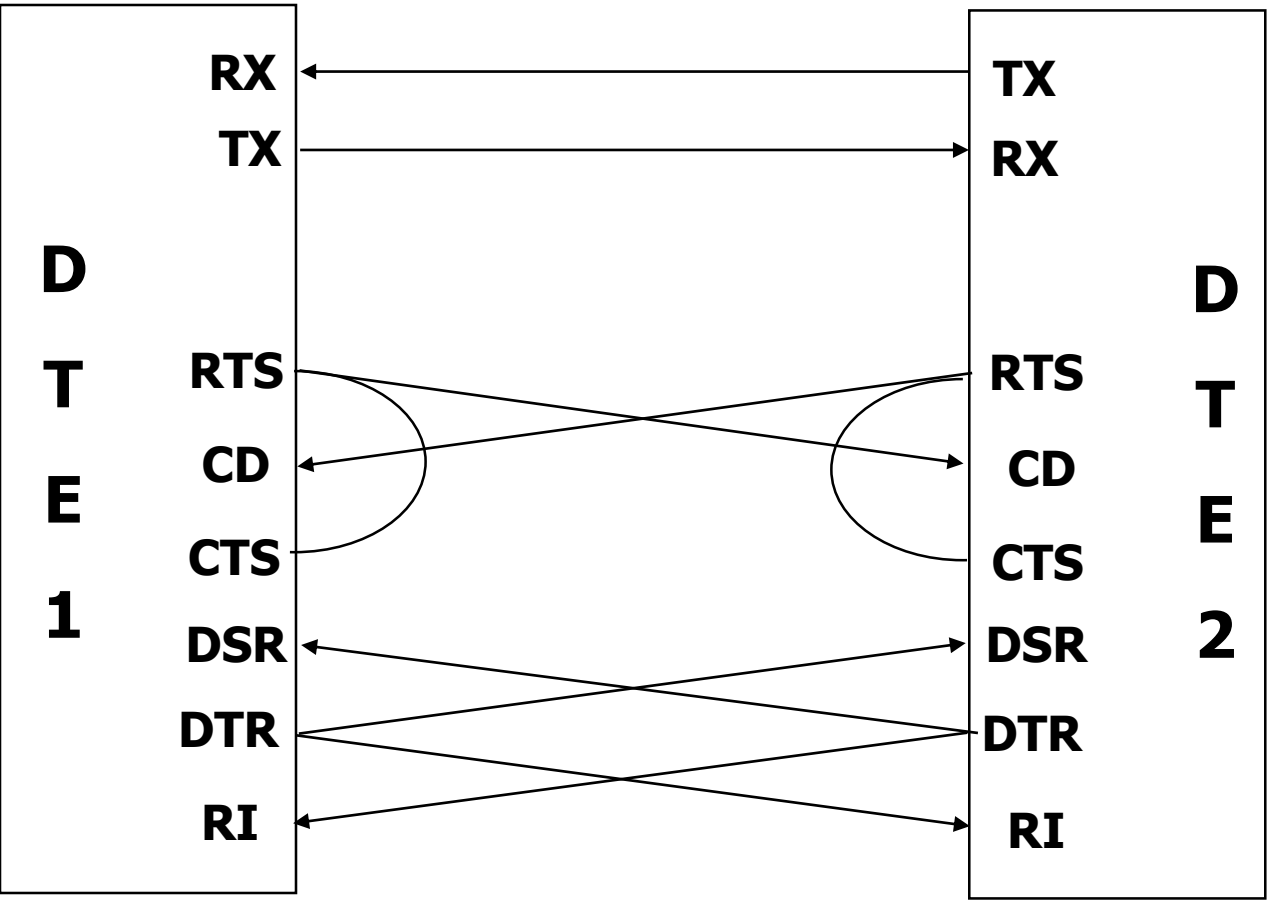
- **Data transfer and end of the call**



- **DTE1 answers (handshake RTS-CTS) …**
- **… and sends those data of which the communication consists**
- **At the end, one of the DTEs closes the connection with a closing handshake so terminating the communication session**

DIGITAL COMMUNICATION TECHNIQUES

- **Null-modem half duplex connection (if DTE close)**

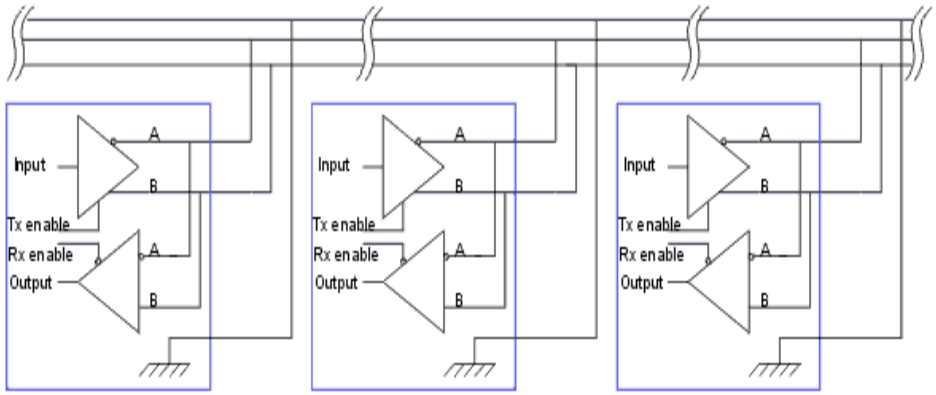


- On the contrary, if full duplex is selected only DTR/DSR/CD and not RTS/CTS are used. DTE1 verifies if DSR of DTE2 is active (ready to work). If yes, DTE1 samples the CD pin to verify the eventual presence of a carrier signal on the line (the partner is trying to communicate) otherwise it starts to communicate.
- If the isochronous temporization is selected pins 15 (TXC) and 17 (RXC) must be used. The transmission clock is used to generate data in isochronous mode, travels with them and will be brought on the RXC pin of the recipient.

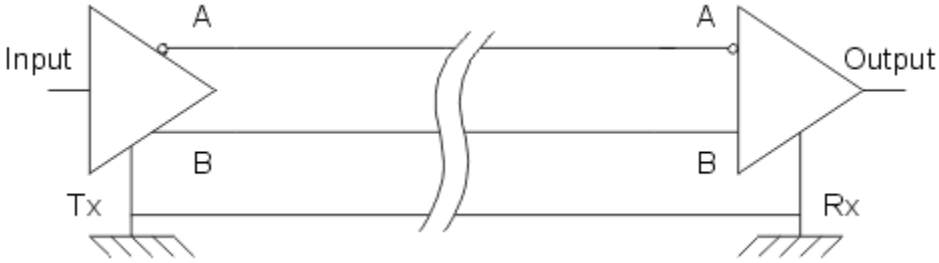
DIGITAL COMMUNICATION TECHNIQUES

•RS232 final considerations

- **Popular variants of the RS232 are RS422 and RS485.**
- **In the first case velocities up to 10 Mbit/sec are achieved on distances close to the Km.**
- **The transmission is in "differential" mode that is using voltage levels slightly different from 232 (A<B "1" e A>B "0") therefore level converters must be introduced.**
- **Up to 10 recipients connected to a single transmission driver are allowed.**



RS485 link schematization



RS422 link schematization

- **The RS 485 is very similar to the 422 from the electric point of view, but allows the coexistence of several couples Tx/Rx on the same transmission line (multi-point), although only one transmitter at a time can be active.**
- **Up to 32 recipients are allowed.**
- **Velocities within 100 Kb/sec and 10 Mb/s.**
- **Distances within 1200 m and 15 m.**
- **Used when connecting PC, PLC and instrumentations in a bus topology.**

DIGITAL COMMUNICATION TECHNIQUES

Characteristics of RS485 compared to RS232, RS422 and RS423

	RS232	RS423	RS422	RS485
Differential	no	no	yes	yes
Max number of drivers	1	1	1	32
Max number of receivers	1	10	10	32
Modes of operation	half duplex full duplex	half duplex	half duplex	half duplex
Network topology	point-to-point	multidrop	multidrop	multipoint
Max distance (acc. standard)	15 m	1200 m	1200 m	1200 m
Max speed at 12 m	20 kbs	100 kbs	10 Mbs	35 Mbs
Max speed at 1200 m	1 kbs	1 kbs	100 kbs	100 kbs
Max slew rate	30 V/µs	adjustable	n/a	n/a
Receiver input resistance	3..7 kΩ	4 kΩ	4 kΩ	12 kΩ
Driver load impedance	3..7 kΩ	450 Ω	100 Ω	54 Ω
Receiver input sensitivity	±3 V	±200 mV	±200 mV	±200 mV
Receiver input range	±15 V	±12 V	±10 V	–7..12 V
Max driver output voltage	±25 V	±6 V	±6 V	–7..12 V
Min driver output voltage (with load)	±5 V	±3.6 V	±2.0 V	±1.5 V

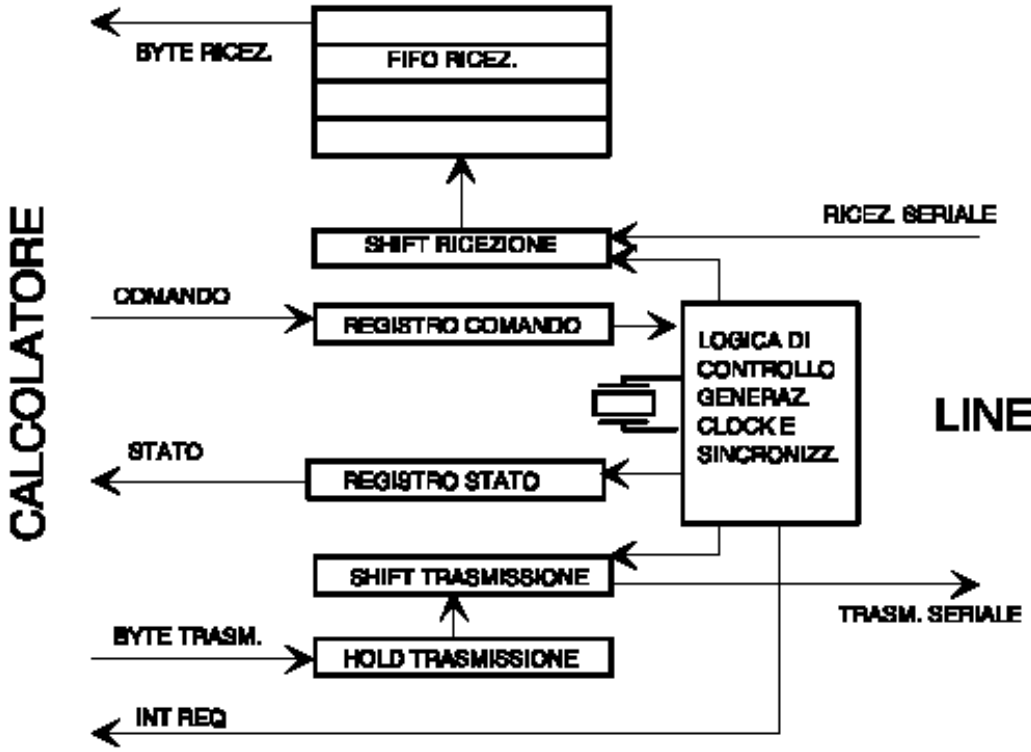
DIGITAL COMMUNICATION TECHNIQUES

USART (Universal Synchronous Asynchronous Receiver Transmitter)

- **IC carrying out synchronous/asynchronous serial transmission**
- **Sometimes embedded within the microcontroller ⇒ single chip computer**
- **UART if the transmission is only asynchronous**
- **The device implements all the transmission/reception tasks, temporization, parity and correctness check …, while the voltage level adaptation is done through suitable drivers/receivers as already mentioned (1488/1489)**
- **Basic structure made up by:**
  - **transmission and reception buffers**
  - **transmission and reception shift registers**
  - **mode, command, state registers**
- **COTS standard devices: NS8250 e NS16550**

DIGITAL COMMUNICATION TECHNIQUES

USART : block scheme



A character written in the *transmission register* triggers the beginning of the transmission

The reception is performed by reading the *receiving register* where data are valid if suitable error flags are not active in the *status register*

The *status (state) register* bits indicate:

- **if synchronization has been done**
- **if the transmission buffer is empty**
- **if transmission shift reg. is empty**
- **if a character is in reception buffer**
- **errors (parity, framing, overrun)**
- **interrupt**

***Command reg.:* reset state indicators, enable interrupt, set line signals (RTS, DSR, …)**

***Mode reg.:* set of the comm. velocity, parity, synchronization characters**

DIGITAL COMMUNICATION TECHNIQUES

UART 8250: registers

8 registers in 8 consecutive address areas of the I/O memory space (base address = 3F8H for Com1 and 2E8H for Com2 ports)

some registers can be used in sole reading mode or in sole writing mode if the bit DLAB in LCR is zero or in both the R/W modes if the bit DLAB=1

Address	Read	Write	Read	Write
base	RBR (rec. buf.)	THR (transm. hold)	DLL (divisor latch LSB)	
base+1	IER (int. enab.)	IER	DLM (divisor latch MSB)	
base+2	IIR (int. identif.)	FCR (fifo control)	IIR	FCR
base+3	LCR (line control, R/W independently on DLAB)			
base +4	MCM (modem control, R/W independently on DLAB)			
base+5	LSR (line status)	/	LSR	/
base+6	MSR (modem status)	/	MSR	/
base+7	SCR (scratch, R/W independently on DLAB)			

DIGITAL COMMUNICATION TECHNIQUES

UART 8250: registers

Baud Rate	High Byte	Low Byte
110	04H	17H
300	01H	80H
1200	00H	60H
2400	00H	30H
4800	00H	18H
9600	00H	0CH
19200	00H	06H
38400	00H	03H
57600	00H	02H
115200	00H	01H

DLR = Divisor Latch Register

16 bit registers used to set the baud rate on the basis of the values written in BA e BA+1

b7 . . b4 - Reserved always zero
b3 - Modem Status Interrupt
b2 - Receiver Line Status Interrupt
b1 - Transmitter Holding Register Empty Interrupt
b0 - Received Data Available Interrupt

IER = Interrupt Enable Register

enable UART to generate interrupt through setting of the correspondent bits

IIR = Interrupt Identification Register

the bit allows to identify the origin of the interrupt service request

b2	b1	What they say	How to reset
1	1	Receiver Line Status	Read LSR
1	0	Received Data Available	Read RBR
0	1	Transm. Hold. Register Empty	Read IIR or write THR
0	0	Modem Status	Read MSR

b0= int. pending; b3= int. time out

DIGITAL COMMUNICATION TECHNIQUES

FCR = Fifo Control Register

8 bit register used to setup the Fifo memories present in the 16550 device: in particular b0 enables the Fifo, b1 and b2 empty reception and transmission Fifos, b6 and b7 define the n° of bytes required to set the full Fifo flag (1, 4, 8, 14)

LCR = Line Control Register

8 bit register to set different functionalities

b7 (DLAB) = if 1 allows to access DLR, if 0 to access RBR, THR, IER

b6 = if 1, a constant voltage level corresponding to that of a start bit is put on the line to signal a break of transmission.

b4, b3 = set the parity

b2 = set the number of stop bits (1, 2)

b1, b0 = set the character length (5, 6, 7, 8 bit)

MCR = Modem Control Register

8 bit register used to set the RTS and DTR signal output from the UART:

b2 = global enable of the UART to interrupt

b1, b0 = if 1 each one activates RTS and DTR

DIGITAL COMMUNICATION TECHNIQUES

LSR = Line Status Register

8 bit register providing info about the transmission status

b5 = 1 if the THR register is empty. The corresponding interrupt is activated, if enabled

b4 = 1 if a low level is detected on the line (Break) longer than a character

b3 = 1 if after the character bits and the parity the expected stop bit is not detected (Framing Error)

b2 = 1 if after the character bits the parity is not correct (Parity Error)

b1 = 1 if a character in RBR has been overwritten before being read (Overrun Error)

b0 = 1 if a new, not read character is available in RBR (becomes 0 after reading)

MSR = Modem Status Register

8 bit register used to detect the values of the CD, RI, DSR e CTS (b7-b4) signals or if they were switched (b3-b0)

SCR = Scratch Register

Service register to read/write temporary values

DIGITAL COMMUNICATION TECHNIQUES

USART sw management

Polling

- The LSR is periodically polled: a new received character or a character to be transmitted or error will be signaled
- Low-medium communication velocities and busy waiting

Interrupt

- Each of the three previous situations generates an interupt request (character reception, character transmission, error)
- IIR is read to adequately serve the interrupt
- Medium transmission velocities (9600 baud)

DMA

- controllers or coprocessors devoted to handshake between memory and USART
- velocities in the order of MB/sec
- Not always provided: only in expensive devices

DIGITAL COMMUNICATION TECHNIQUES

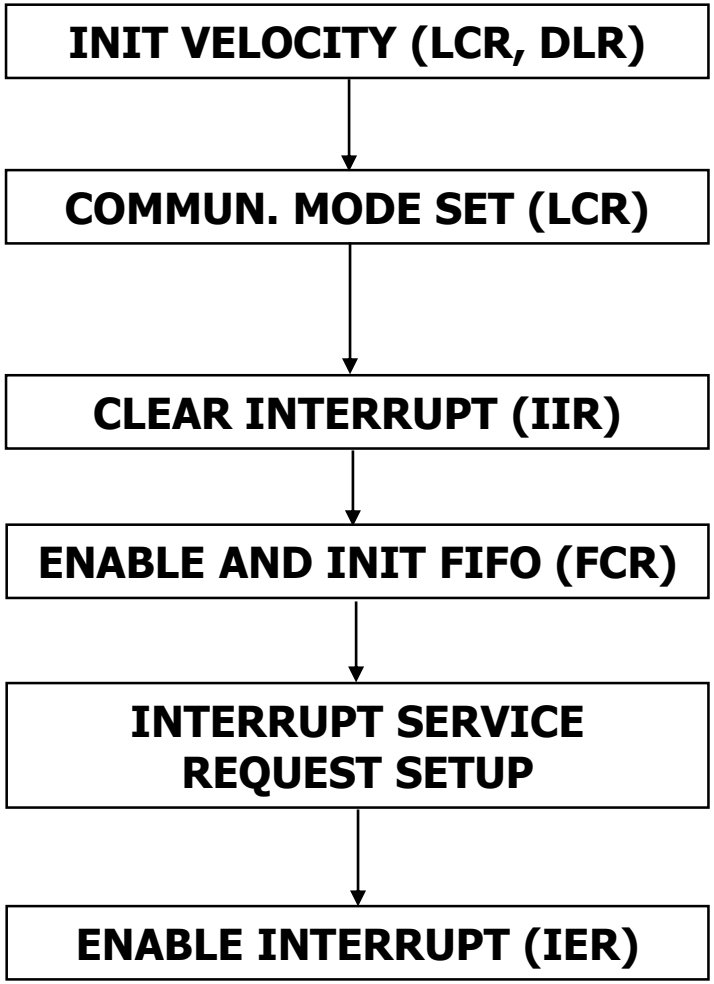
USART sw management

Init registers addresses and main masks for bits

```
BA     EQU   3F8H          ;indirizzo base

RBR    EQU   BA+0          ;dato ricevuto
THR    EQU   BA+0          ;dato da trasmettere

DLRL   EQU   BA+0          ;seleziona baud L
DLRH   EQU   BA+1          ;seleziona baud H
B1200  EQU   0060H         ;valore per 1200 baud
B9600  EQU   000CH         ;valore per 9600 baud

IER    EQU   BA+1          ;reg. abilitaz. interrupt
MKEIT  EQU   2             ;maschera int. trasmissione
MKEIR  EQU   1             ;maschera int. ricezione

IIR    EQU   BA+2          ;reg. tipo interrupt
MKI    EQU   6             ;maschera bit di tipo

FCR    EQU   BA+2          ;reg. controllo FIFO
MKF1   EQU   0             ;int. ogni byte
MKF4   EQU   04H           ;int. ogni  4 byte
MKF8   EQU   08H           ;           8 byte
MKF14  EQU   0CH           ;           14 byte
MKFRT  EQU   04H           ;mschera reset FIFO trasm.
MKFRR  EQU   02H           ;maschera reset FIFO ricez.
MKFEN  EQU   01H           ;maschera abilita FIFO

LCR    EQU   BA+3          ;reg. controllo linea
MKDLAB       EQU   80H
STDLINE      EQU   1BH     ;impostaz. standard pari, 1 stop, 8 bit

MCR    EQU   BA+4

LSR    EQU   BA+5          ;reg. stato linea
MKTHRE       EQU   20H     ;maschera THRE
MKBRK        EQU   10H     ;maschera Break
MKFRE        EQU   08H     ;maschera framing err.
MKPAE        EQU   04H     ;maschera parity err.
MKOVE        EQU   02H     ;maschera overrun err.
MKDRDY       EQU   01H     ;maschera dato ricevuto ready

MSR    EQU   BA+6
```

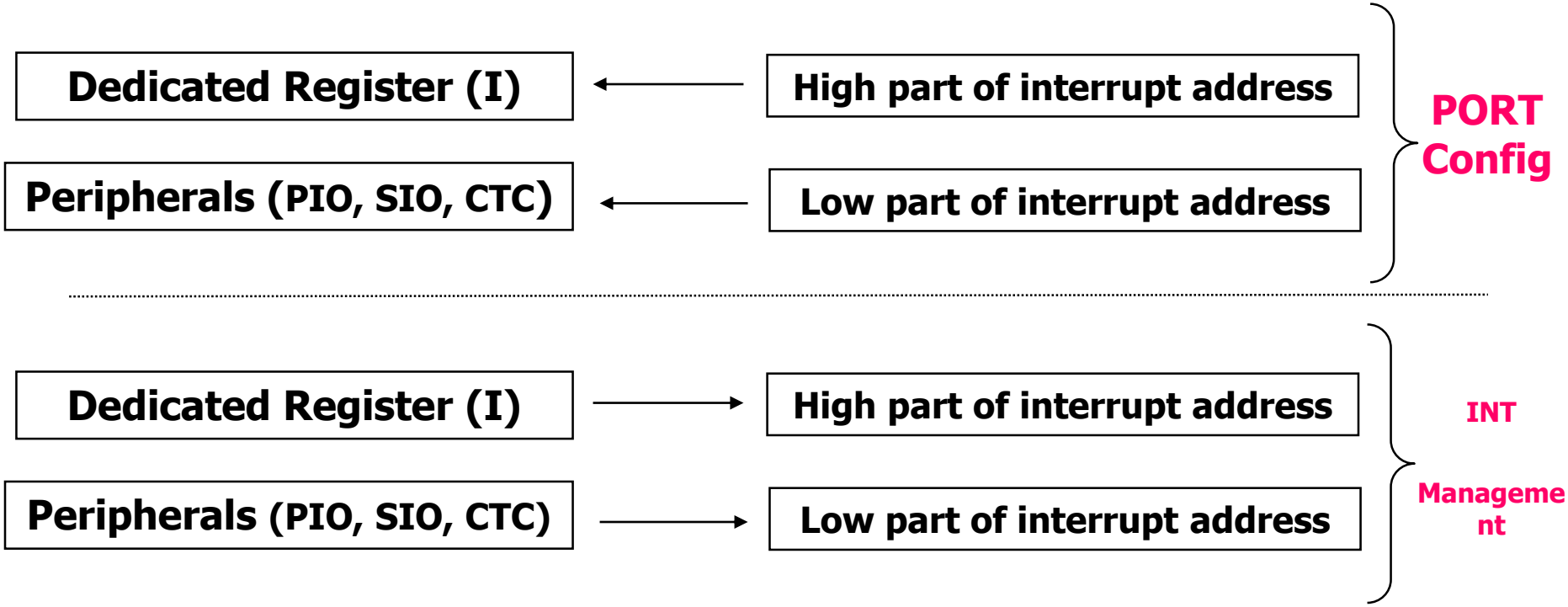
UART setup

```
;imposta velocita' di comunicazione
      MOV   AL, MKDLAB
      MOV   DX, LCR
      OUT    [DX],AL
      MOV   AX, B9600
      MOV   DX, DLRL
      OUT    [DX],AL
      INC   DX
      OUT    [DX],AH
;imposta modalita' di comunicazione
      MOV   AL, STDLINE
      MOV   DX,LCR
      OUT    [DX],AL
;cancella eventuali interrupt pendenti
      MOV   DX, IIR
      IN    AL,[DX]
;inizializza la FIFO e la abilita
      MOV   AL, MKFRT+MKFRR+MKFEN
      MOV   DX, FCR
      OUT    [DX],AL
;aggancia la routine di servizio interrupt
      CALL  AGGANCIA_INT_UART
;abilita UART a generare richieste di interrupt
      MOV   AL, MKEIT+MKEIR
      MOV   DX, IER
      OUT    [DX],AL
```



DIGITAL COMMUNICATION TECHNIQUES

Aggancia_INT_UART



In memory:

High part of interrupt address +0: interrupt service peripheral 1

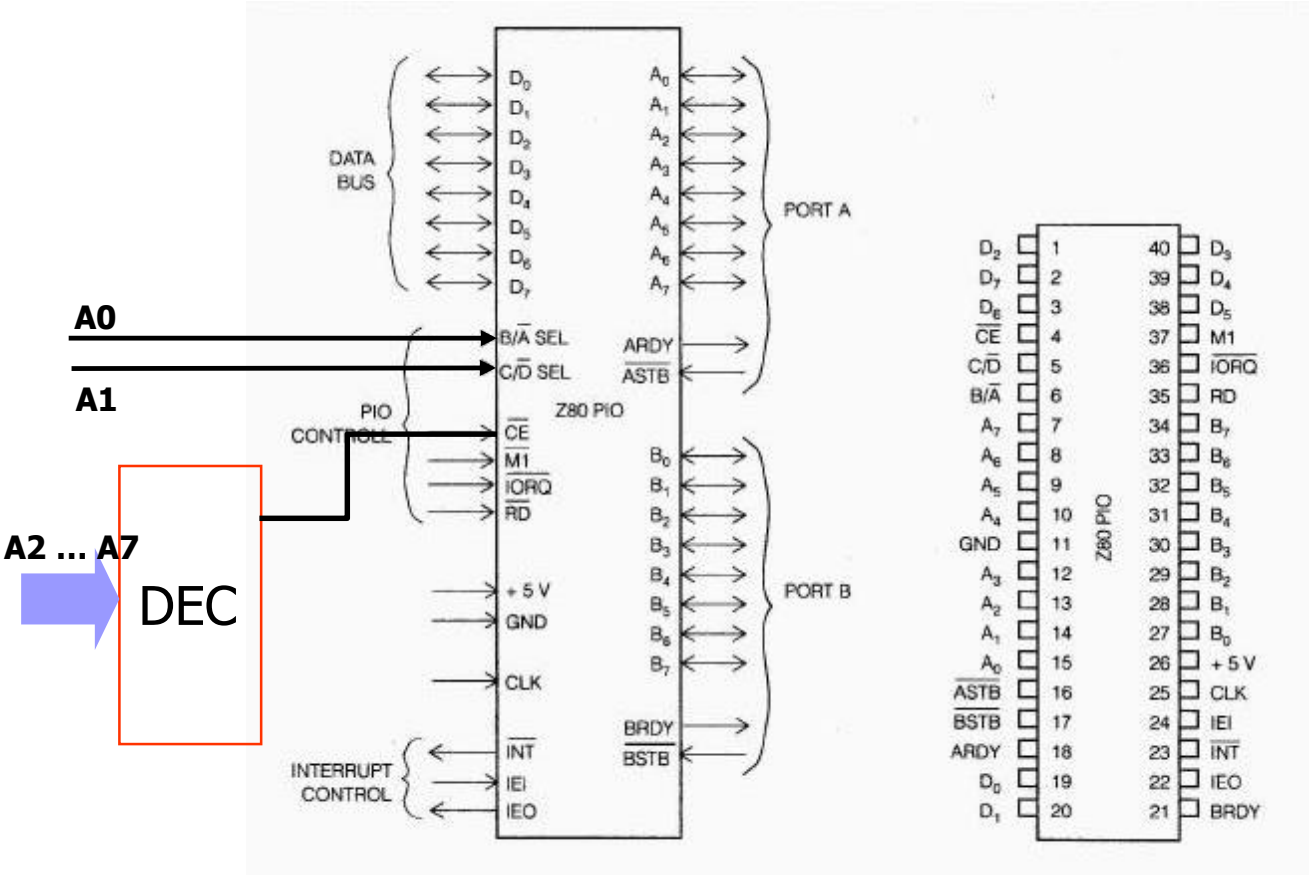
High part of interrupt address +2: interrupt service peripheral 2

.....

High part of interrupt address +2k: interrupt service peripheral k

DIGITAL COMMUNICATION TECHNIQUES

- The parallel port of a typical microprocessor



B/A: select port B o A

CE: chip enable

M1: fetch active

IEO/IEI: interrupt en.

A/B RDY: port ready to communicate

A/B STB: peripheral ready to communicate

INT: int. request

IORQ: mem. mapped I/O

C/D: command or data

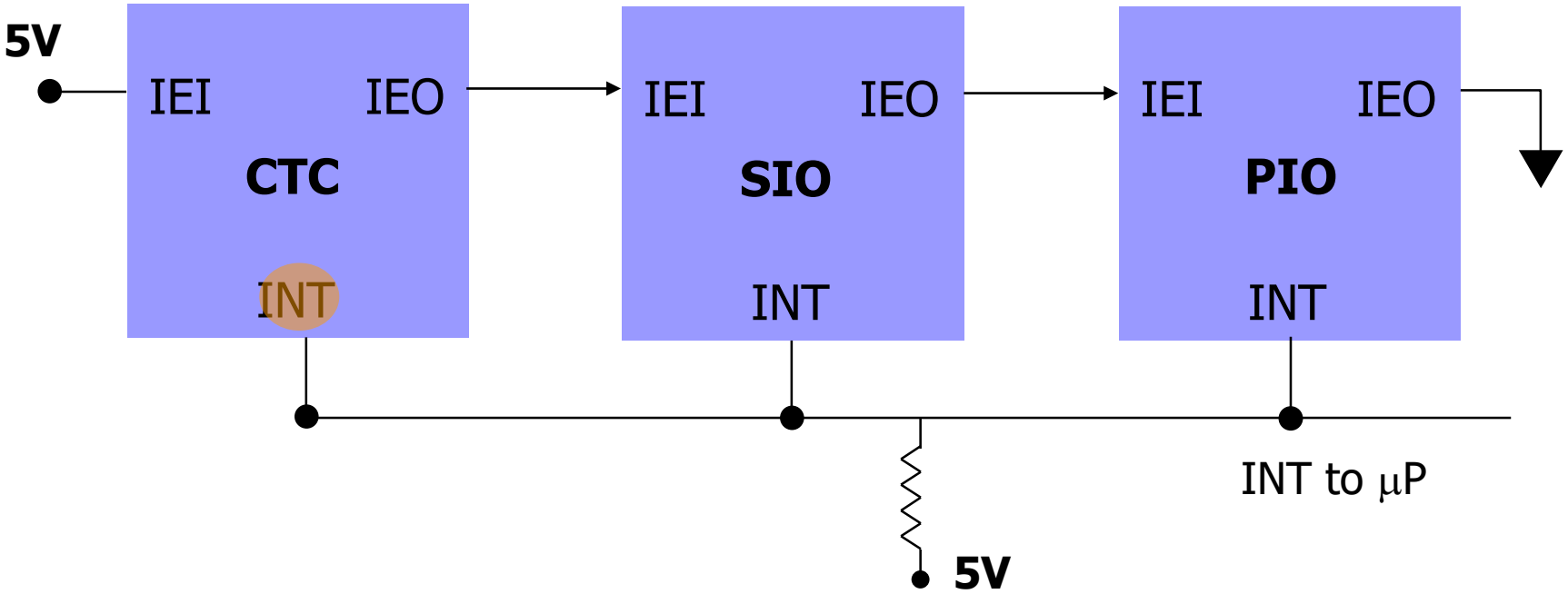
RD: read o write

A0-A7: out port A

B0-B7: out port B

DIGITAL COMMUNICATION TECHNIQUES

- A hardware priority chain: *daisy chain* (priority diminishes from left to right)

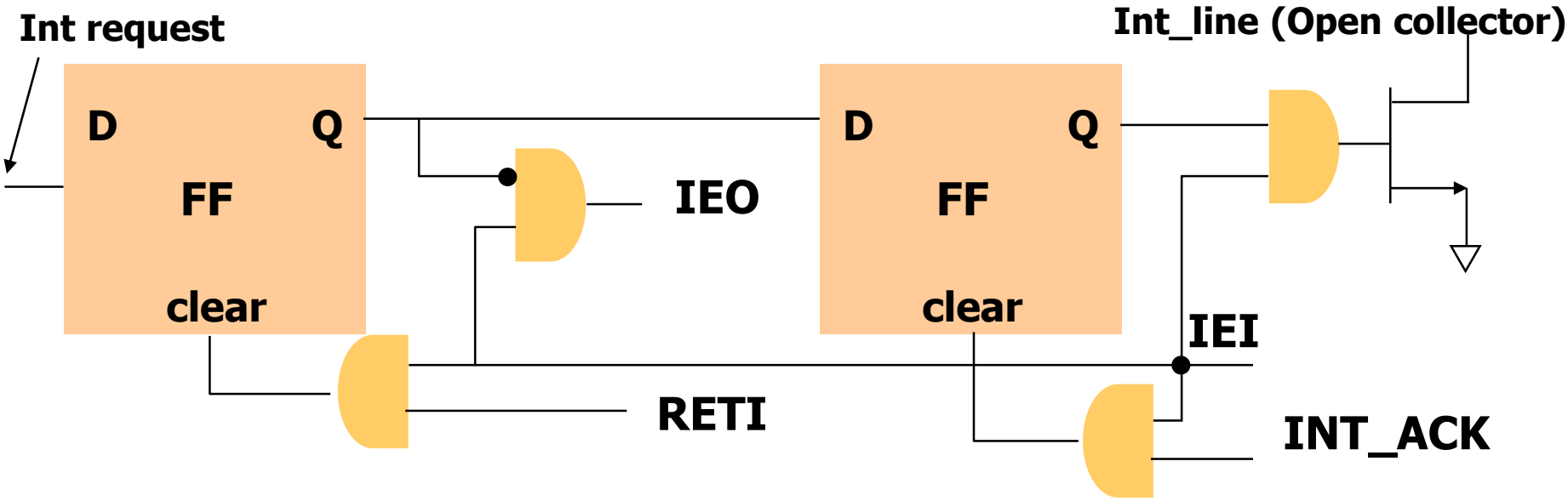


When a peripheral (parallel, serial, timer) generates an interrupt the μP can recognize the request and sets M1 and IORQ lines simultaneously at zero (IntAck).

The peripherals, looking two lines at zero, will put on the bus the low part of the interrupt vector (specific for itself). The PIO must be connected with the M1 e IORQ lines to know that the microprocessor acknowledged the interrupt.

DIGITAL COMMUNICATION TECHNIQUES

- Interrupt request management on each peripheral connected in daisy chain



- The first FF receives a service request that is passed to the II FF only if the IEI signal is active. In this case the II FF forces the MOS commutation by activating the interrupt line (supposed low) that is managed in a open collector fashion.
- Int_ack clears only the port enabled to request (who has been already serviced must not require anymore while who is waiting keeps its request still active).
- The RETI instruction (executed when the interrupt routine ends) clears the first FF (the one that maintains its request active).

DIGITAL COMMUNICATION TECHNIQUES

•**PIO port programming (modes 0, 1, 2)**

A port can be configured to work in mode 0 (output only), 1 (input only), 2 (bidirectional) or 3 (bit by bit). This is possible by writing suitable bytes to the port that however must be set in "command" (configuration) mode (C/D=1).

The first three modes require a control word into which the configuration bits setting mode 0, 1 or 2 are coded in bits D7 and D6

M2	M1	X	X	1	1	1	1

Then the low part of the interrupt vector (address) will be sent to the port (that is the value given back by the port itself on the data bus when it will require an interrupt service).

Finally it will be possible to disable the interrupt for that port by sending the following word with the bit 7 set to 1 (enable) or to 0 (disable).

E/D	X	X	X	0	0	1	1

DIGITAL COMMUNICATION TECHNIQUES

•PIO port programming (mode 3)

In mode 3 the same procedure is followed as before that sends a *mode control* word and the interrupt vector to the port.

Then a word is sent that specifies what bits are in input (set to 1) and what in output (set to 0); for instance below the bits 7-6-3-2-1-0 are set as input and 5-4 as output.

1	1	0	0	1	1	1	1

The *interrupt control word* features the following format into which the bit D4 specificies if inputs must be masked or not i. e. if all of them (0) or only few (1) will cause interrupt service request. In this case a mask will follow that indicate what bit will generate the interrupt (0) and what ones must be ignored.

1/0	1/0	1/0	1/0	0	1	1	1

The bit D5 indicated the interrupt activation edge (0 falling / 1 raising), D6 if the generating condition is the OR or the AND of more bits, and D7 if the interrupt request is enabled or not.

DIGITAL COMMUNICATION TECHNIQUES

```
                        NAME MAIN
4  0000
6               LIST E,O,S,T
7  0000
8               INCLUDE COST.H   ; equates
8  0030          TABINT  EQU    30H  ;most sign. byte in interr. address
8  0040          PIO1A   EQU    40H
8  0042          PIO1AC  EQU    42H
8  0041          PIO1B   EQU    41H
8  0043          PIO1BC  EQU    43H
8  0060          PIO2A   EQU    60H
8  0062          PIO2AC  EQU    62H
8  0061          PIO2B   EQU    61H
8  0063          PIO2BC  EQU    63H
8  0000
8  000F          MODO0   EQU    0FH
8  004F          MODO1   EQU    4FH
8  008F          MODO2   EQU    8FH
8  00CF          MODO3   EQU    0CFH
8  0007          DISINT  EQU    07H  ;disable interrupt
8  0083          LIN2A   EQU    83H  ;ISOL\, REEDB input, ADC-BSY input
8  00A7          CNINTP  EQU    0A7H  ;int. abil., OR, high edge active
8  00FE          MASK    EQU    0FEH  ;the line D0 only activates interrupt
8  0000          CTCCH0  EQU    00H  ;channel 0 CTC
8  0001          CTCCH1  EQU    01H  ;channel 1 CTC
8  0002          CTCCH2  EQU    02H  ;channel 2 CTC
8  0003          CTCCH3  EQU    03H  ;channel 3 CTC
8  0007          CTCCN2  EQU    07H  ;no int, timer mode, prescaler=16, whatever edge
8                                     ;no trigger, time constant follows, reset
8  00C7          CTCCN3  EQU    0C7H ;enable int., counter mode, no prescaler, neg. edge
8                                      ;no trigger, time constant follows, reset
8  0017          SIOCNT  EQU    17H  ;control for SIO temporization
8                                    ;through CTC outputs, no int,. timer mode
8                                    ;16, positive edge, time constant follows, reset
8  0028          TIME2   EQU    40  ;CTC time constant channel 2 (9600 baud)
8  0028          TIME3   EQU    40  ;CTC time constant channel 3
8  0001          BAUD    EQU     1  ;temp. channels 0 and 1 CTC (9600 baud)
8  000A          LENGA   EQU    0AH  ;length tab. prog. SIOA
8  000C          LENGB   EQU    0CH ;lenght tab. prog. SIOB
8  0020          SIOA    EQU    20H
8  0021          SIOB    EQU    21H
8  0022          SIOAC   EQU    22H
8  0023          SIOBC   EQU    23H
8  00F0          SIOVEC  EQU    0F0H  ;tab. interrupt SIO
8  00FF          LCD_C   EQU    0FFH  ;address command/state LCD
8  00FE          LCD_D   EQU    0FEH  ;address dat LCD
30 0000
31                                                  CSEG
33 0000 F3                                          DI
34 0001
35 0001 CD BE 06   C                CALL   RITARDO
36 0004 31 FF F7                    LD     SP,STAC .....
47 0022 ED B0                       LDIR
48 0024 08                          EX     AF,AF'
49 0025 D9                          EXX
50 0026
51 0026 CD CB 03   C                CALL RETI          ;reset hw all peripherals (4)
52 0029 CD CB 03   C                CALL RETI
53 002C CD CB 03   C                CALL RETI
54 002F CD CB 03   C                CALL RETI
55 0032 ED 5E                       IM 2               ;vector interrupt mode set
56 0034 3E 30                       LD A,TABINT
57 0036 ED 47                       LD I,A
59              ; Programming PIO, CTC, SIO
60              ; inizialization PIO
61              ; ***********************************
62 0038 3E 07                       LD     A,DISINT
63 003A D3 42                       OUT    (PIO1AC),A
64 003C D3 43                       OUT    (PIO1BC),A
65 003E D3 63                       OUT    (PIO2BC),A     ;disab. interrupt on PIO1A, PIO2B, PIO1B
66 0040
67 0040 3E 0F                       LD     A,MODO0
68 0042 D3 63                       OUT    (PIO2BC),A
```

DIGITAL COMMUNICATION TECHNIQUES

```
 69  0044 D3 43                OUT    (PIO1BC),A
 70  0046 3E 30                LD     A,30H    ;READ e WRITE high to do not write spurious data
 71  0048 D3 41                OUT    (PIO1B),A
 72  004A
 73  004A 3E CF                LD     A,MODO3
 74  004C D3 42                OUT    (PIO1AC),A
 75  004E 3E FF                LD     A,0FFH
 76  0050 D3 42                OUT    (PIO1AC),A
 77  0052
 78  0052 3E E6                LD     A,0E6H       ; LSB interrupt routine
 79  0054 D3 62                OUT    (PIO2AC),A
 80  0056 3E CF                LD     A,MODO3       ; PIO 3 mode
 81  0058 D3 62                OUT    (PIO2AC),A
 82  005A 3E 83                LD     A,LIN2A       ;  what input/output lines
 83  005C D3 62                OUT    (PIO2AC),A
 84  005E 3E A7                LD     A,CNINTP      ; interrupt control
 85  0060 32 B1 02    D        LD     (PROG2A),A    ; port 2A re-programmed in interrupt
 86  0063 C6 10                ADD    A,10H         ; a mask will follow
 87  0065 D3 62                OUT    (PIO2AC),A
 88  0067 3E FE                LD     A,MASK        ; mask for those lines activating interrupt
 89  0069 D3 62                OUT    (PIO2AC),A
 90  006B
 91                ; init CTC
119                ; init SIO
120                ; ********************
147
148  00C6
149  00C6 FB          MAIN:    EI
257  0168 C9                   RET
513  0393
514
;*******************************************************************************
517  0393 08          PIOINT           EX     AF,AF'
518  0394 D9                   EXX
519  0395 3A B1 02    D        LD     A,(PROG2A)  ;reprogramming the port on the negative edge
520  0398 EE 20                XOR    20H          ; test bit D5 edge sensitive ....
521  039A D3 62                OUT    (PIO2AC),A
522  039C 32 B1 02    D        LD     (PROG2A),A   ; to successively program the port 2A
523  039F
524  039F ED 4B 11 02 D        LD     BC,(CIMP)      ;load pulse counter REEDB
525  03A3 3A CE 02    D        LD     A,(KEY)
526  03A6 CB 7F                BIT    PWRCLOSB,A
527  03A8 C2 AF 03    C        JP     NZ,TRY6
528  03AB 0B                   DEC    BC
529  03AC C3 B5 03    C        JP     CARICA
530  03AF CB 77          TRY6  BIT    PWROPENB,A
531  03B1 C2 B5 03    C        JP     NZ,CARICA
532  03B4 03                   INC    BC
533  03B5 ED 43 11 02 D  CARICA  LD    (CIMP),BC
534  03B9 D9                   EXX
535  03BA 08                   EX     AF,AF'
536  03BB FB                   EI
537  03BC ED 4D                RETI
538  03BE
....
1078                          ORG 30E6H
1079  30E6
1080  30E6 93 03       C        DEFW PIOINT
1081  30E8 12 05       C        DEFW EIRETI
1082  30EA 12 05       C        DEFW EIRETI
1083  30EC 12 05       C        DEFW EIRETI
1084  30EE 69 01       C        DEFW CTCINT
1085  30F0 CD 03       C        DEFW TRASB
1086  30F2 D9 03       C        DEFW EXTB
1087  30F4 E3 04       C        DEFW RICEB
1088  30F6 0A 05       C        DEFW SPECB
1089  30F8 BE 03       C        DEFW TRASA
1090  30FA E4 03       C        DEFW EXTA
1091  30FC EF 03       C        DEFW RICEA
1092  30FE EA 04       C        DEFW SPECA
1093  3100
```

DIGITAL COMMUNICATION TECHNIQUES

```
                        NAME MAIN
 4  0000
 6                      LIST E,O,S,T
 7  0000
 8                      INCLUDE COST.H   ; equates
 8  0030                TABINT  EQU    30H
;most sign. byte in the address of interrupt
tab.
 8  0040                PIO1A   EQU    40H
 8  0042                PIO1AC  EQU    42H
 8  0041                PIO1B   EQU    41H
 8  0043                PIO1BC  EQU    43H
 8  0060                PIO2A   EQU    60H
 8  0062                PIO2AC  EQU    62H
 8  0061                PIO2B   EQU    61H
 8  0063                PIO2BC  EQU    63H
.....
```

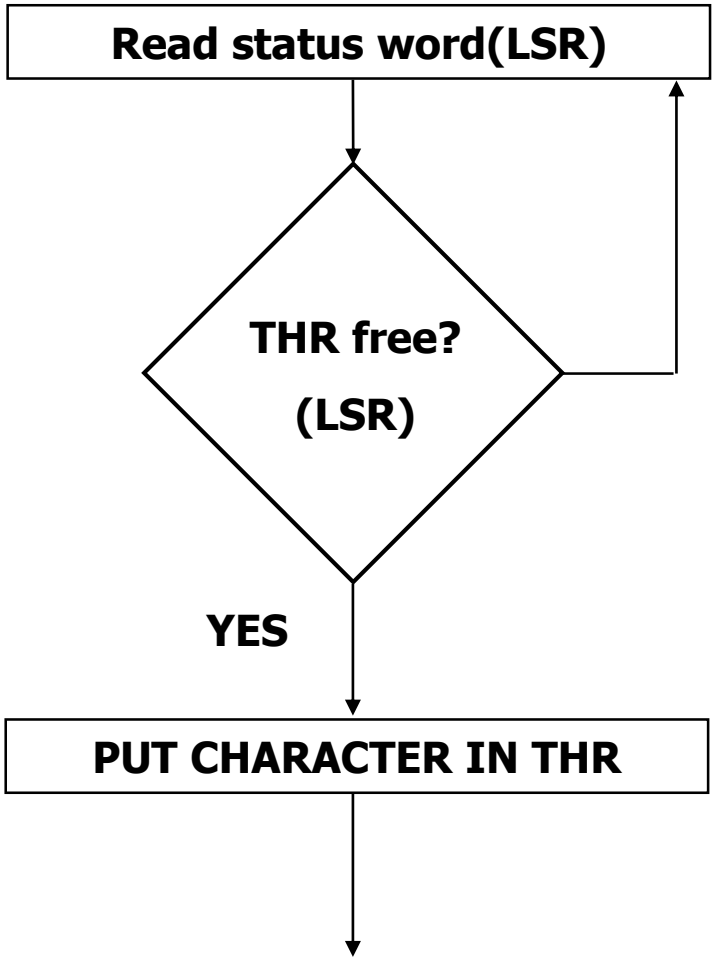
```
31                                      CSEG
33  0000 F3                             DI
34  0001
35  0001 CD BE 06    C      CALL    RITARDO
36  0004 31 FF F7           LD      SP,STAC .....
47  0022 ED B0              LDIR
48  0024 08                 EX      AF,AF'
49  0025 D9                 EXX
50  0026
51  0026 CD CB 03    C      CALL RETI       ;reset hw all peripherals (4)
52  0029 CD CB 03    C      CALL RETI
53  002C CD CB 03    C      CALL RETI
54  002F CD CB 03    C      CALL RETI
55  0032 ED 5E              IM 2            ;vector interrupt mode set
56  0034 3E 30              LD A,TABINT
57  0036 ED 47              LD I,A
58  0038
59                  ; Programming PIO, CTC, SIO
61                  ; ***********************************
62  0038 3E 07              LD      A, 07H
63  003A D3 42              OUT     (PIO1AC),A
64  003C D3 43              OUT     (PIO1BC),A
65  003E D3 63              OUT     (PIO2BC),A      ;disab. Int on PIO1A, PIO2B,
PIO1B
66  0040
67  0040 3E 0F              LD      A,MODO0
68  0042 D3 63              OUT     (PIO2BC),A
```

DIGITAL COMMUNICATION TECHNIQUES

```
 78  0052 3E E6          LD     A,0E6H       ; LSB interrupt routine
 79  0054 D3 62          OUT    (PIO2AC),A
 80  0056 3E CF          LD     A,MODO3      ; PIO 3 mode
 81  0058 D3 62          OUT    (PIO2AC),A
 82  005A 3E 83          LD     A,LIN2A      ;  what input/output lines
 83  005C D3 62          OUT    (PIO2AC),A
 84  005E 3E A7          LD     A,CNINTP     ; interrupt control
 85  0060 32 B1 02    D  LD     (PROG2A),A   ; port 2A re-programmed
 86  0063 C6 10          ADD    A,10H        ; a mask will follow
 87  0065 D3 62          OUT    (PIO2AC),A
 88  0067 3E FE          LD     A,MASK       ; mask for lines activating int
 89  0069 D3 62          OUT    (PIO2AC),A
 90  006B
....
148  00C6
149  00C6 FB          MAIN:                  EI
...

;*****************************************************************
********
517  0393 08          PIOINT                 EX     AF,AF'
518  0394 D9             EXX
519  0395 3A B1 02    D  LD     A,(PROG2A)  ;reprog port on negative edge
520  0398 EE 20          XOR    20H          ; test bit D5 edge sensitive
....
521  039A D3 62          OUT    (PIO2AC),A
522  039C 32 B1 02    D  LD     (PROG2A),A   ; to successively program
the port 2A
523  039F
....
534  03B9 D9             EXX
535  03BA 08             EX     AF,AF'
536  03BB FB             EI
537  03BC ED 4D          RETI
538  03BE
....
1078                     ORG 30E6H
1079  30E6
1080  30E6 93 03      C   DEFW PIOINT
1081  30E8 12 05      C   DEFW EIRETI
1082  30EA 12 05      C   DEFW EIRETI
1083  30EC 12 05      C   DEFW EIRETI
1084  30EE 69 01      C   DEFW CTCINT
1085  30F0 CD 03      C   DEFW TRASB
1086  30F2 D9 03      C   DEFW EXTB
1087  30F4 E3 04      C   DEFW RICEB
1088  30F6 0A 05      C   DEFW SPECB
1089  30F8 BE 03      C   DEFW TRASA
1090  30FA E4 03      C   DEFW EXTA
1091  30FC EF 03      C   DEFW RICEA
1092  30FE EA 04      C   DEFW SPECA
```

Transmission of a character through polling control

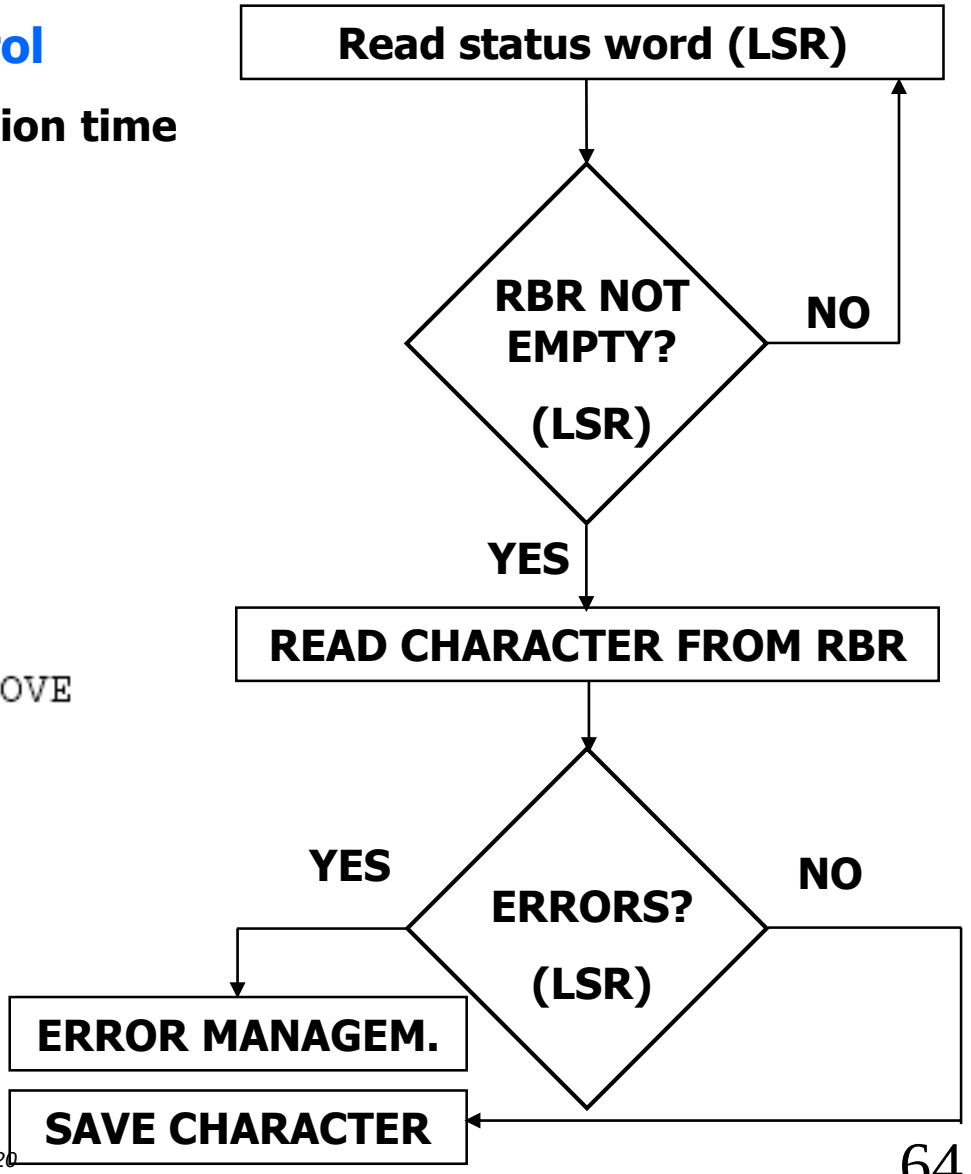
```
        mov     dx, LSR
wait_tr:
        in      al,[dx]
        and     al,MKTHRE
        jz      wait_tr
        mov     dx,THR
        mov     al,[carattere]
        out     [dx],al
```



Read character through polling control

polling with period < character transmission time

```
        mov    dx, LSR
wait_rc:
        in     al,[dx]
        mov    ah,al
        and    al,MKDRDY
        jz     wait_rc
        mov    dx,RBR
        in     al,[dx]
        and    ah, MKBRK+MKFRE+MKPAE+MKOVE
        jnz    err_ricez
        mov    [carattere],al
```



DIGITAL COMMUNICATION TECHNIQUES

Message format

Structured as "container" + "content"

Container = "header"+ "trailer"

Initial characters = Start of message (1/2 bit)

Destination code for multipoint connections. Multicast and broadcast possible

Total n° of characters to establish the end of a binary message

Sender code (in particular for multipoint connections)

Progressive of transmission = order n° of the message (to verify the sequence correctness)

Progressive of reception = order n° of the last correct message received

Function code = the role played in the message by the current received fragment

Data (ASCII/binary)

Verification code = checksum, longitudinal parity or CRC

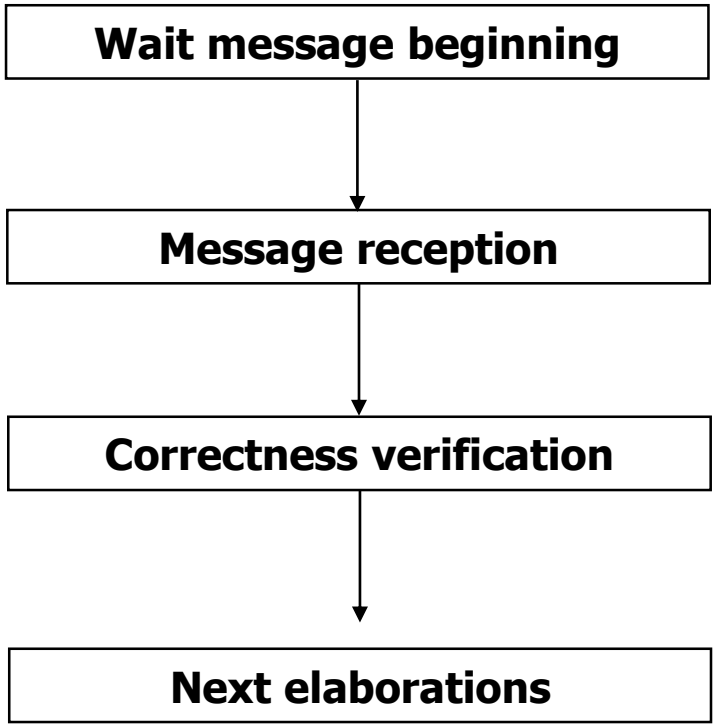
Message reception

"Relaxed" wait of the first character and fast reception of the next ones

Reception through polling/interrupt (often the 1° character with polling, the others with polling or interrupt), verification of the correctness and calculation of the verification code

Comparison of the two verification codes

Message elaboration and answer



DIGITAL COMMUNICATION TECHNIQUES

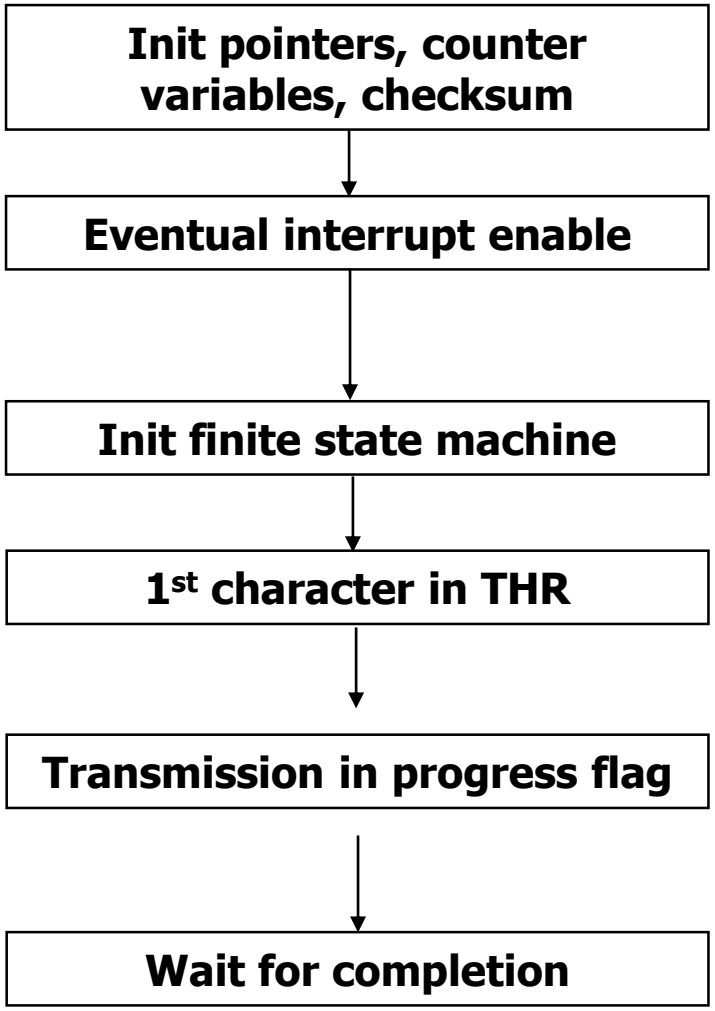
Transmission of messages - Constraints

if polling and if asynchronous transmission are implemented, too long break must be avoided to be no interpreted as time out (end of transm.)

if interrupt, the 1st character must be written in THR so triggering the interrupt request for next characters

The code is organized as a finite state machine

Before disabling the line, wait the end of the last character transmission



An example: half duplex point-2-point protocol

Message format in application processes

`tipo`	byte di codice tipo messaggio
`ncar`	numero byte di testo (0 < `ncar` < 256)
`testo`	sequenza di byte di informazione utile

Available functions in application processes

`void ini_com (void)`	inizializza la comunicazione
`void put_tra (mess)`	inserisce messaggio in coda di trasmissione
`mess get_ric (void)`	estrae messaggio da coda di ricezione. Se il sistema remoto non aveva nulla da trasmettere si riceve un messaggio di tipo = 0 e contenente un solo byte = 0 di testo.

Message format for communication processes

`SOH`	carattere iniziale *Start Of Header*
`prog_t`	codice progressivo del messaggio corrente
`prog_r`	codice progressivo dell'ultimo messaggio ricevuto correttamente
`tipo`	codice tipo messaggio
`ncar`	numero byte del testo
`testo`	contenuto del messaggio
`cks`	checksum di tutti i byte da SOH escluso

DIGITAL COMMUNICATION TECHNIQUES

An example: half duplex point-2-point protocol

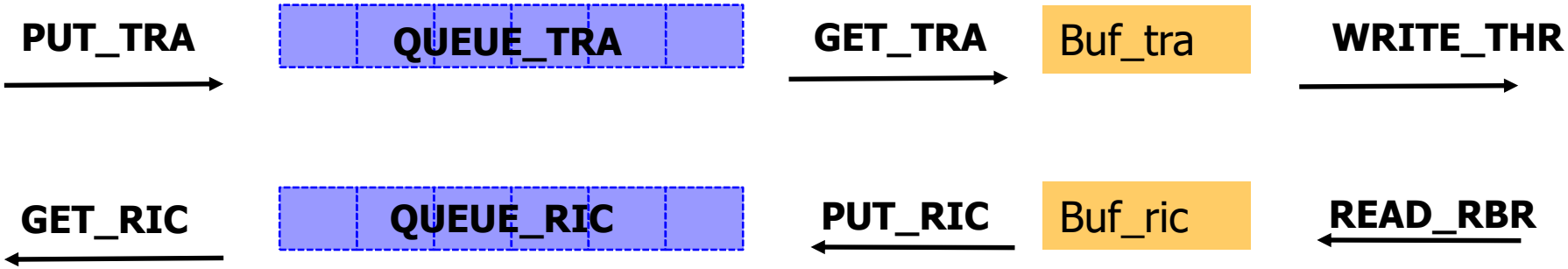
Data structures
(communication level, 8 bit characters + parity)

`coda_ric`	coda di messaggi correttamente ricevuti in attesa di essere consumati dai processi applicativi.
`coda_tra`	coda di messaggi prodotti dai processi applicativi e da trasmettere.
`buf_ric`	array di caratteri per la ricezione del messaggio corrente.
`buf_tra`	array di caratteri per la trasmissione del messaggio corrente.

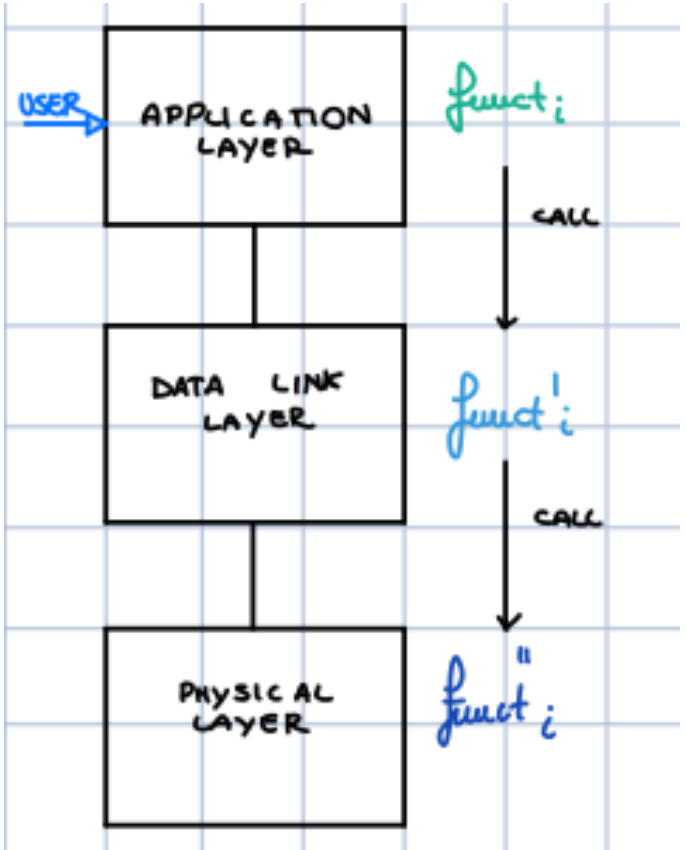
Communication procedures

`mess get_tra (void)`	estrae messaggio dalla coda di trasmissione. Se la coda è vuota si invia un messaggio di tipo vuoto (=0) con un solo byte (=0).
`void put_ric (mess)`	inserisce messaggio nella coda di ricezione
`interrupt int_ric ()`	funzione chiamata da interrupt di ricezione, cioè quando si è ricevuto un carattere o si ha un errore in ricezione. Organizzata come automa a stati che evolve con l'andamento del messaggio in corso di ricezione.
`interrupt int_tra ()`	funzione chiamata da interrupt di trasmissione, cioè quando si ha *Transmit Register Empty*. Organizzata come automa a stati che evolve con l'andamento del messaggio in corso di ricezione.
`int leggi_tempo (void)`	lettura tempo corrente, usata per il rilievo di time-out di fine messaggio

DIGITAL COMMUNICATION TECHNIQUES



An example: half duplex point-2-point protocol



An example: half duplex point-2-point protocol

In the following slides an example of code managing message reception and transmission interrupt is presented-

The interrupt vectorization routines have been settled together with initialization of registers and identification through IIR of the interrupt origin

Communication level variables

```
char
     stato_t     stato dell'automa di trasmissione
     stato_r     stato dell'automa di ricezione
     prog_t_rem      codice progressivo di trasmissione del sistema remoto
     prog_t_loc      codice progressivo di trasmissione del sistema locale
     prog_r_rem      codice progressivo di ricezione del sistema remoto
     prog_r_loc      codice progressivo di ricezione del sistema locale
     cks_t           checksum corrente del messaggio in trasmissione
     cks_r           checksum corrente del messaggio in ricezione
     count           numero caratteri del testo da ricevere
int
     it              indice corrente del buffer di trasmissione
     ir              indice corrente del buffer di ricezione
long int
     tempo_prec          tempo di ricez. carattere precedente (per t-out)
     tempo_corr          tempo di ricez. carattere attuale (per t-out)
```

DIGITAL COMMUNICATION TECHNIQUES

An example: half duplex point-2-point protocol– Reception interrupt

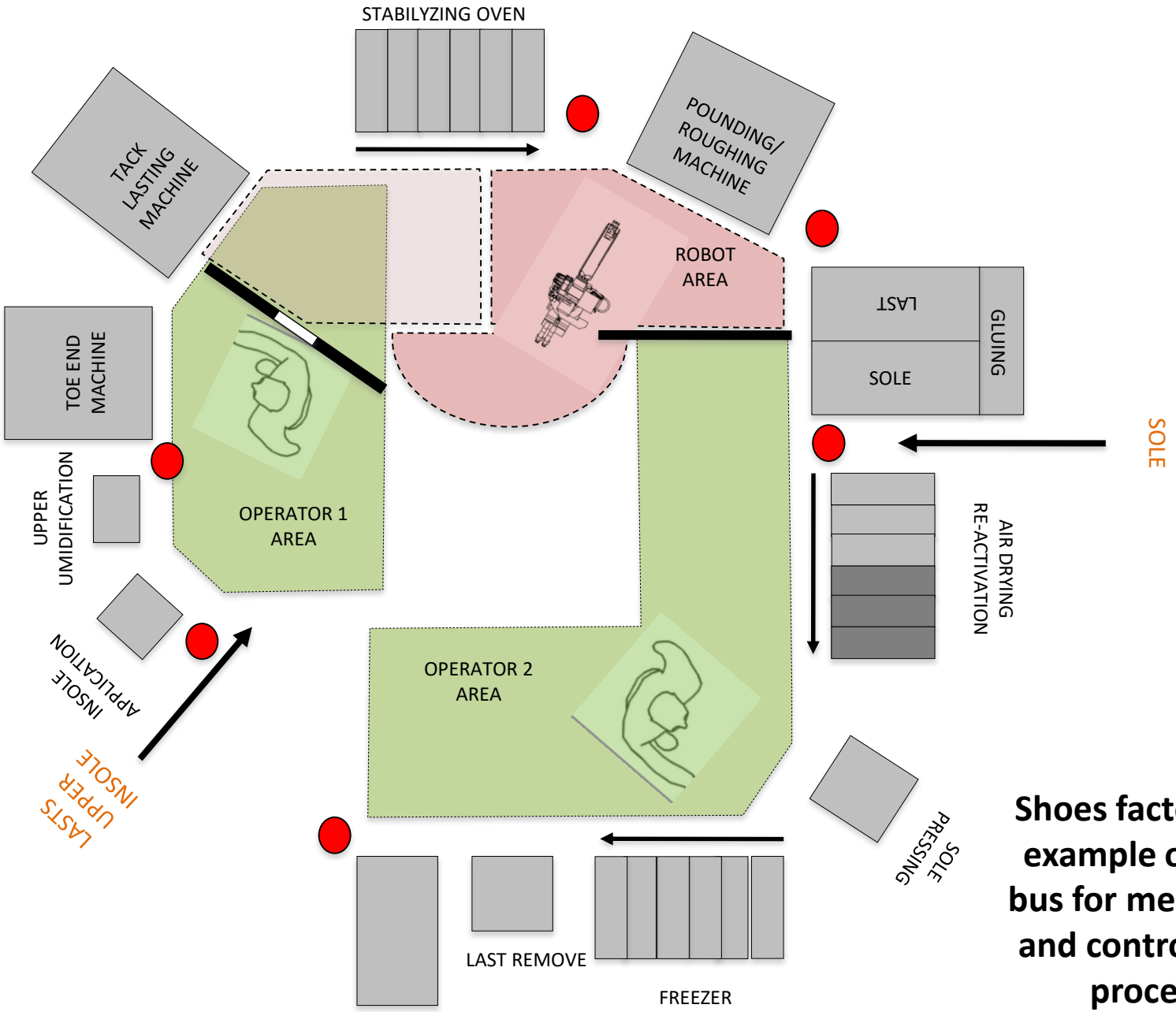
```
void leggi_byte_uart (void)
  {
  byte = leggi_rbr (); // registro ricezione UART
  errori = check_lsr ();      // line status per verif
  if (errori)
      {tempo_prec = leggi_tempo (); // tempo arrivo by
      stato_r = S_ER;}
  switch (stato_r)
  {case S0:
      if (byte == SOH)
            stato_r = S1;
      else
            {tempo_prec = leggi_tempo (); // t arrivo
            stato_r = S_ER;}
      break;
  case S1:  // aspetta progressivo trasmesso remoto
      cks_r = 0;  // azzera checksum
      ir = 0;           // indice buffer a inizio
      prog_t_r = byte;
      cks_r += byte;
      stato_r = S2;
      break;

  case S2:  // aspetta progressivo ricevuto remoto
      prog_r_r = byte;
      cks_r += byte;
      stato_r = S3;
      break;

  case S3:  // aspetta codice tipo
      buf_ric [ir] = byte;
      ir++;
      cks_r += byte;
      stato_r = S4;
      break;

case S4:  // aspetta numero byte testo (>=1)
    buf_ric [ir] = byte;
    ir++;
    count = byte;
    cks_r += byte;
    stato_r = S5;
    break;

case S5:  // aspetta i byte del testo
    buf_ric [ir] = byte;
    ir++;
    cks_r += byte;
    if (--count == 0)
          stato_r = S6;
    break;

case S6:  // aspetta byte di checksum
    if (byte == cks_r)      // verifica checksum
          {put_ric (buf_ric);     // nuovo messaggio corretto
          prog_r_l = prog_t_r;
          disab_ric ();
          enab_tra ();            // abilita trasmissione
          stato_r = S0;}
    else                    // errore checksum
          {tempo_prec = leggi_tempo (); // t arrivo byte
          stato_r = S_ER;}

    break;

/* ERRORE - Lascia passare, ignorandoli, tutti gli eventuali
caratteri rimanenti e aspetta l'inizio del messaggio successivo
riconosciuto dal carattere SOH ricevuto dopo lo scadere di time-o
*/
case S_ER:      // ERRORE -
    tempo_corr = leggi_tempo ();  // tempo arrivo byte
    if (((tempo_corr-tempo_prec)>=TOUT) && (byte==SOH))
          stato_r = S1;
    else
          tempo_prec = tempo_corr;
    break;
}
```

An example: half duplex point-2-point protocol – Transmission interrupt

```
void scrivi_byte_uart (void)
  {
  switch (stato_t)
  case S0:
      {/* vede se l'ultimo messaggio trasmesso e' stato
       ricevuto correttamente da sistema remoto */
      if (prog_r_r == prog_t_l)
            prog_t_l++;
            buf_tra = get_tra ();   // nuovo messaggio
            if (buf_tra == NULL)    // nulla da trasmet
                  {buf_tra[0] = 0;  // tipo
                  buf_tra[1] = 1;   // num. byte
                  buf_tra[2] = 0; } // byte testo
      byte = SOH;
      scrivi_thr (byte);
      stato_t = S1;
      break;

  case S1:  // trasmette progressivo trasmesso locale
      cks_t = 0;  // azzera checksum
      it = 0;           // indice buffer a inizio
      byte = prog_t_l;
     scrivi_thr (byte);
     cks_t += byte;
     stato_t = S2;
     break;

 case S2:  // trasmette progressivo ricevuto locale
     byte = prog_r_l;
     scrivi_thr (byte);
     cks_t += byte;
     stato_t = S3;
     break;
```

```
case S3:  // scrive codice tipo
    byte = buf_tra [it];
    scrivi_thr (byte);
    it++;
    cks_t += byte;
    stato_t = S4;
    break;

case S4:  // trasmette numero byte testo (>=1)
    byte = buf_tra [it];
    scrivi_thr (byte);
    it++;
    count = byte;
    cks_t += byte;
    stato_t = S5;
    break;

case S5:  // trasmette i byte del testo
    byte = buf_tra [it];
    scrivi_thr (byte);
    it++;
    cks_t += byte;
    if (--count == 0)
          stato_t = S6;
    break;

case S6:  // trasmette byte di checksum
    byte = cks_t;
    scrivi_thr (byte);
    stato_t = S7;
    break;

case S7:  // termine trasmissione
    disab_tra ();
    enab_ric ();
    stato_t = S0;
    break;
}
```

FIELD BUS – A typical application



Shoes factory: an example of field bus for measuring and controlling a process

PC CONTROL and its connections

Serial line for communication with the machines

- Tack lasting machine
- Waist lasting machine
- Oven for stabilization
- Pounding/Roughing machine
- Gluing machine
- Heel fixing machine
- COMAU ROBOT

- COMPONENTS IDENTIFICATION
- MOUNTING STATION (island)
- POUNDING/ROUGHING INPUT
- SOLE IDENTIFICATION
- HEEL FIXING

Connection through antennas to read the semi-manufactured product

DIGITAL COMMUNICATION TECHNIQUES

Field Bus

Standardization of communications between devices used in data acquisition and industrial process control.

Two possibilities: industrial networks and field buses

•Networks (Ethernet) diffused and popular. Faster, wider, more expensive than field buses and general purpose

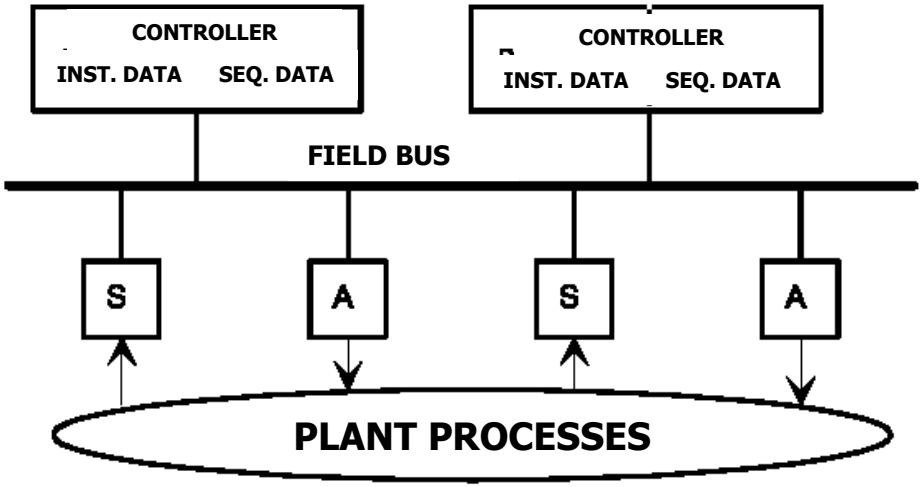
•Field buses (Can) good compromise among performance, real time, robustness, costs

A field bus is a *distributed control system* made up by one or more controllers, sensors and actuators

Data are "logically divided" in controllers:

•"process data base": status process variables, periodically refreshed through instantaneous sensor measurements

•"historical data base" with significant events organized in queues (sequential)



DIGITAL COMMUNICATION TECHNIQUES

Required services

periodic communication of sensor acquisitions and command towards the actuator

aperiodic communication with high priority of events (alarms, anomalies, …)

aperiodic communication of low priority infos (configurations, on line diagnostics)

Constraints (specs)

Instantaneous data refresh frequency = sensor sampling frequency

Time validity of values = consider that the physical distance between producer and consumer implies a delay that must not compromise the system stability

Temporal coherence = data received from many different producers must refer to the same "time period"

Spatial coherence = if more controllers use the same variable they must use identical values

Interoperability = working although different vendors

Sporadic messages must be affected by the minimum possible delay (real time?)

Simple interfaces, cheap, reliable and safe

DIGITAL COMMUNICATION TECHNIQUES

Transmission media

copper cables: **for short-medium distance connections (from m to Km) but the bandwidth diminishes with the distance due to dissipation in the medium. To achieve velocities greater than Mb/sec at great distances repeaters or bridges are necessary**

twisted pairs: **cheap, with good noise immunity and transmission velocities (Mb/sec). They can be shielded (but sometimes no) and they are also used to power supply the devices connected through the bus (modulation)**

coaxial cables: **definitively better (in terms of bit rate, electromagnetic compatibility and reachable distances) than twisted pairs; used where technical difficulties do not allow the use of optical fibers (i. e. vibrations)**

cables for energy transport: **used in automatic devices installation in buildings (domotic) or to bring digital communication exploiting wide distribution supports (water, electricity, gas); transmission is made through signal modulation superimposed on the AC network voltage thus using the same transmission medium. Sensible to electromagnetic disturbs.**

optical fibers: **at now the most performing technology in terms of velocity, distance and electromagnetic compatibility however with important costs. That can be partially reduced by the relatively new plastic optical fiber technology**

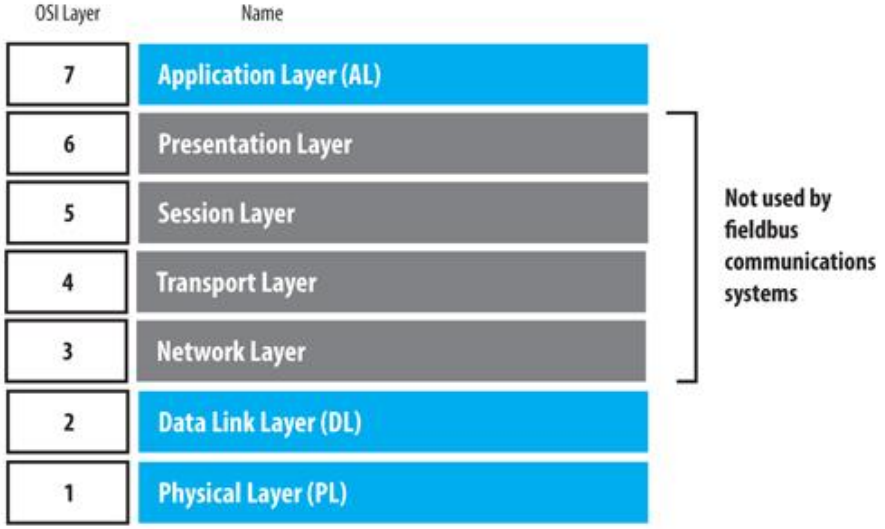
Wi fi: **ever and ever increasing diffusion but still very sensible to disturbs**

DIGITAL COMMUNICATION TECHNIQUES

Architecture of a typical field bus

For high performance communication systems (low latency, determinism, high throughput) or in case of limited available resources a slimmer architecture version can be implemented of the ISO/OSI model featuring the sole Application, Data Link and Physical (Enhanced Performance Architecture) levels

The 3, 4, 5, 6 levels are not explicitly implemented except some minimal functions that are included in Application or Data Link or implemented in a new intermediate level



Efficiency

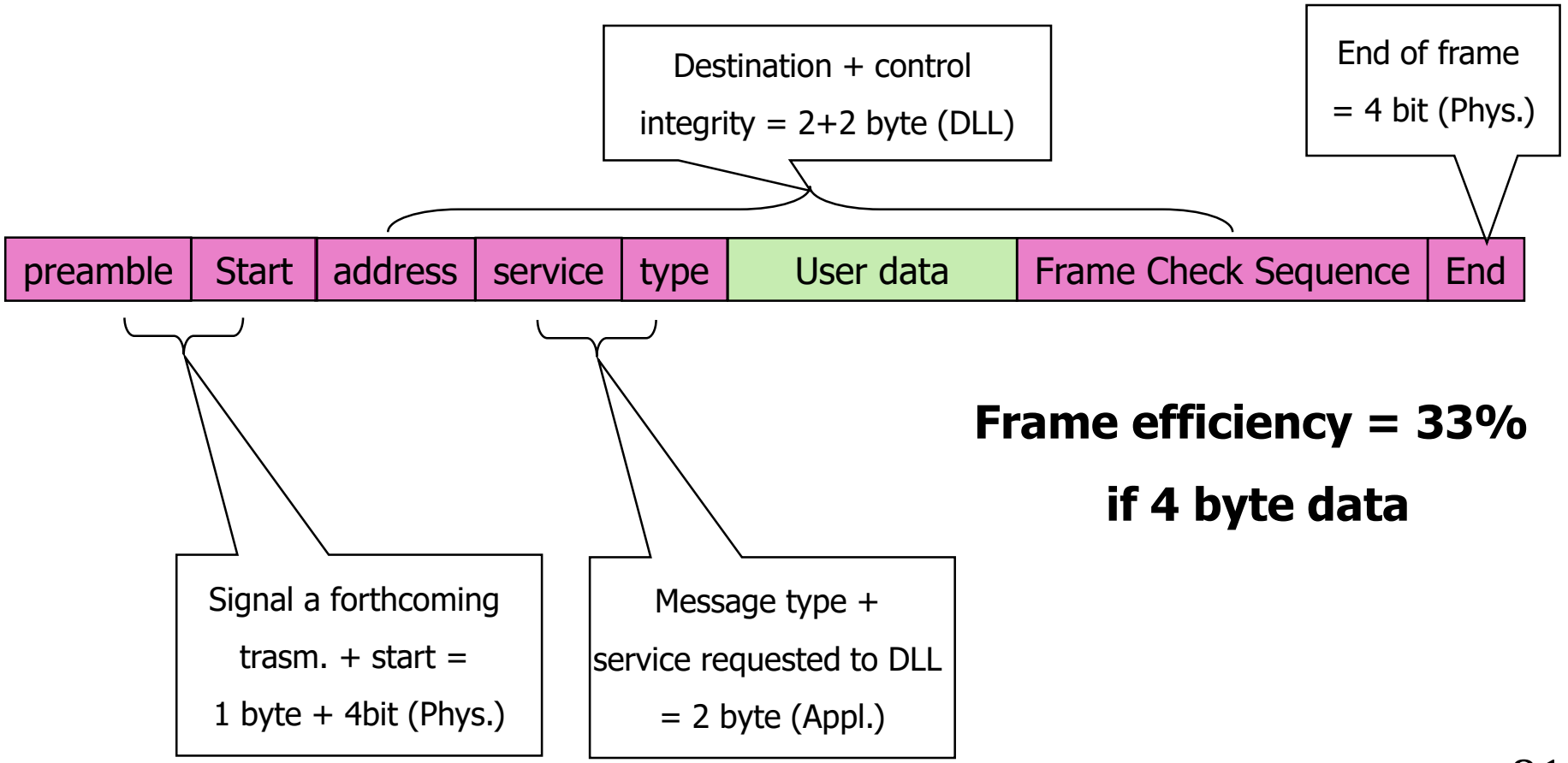
parameter that combines transmission velocity with the n° of data transferred in the unit of time

$$\tau \text{ (throughput)} = \varepsilon * \nu \text{ (velocity)}$$

while the velocity is generally known, the throughput is not since due to specific working conditions and depending on the frame overhead, the bus access mode, address technique and communication model, mode of use of the bus

Frame overhead

The transmitted data frames are "packed" within predefined structure (Protocol Data Unit) that are composed in the different layers of the ISO/OSI model, for instance:



Frame efficiency = 33%

if 4 byte data

Medium Access Control

Many devices access to the same transmission medium: this can create conflicts so compromising the data exchange correctness

MAC policy is fundamental to assure that data traffic respects the required time constraints i. e. *deterministic*

MAC is implemented in the Data Link Layer and differentiates the field buses on the basis of three access control modes:

- **centralized: only 1 node assigns the control to the different connected stations**
  - **master/slave or bus-arbiter**
- **distributed: many or all the stations can assume the transmission mode control**
  - **Token passing or TDMA**
- **multiple: policies in which all the nodes can transmit at the same time; the difference is made on the basis of conflicts management**
  - **Carrier Sense Multiple Access with collision detection or avoidance or resolution or arbitration**

DIGITAL COMMUNICATION TECHNIQUES

Master/slave

A unique station called master is authorized to spontaneously transmit

The other nodes can answer on the basis of a cyclic interrogation by the master following a schedule predefined (polling list) or modifiable (dynamic) traffic-sensitive

Every slave, when asked, provide data or asks for a service

Advantage: deterministic traffic control ⇒ upgrade times assured

Bus arbiter (Link Active Scheduler)

A unique station is the communication arbiter and controls a network of nodes configured as information (variables) producers or consumers

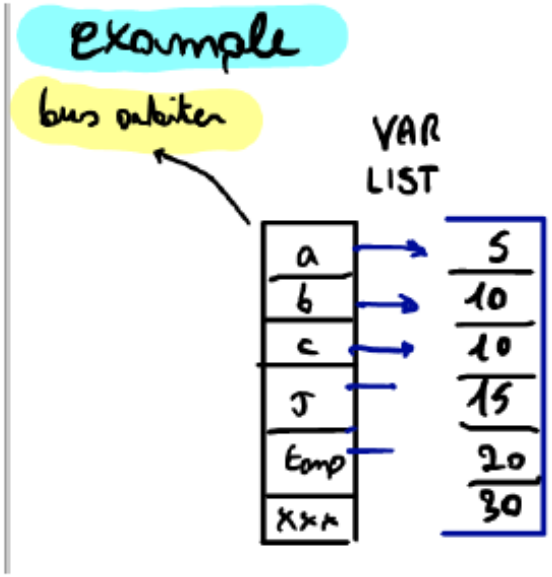
Every variable is identified by its proper name and is "called" by the arbiter on the bus following a scheduling table dynamically managed

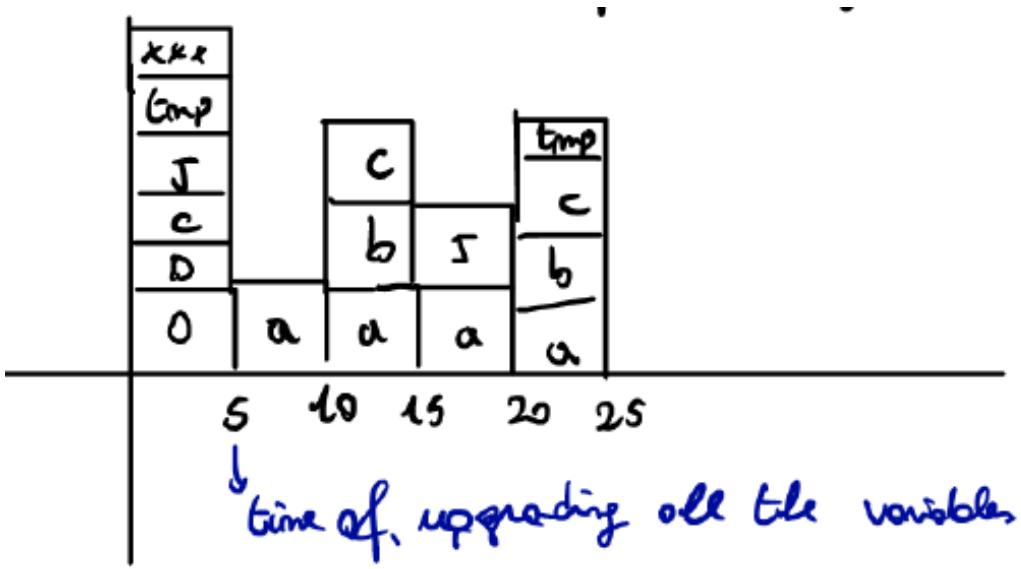
The producer node will put its own variable on the bus at every call

The consumer node will read that variable at every transmission

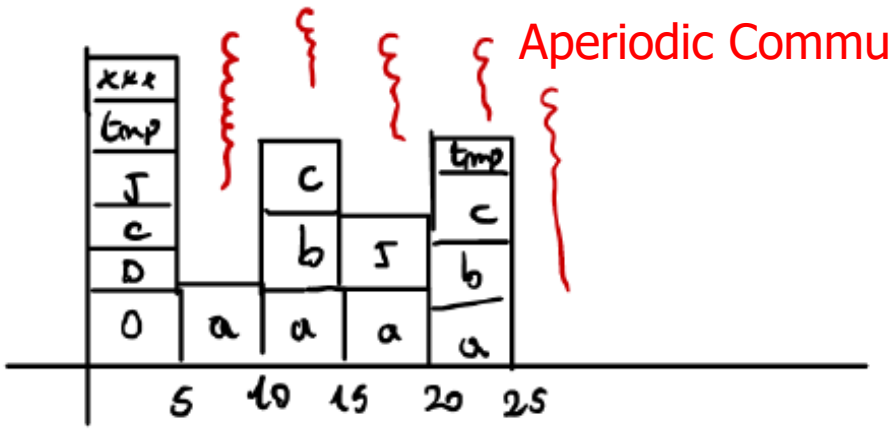
The arbiter can temporarily distribute the control to other nodes to support different communication modes (client server)

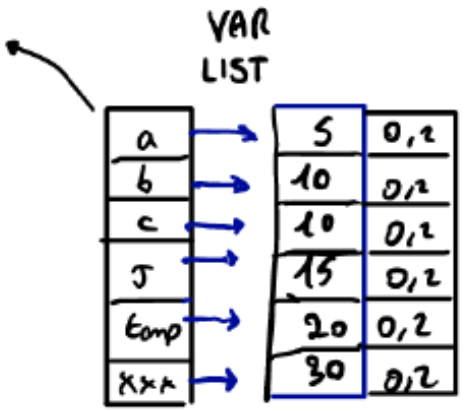
Advantage: deterministic traffic control ⇒ upgrade times assured





Aperiodic Communications





DIGITAL COMMUNICATION TECHNIQUES

Token passing

The access to the bus is reserved to the station (master) that holds a suitable token

A token is a particular frame format exchanged among stations

The token is held until a suitable maximum time or until the communication ends

We have a "ring" among token exchanging nodes (with a fixed Token Rotation Time)

If the token acquisition time is constant within the TRT a "Time Locked Token Rotation" is performed otherwise "Free Running Token Rotation"

Advantage: deterministic traffic control ⇒ upgrade times assured

TDMA (Time Domain Multiplexed Access)

A *cycle time* is defined within which all the nodes must have the possibility for communicating

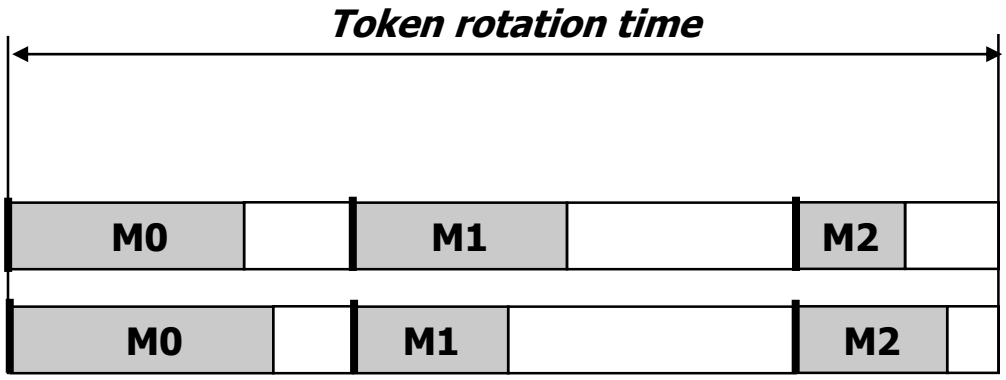
Within the cycle every node is assigned with a "time window" to be the bus master

Similar mechanism as Token passing Time Locked (without token)

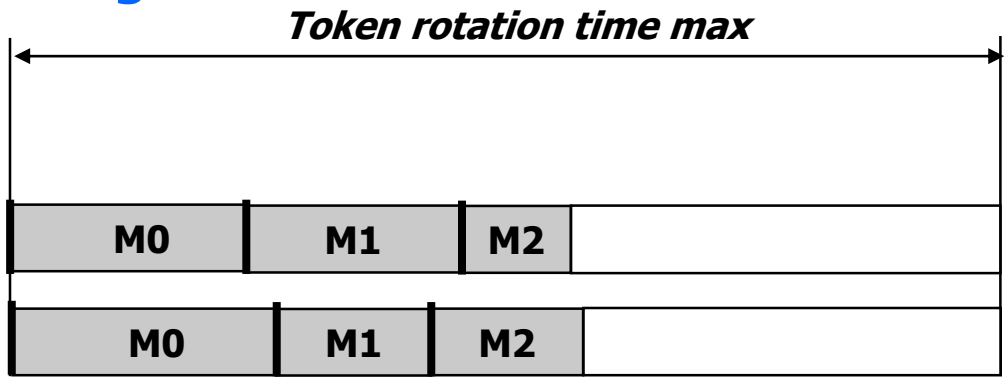
Every station "knows" its proper time since it has a clock synchronous with the bus one

DIGITAL COMMUNICATION TECHNIQUES

Time Locked



Free Running



Bus idle

Trasmission

Token passage

DIGITAL COMMUNICATION TECHNIQUES

Carrier Sense Multiple Access (CSMA)

All the network nodes can send data. Every node is listening by default

If the transmission medium is not occupied, the node sends the data: possibility of collisions. Collisions can be managed adopting different approaches:

1-Collision Detection

In presence of a collision the transmission is suspended

Collided messages are deleted from the medium and they will be retransmitted

The transmission resumes after a random time. The probability of new collisions is related to the network load and to the number of nodes connected to the network

2-Collision Avoidance

Similar to Collision Detection

The waiting time before retransmitting data is randomly generated by each node (it is not a unique values for all the nodes)

The probability of new collisions is related to network load and to the nodes connected to the network. The probability of collision is lower than in Collision Detection since each node waits for a different time before resuming the transmission

DIGITAL COMMUNICATION TECHNIQUES

3-Collision Resolution

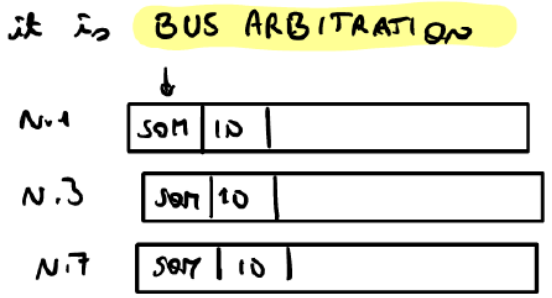
A priority is assigned to each node during network configuration

Collided messages are deleted from the network for being retransmitted

Transmission resumes after an inactivity time related to the priority assigned to the node or to the message

It is similar to Collision Avoidance, bur the waiting time is not randomly generated since it is computed based on priority (this guarantees that the collision cannot be repeated = resolution)

4-Bus Arbitration



A priority is assigned to each node during network configuration

Transmission resumes when the medium is not occupied

If there is a collision, only the transmission with the bigger priority continues. The other nodes will resume their transmission when higher priority transmissions end

Priority frame and check of what a node is transmitting and what is receiving from the bus. A frame indicates that a higher priority node is transmitting. Ex. Can bus

This technique is not destructive since, when a collision happens, transmitted messages are not deleted but their transmission is blocked

DIGITAL COMMUNICATION TECHNIQUES

Communication services

Communication services of a field bus can be:

The sender doesn't know:	The receiver doesn't know:
• If the receiver is present • If the letter arrived • If the receiver can read the letter	• When it is coming

- ***connectionless*** **based on complete and self-describing messages that can be understood by the receiver without additional information. Messages contain format, length, encoding, data and how to use them**

  Useful approach when communication requires occasional accesses to remote stations (commissioning data, identification, file and programs download or upload). The network configurator/manager is not responsible of establishing the connection

  Approach not useful in case of repetitive communications (i. e. cyclical updating of information). The overhead related to descriptive information becomes useless and heavy

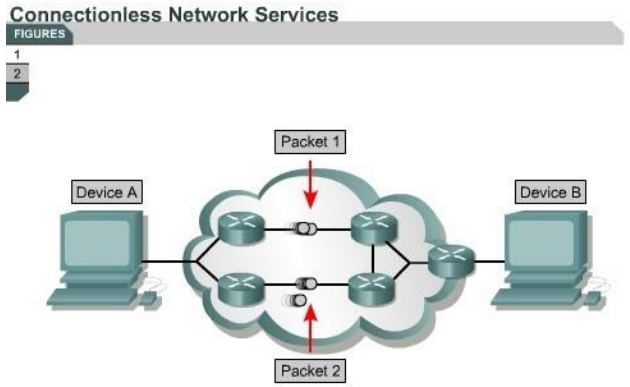
DIGITAL COMMUNICATION TECHNIQUES

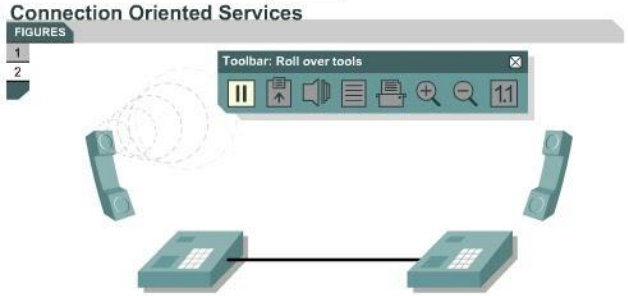
Communication services

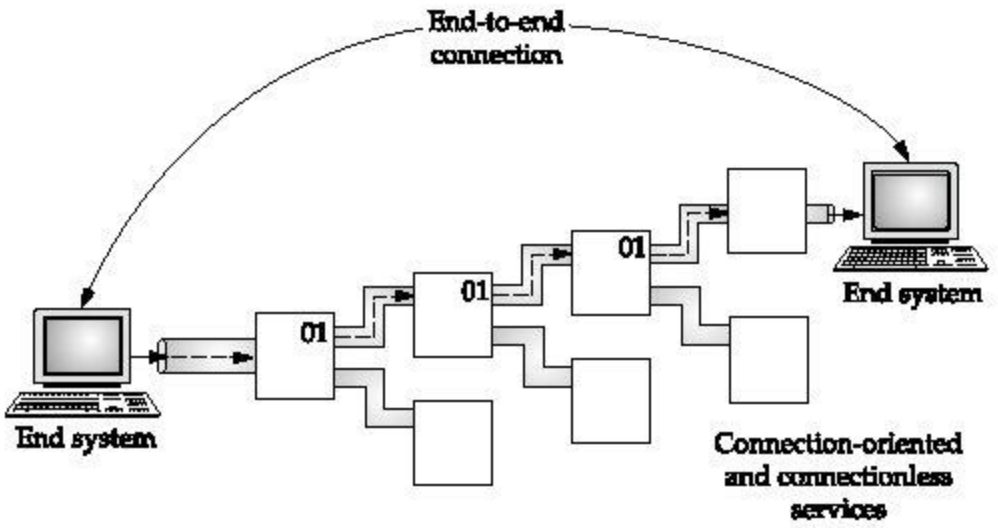
Communication services of a field bus can be:

connection oriented **the coupling between sender and receiver is established by a connection ID. Exchanged messages contain only data and ID with a higher coding efficiency with respect to connectionless services. There is an implicit coupling between info source and destination**

Useful approach for applications based on cyclical repetitions of great numbers of similar messages and for Data Link Layer services (at Application level it is possible to make implicit only information related to high level protocols)







DIGITAL COMMUNICATION TECHNIQUES

Addressing Techniques

- ***point-to-point addressing***

Technique used by Client-Server communication services.

Source Address (SA) and Destination Address on Protocol Data Unit of the Data Link level

- ***multicasting***

Messages are identified by a SA and a Group Address, known by nodes belonging to a group (i.e. nodes connected to the same functional unit).

Advantages are the lower transmission loads. There is the need of physical and temporal consistency of transmitted information

- ***broadcasting***

Messages are concurrently transmitted to all the network nodes using a global ID

Essential for supporting network maintenance and management services. Each field bus has a set of global IDs

Communication models

- ***producer consumer***

It combines the concepts of global addressing and connection oriented service

Each station configured as producer couples data with its connection-ID (unambiguous in the network). Receiving stations will verify at Data Link Level if information must be forwarded to higher levels, checking if they are consumers coupled with the ID or not

It combines sending efficiency of cyclical information, without transmission repetition, with the possibility of guaranteeing physical and temporal consistency

Unsuitable in case of acknowledged services

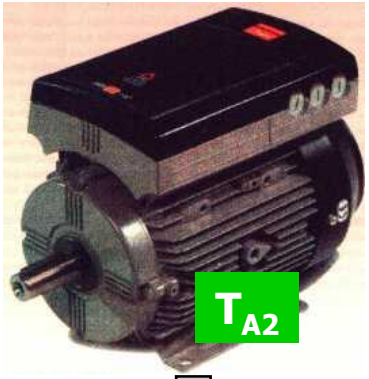
- ***client server***

Point-to-point addressing of connectionless and/or with acknowledged services: a node becomes a client when it explicitly requests a services to a remote station (server)(command execution, event based transmission, I/O…)

DIGITAL COMMUNICATION TECHNIQUES

Response time and latency of a field bus: an example

Push button



T_{L1}

T_{T1}

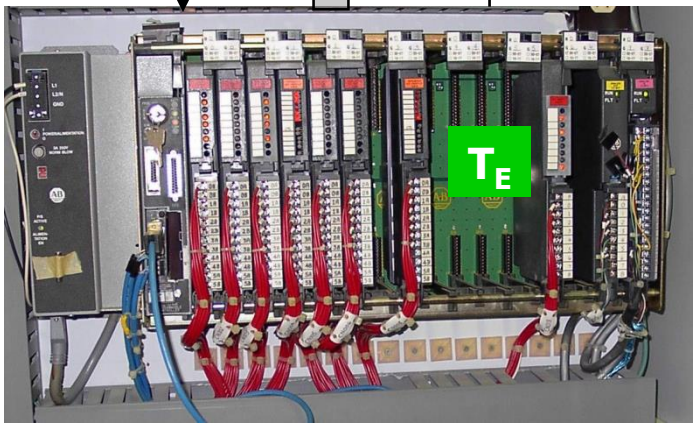
T_{T2}

Electrical drive + actuator

BUS

T_{A1}

T_{L2}



PLC control board

Response time and latency of a field bus: an example

t=0, the operator presses the button

t=T_{L1} The "pressed button" information is on the bus. Latency due to the bus DLL and related to the implemented MAC (centralized/distributed) and to the transmission technique (polling/interrupt). If the protocol is deterministic T_{L1} can be estimated (2 ms)

t=T_{L1}+T_{T1} The information arrives to the PLC after a time that depends on the bit transmission speed and the bit number added to the propagation time of the medium (propagation speed and distance). i.e. 0.06 msec

t=T_{L1}+T_{T1}+ T_{A1} The information is in the PLC registers, ready for being elaborated by the CPU after a time related to internal interface register communication transmission speed

t=T_{L1}+T_{T1}+ T_{A1}+T_E+T_{L2} The information is sent to the drive after a time T_E equal to a PLC elaboration cycle (IN, EXEC, OUT) with a latency T_{L2} as discussed before. T_E is in the range 0,2-20 ms/Kinstructions

t=T_{L1}+T_{T1}+ T_{A1}+T_E+T_{L2}+T_{T2} +T_{A2} the command is implemented after a time T_{A2} needed by the drive software for acquiring information from the interface and to activate the drive (0.01 ms) and after a time T_{T2} due to transmission on the medium same as before)

Response time and latency of a field bus: an example

$$T_{Rmax}=T_{L1max}+T_{T1}+T_{A1max}+T_E+T_{L2max}+T_{T2}+T_{A2max}$$

T_{A1max} at maximum equal to a PLC elaboration cycle so

$$T_{Rmax}=(T_{L1max}+T_{L2max})+(T_{T1}+T_{T2})+2T_E+T_{A2max}=$$

$$4 + 0.12 + (0.4\text{-}40) + 0.01 \approx 4.5\text{-}44.1 \text{ ms}$$

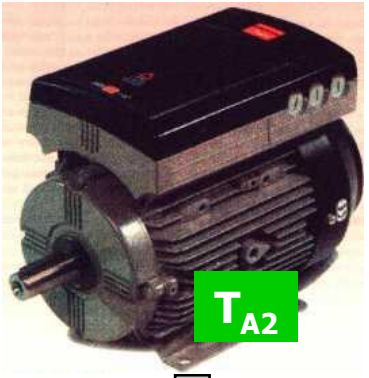
The field bus performance is only one of the elements that determine the response time of the entire system.

This is further highlighted if we add to this scheme an emergency button to stop the system as fast as possible eventually overtaking the elaboration of the PLC.

DIGITAL COMMUNICATION TECHNIQUES

Response time and latency of a field bus: an example

Push button



T_{L1}

T_{T1}

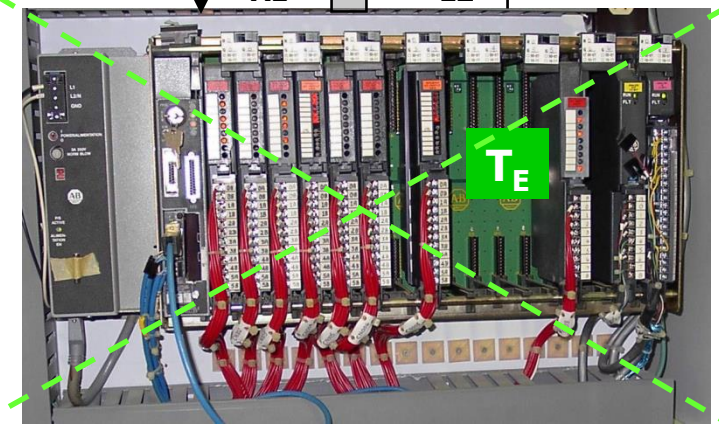
T_{T2}

Electric drive + actuators

BUS

T_{A1}

T_{L2}



PLC control board

Response time and latency of a field bus: an example

t=0, the operator presses the button

t=T_{L1} The "emergency stop" information is on the bus. Latency is due to the bus DLL, to the implemented MAC (centralized/distributed) and to the transmission technique (polling/interrupt). If the protocol is deterministic TL1 can be still estimated (2 ms)

t=$T_{L+}T_T$ the info arrives to the drive after a transmission time T_T; again, this time depends on the modulation time of the frame and on the transmission on the medium. It can be estimated in 0.06+0.06 ms (as before more or less)

t=T_L+T_T+ T_A the command is implemented after a time T_A needed by the drive sw to acquire the information from the interface and to stop the drive (0.01 ms)

In this case the response time will be

$$T_{Rmax}=T_{Lmax}+T_T+ T_A \approx 2 \text{ ms}$$

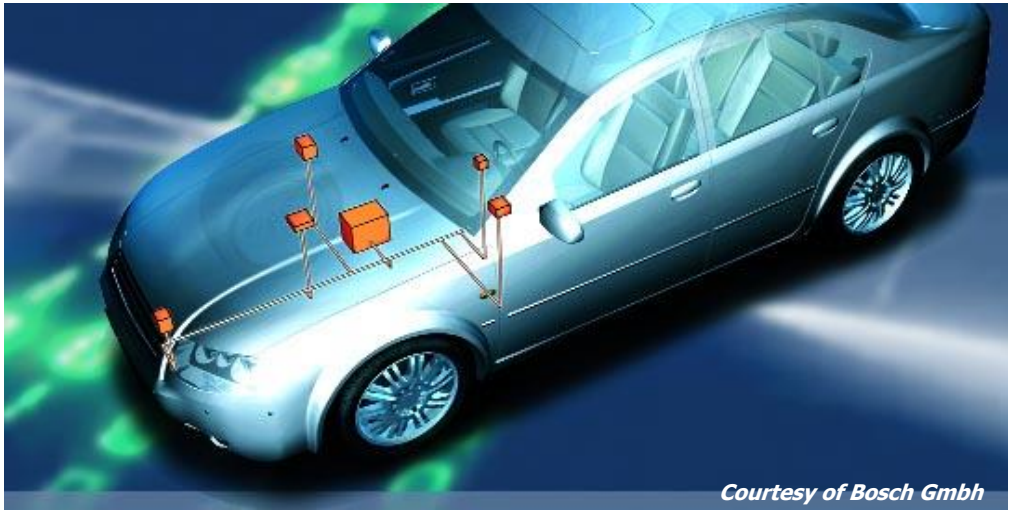
- **In this case the bus performance is responsible of the response times**
- **The transmission latency is a good indicator of the bus performance**
- **The control distribution (especially when simple) helps**

DIGITAL COMMUNICATION TECHNIQUES

CAN (Controller Area Network) Bus

Serial communication protocol proposed by Bosch in the 80s in automotive applications to allow a distributed real time control, using intelligent sensors and actuators mounted on board of a vehicle assuring also a high safety level.

Successively it has been adopted in several industrial fields



www.semiconductors.bosch.de/pdf/can2spec.pdf

Serial transmission with velocity up to 1 Mb/sec (some releases until 10 Mb/s)

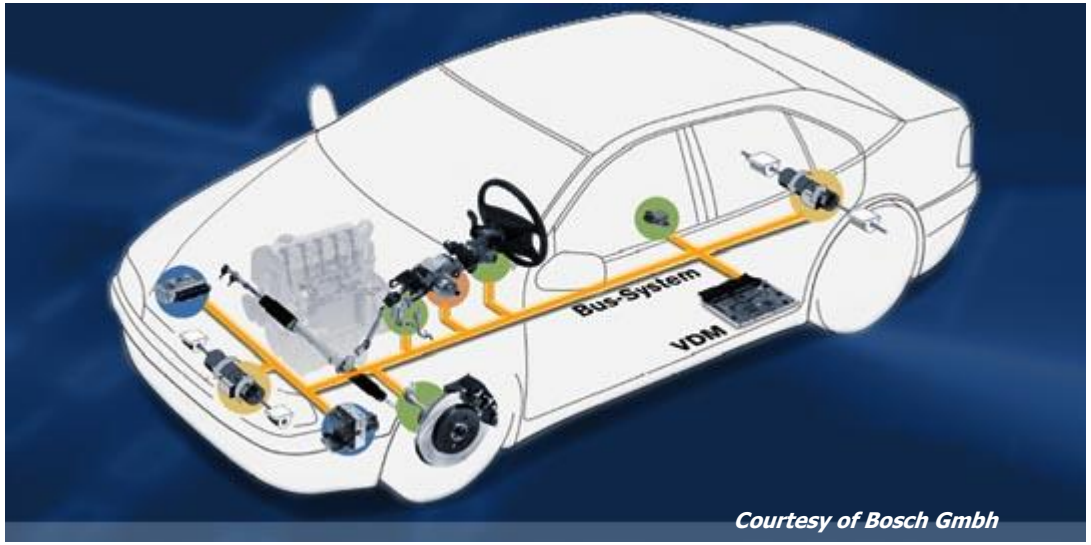
Excellent error detection and management with restriction of possible causes and confinement of the responsible, low costs, noise tolerance (i. e. low probability of a not detected error in normal working conditions)

Real time applications (tools available to have reliable and fixed answer times)

CAN (Controller Area Network) Bus

A simple and flexible wiring : twisted pairs are used (sometimes shielded)

Nodes can be added or cancelled without any re-configuration (no address nodes)



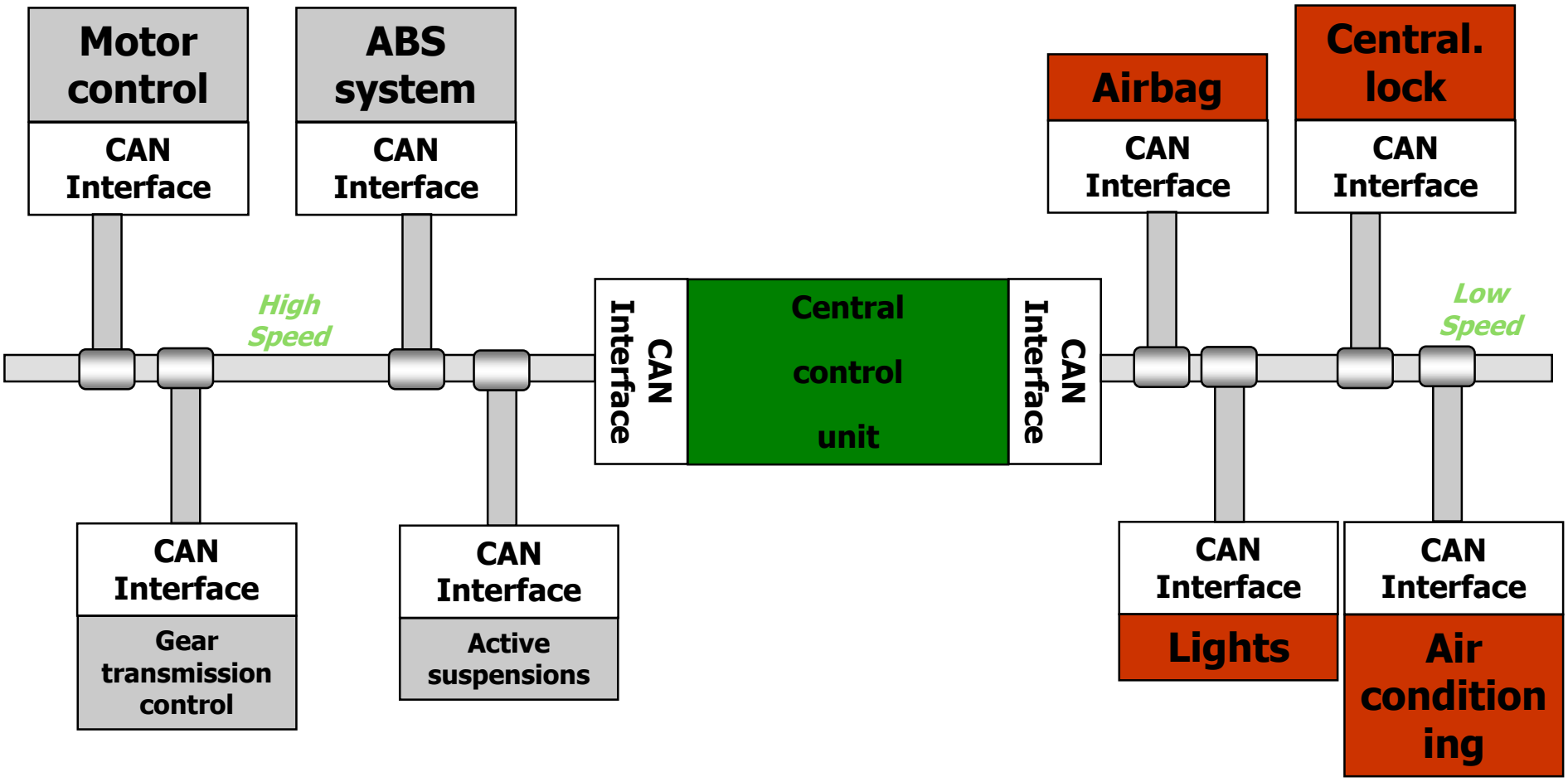
Noise immunity: shielding and differential transmission of the signal

Stabilized standard: wide diffusion and induced availability of hw devices and development tools for different application fields

Recognized as standard (ISO 11898)

DIGITAL COMMUNICATION TECHNIQUES

CAN (Controller Area Network): example



CAN Bus: communication

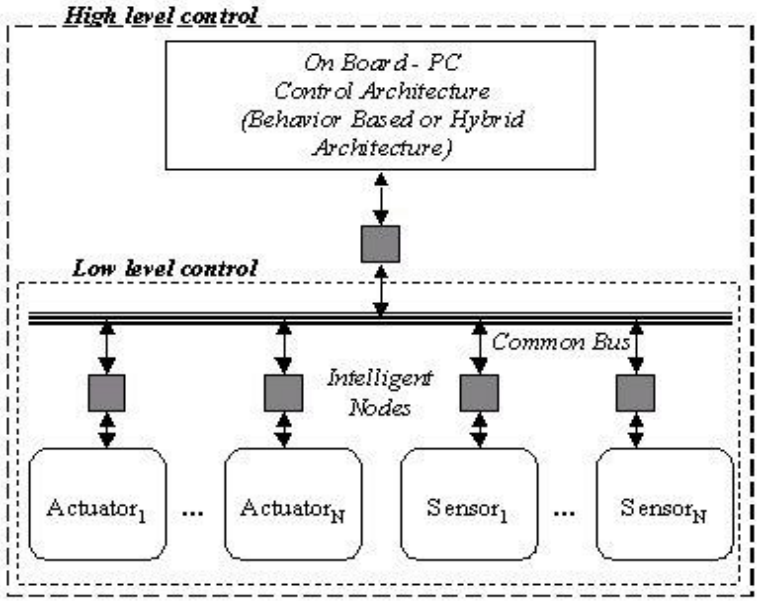
Intelligent sensors and actuators, (equipped with CPU, ADC, RAM e EPROM, timer integrated in the chip) able to pre-amplifying, convert the acquired value and partially pre-elaborate it before transmitting

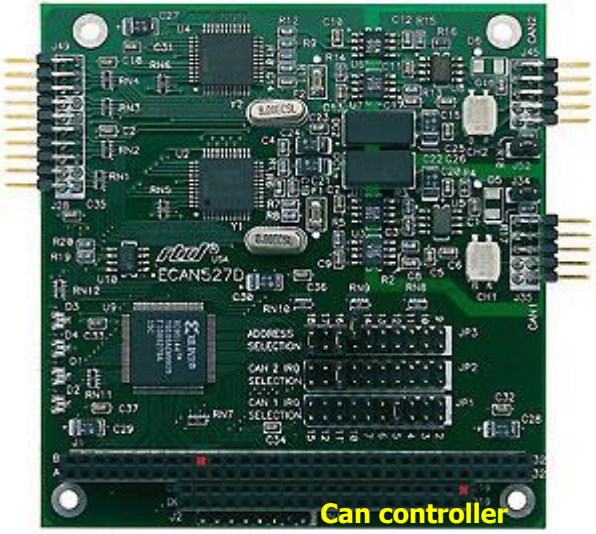
MAC of the kind "Multi-Master" where all the nodes can transmit at the same time (Multiple Access)

Nodes are not addressed but a message is labeled with a suitable *identifier* (maybe of the sender), on the basis of which the receivers elaborate it or not

A suitable priority in the bus access (arbitration) is associated to the identifier

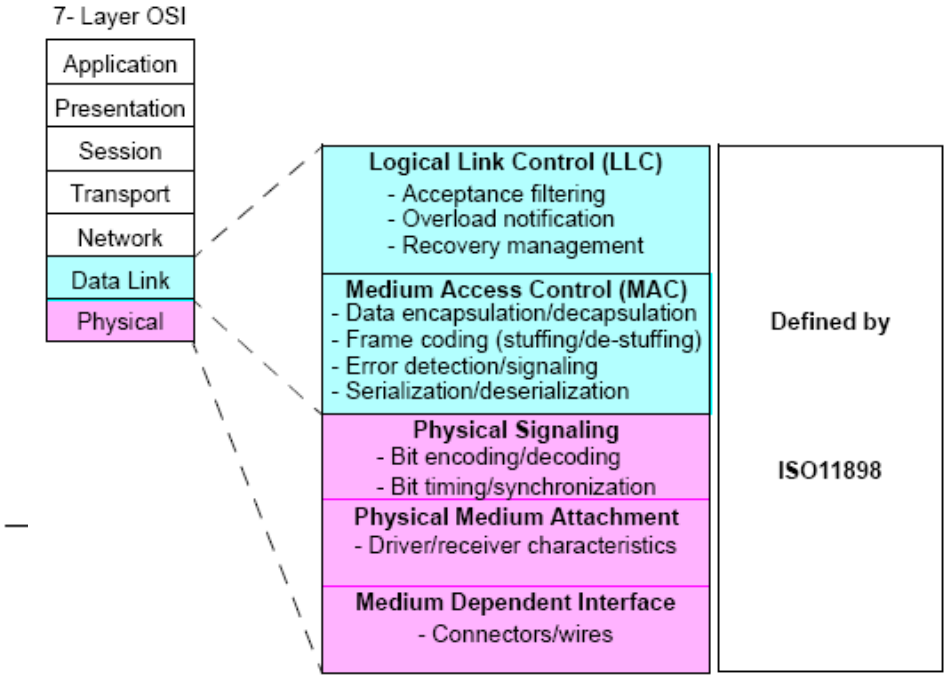
The CAN protocol implements only the Physical and the Data Link layers of the ISO/OSI model while a Application layer is envisioned however it is not implemented in the standard but left to the user

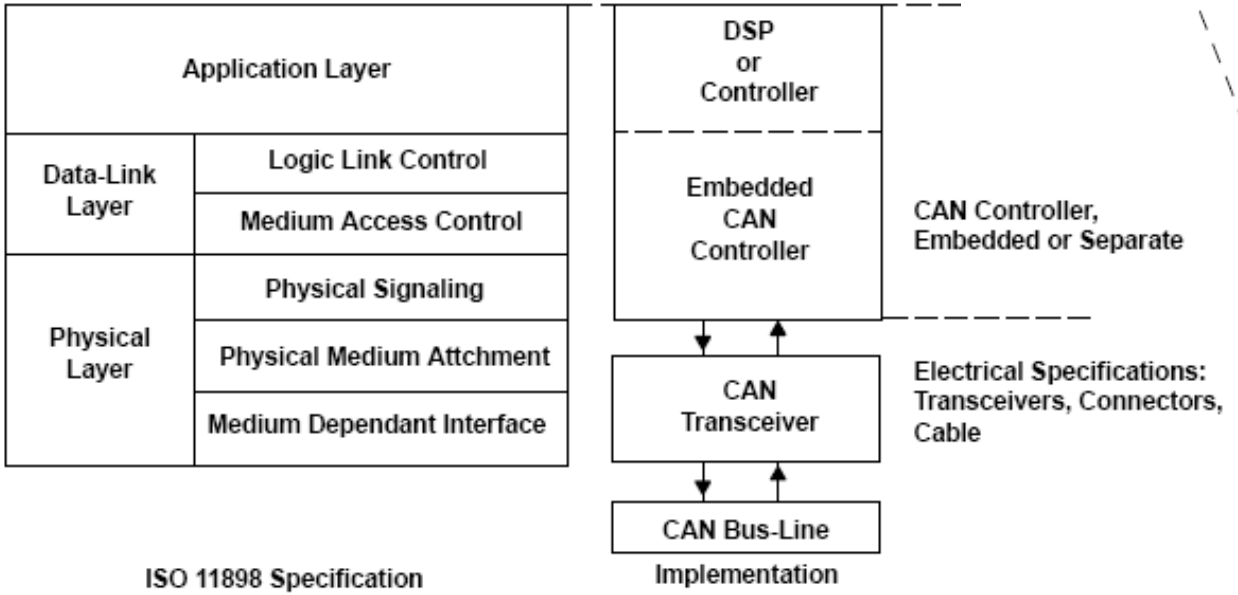




CAN Bus: implementation

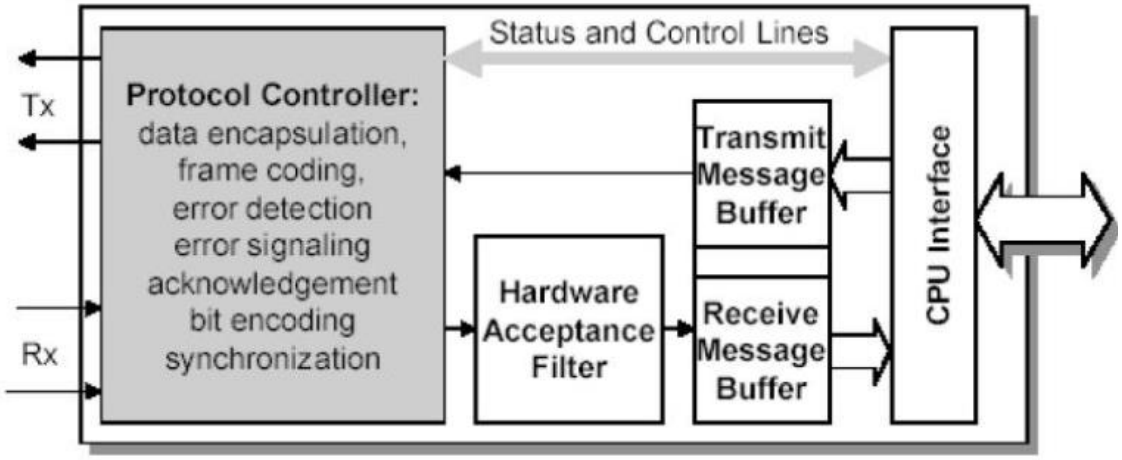
Layer ISO-OSI /functions

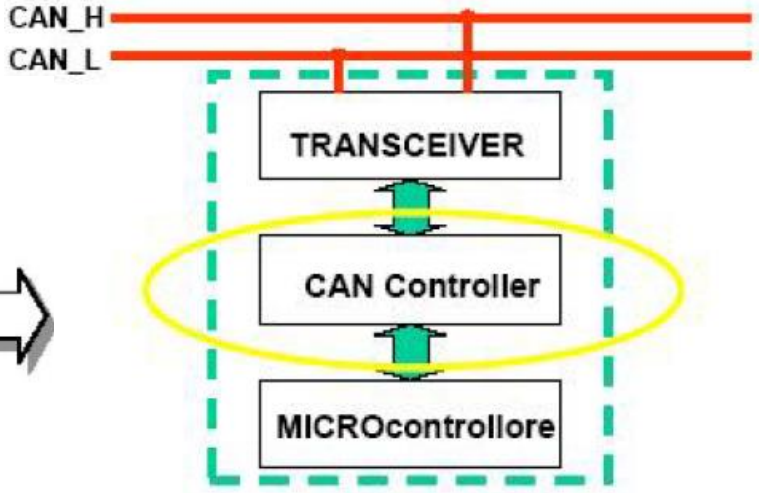




Layer ISO-OSI / devices

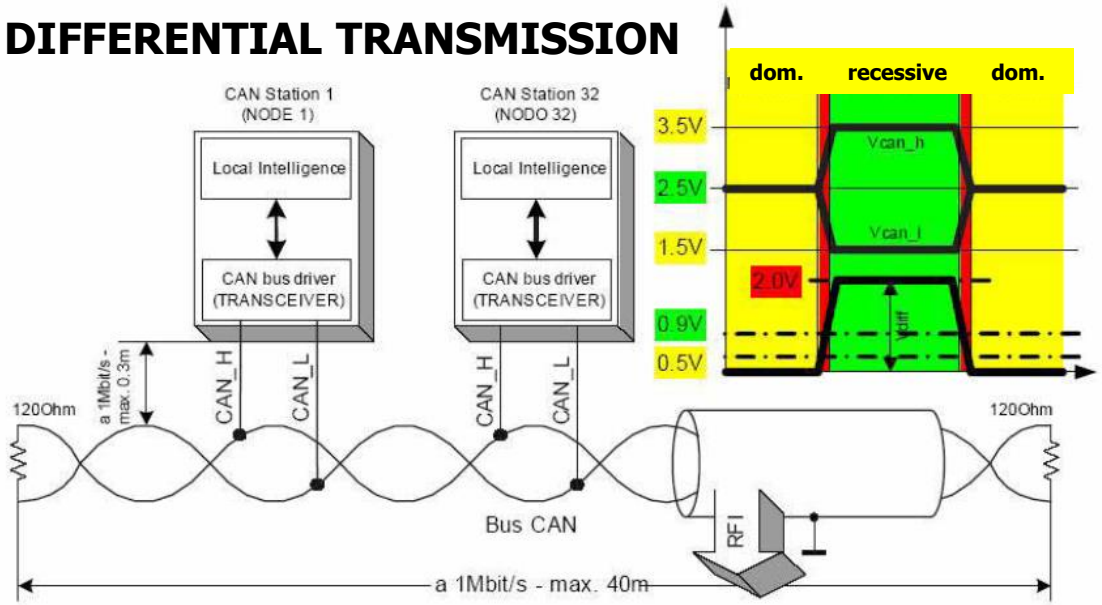
CAN Bus: controller structure



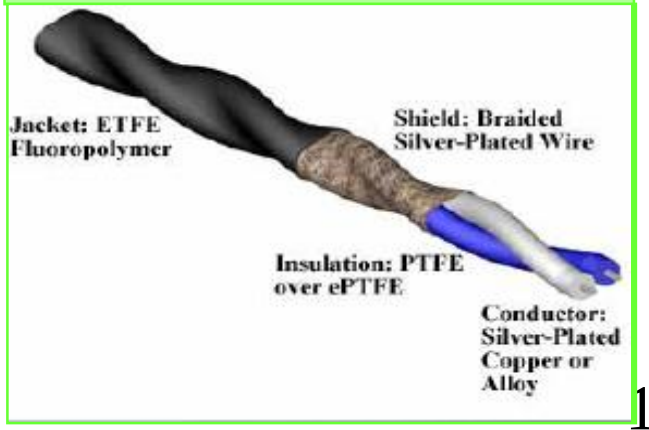


CAN Bus: controller structure

DIFFERENTIAL TRANSMISSION



SJA 1000 Philips (twisted pair shielded)

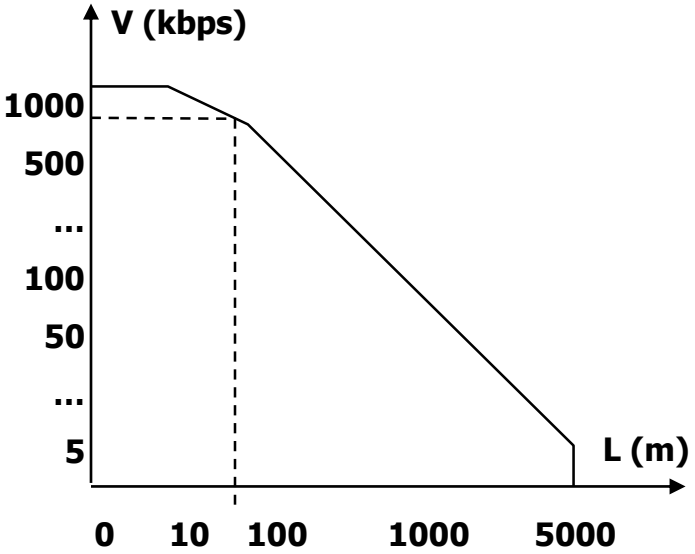


CAN Bus: physical level

Defines the electrical features of the communications

The typical transmission media is made of a twisted pair, shielded (depending ont he environmental noise), in differential mode connected to a differential amplifier (CAN_H and CAN_L, ok if the noise is the same on the two cables)

Close to linear dependency (log log) on an electric transmission media (1 Mb/s within 40 m)



NRZ (not return to zero) base band codification: sinchronization more critical

Logic transmitted levels are defined as "dominant" and "recessive" and their relative correspondence with "1" and "0" depends on the type of bus connection (AND/OR)

In case of contemporaneous transmission of a "r" bit together with "d" bit by another node, the dominant prevails while the transmitter of the "r" notices the conflict since it reads the bus content and verifies the difference between qhat it has transmitted and what it detects on the bus

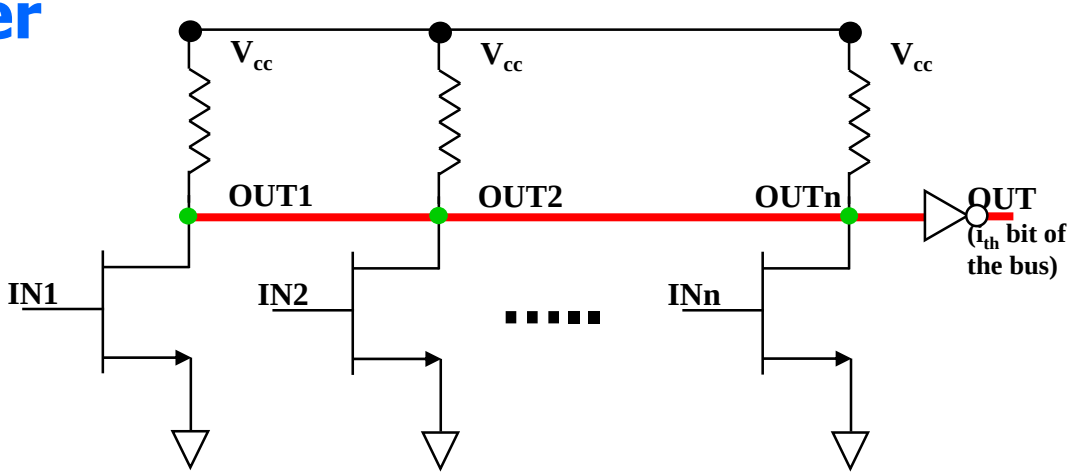
CAN Bus: physical layer

CONNECTION WIRED OR

OUT = IN1 ¦ IN2 ¦ … ¦ INn

Dominant level "1"

Recessive level "0"

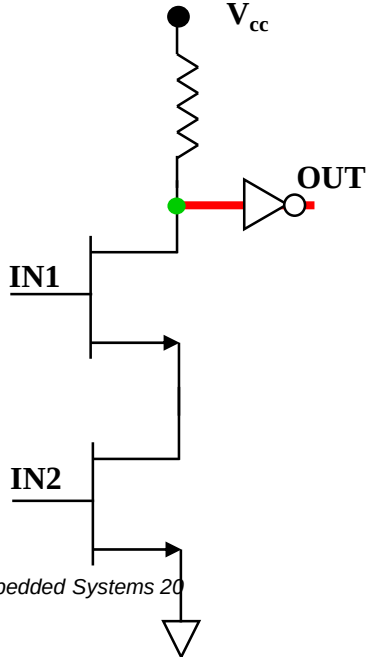


CONNECTION WIRED AND

OUT = IN1 & IN2

Dominant level "0"

Recessive level "1"

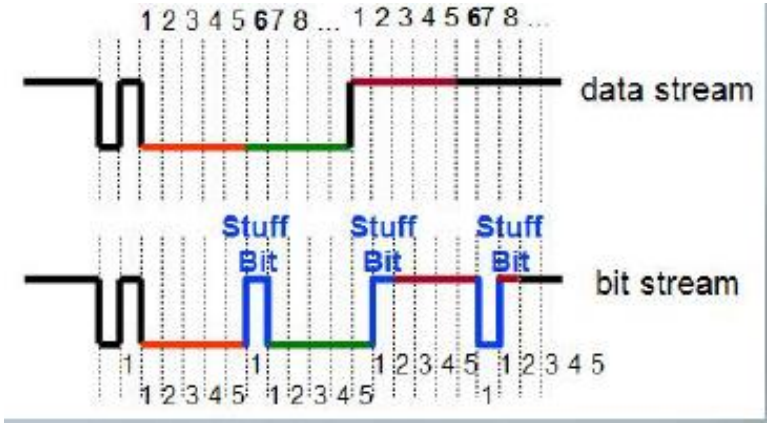


	Node 1 Dominant	Node 1 Recessive
Node 2 Dominant	*dominant*	*dominant*
Node 2 Recessive	*dominant*	*recessive*

CAN Bus physical layer: bit timing configuration

It consists of 3 sub-layers: PLS (Physical Signaling), PMA e MDI

PLS manages the signal temporization and synchronization through the so-called "bit stuffing" technique: every 5 consecutive equal bits, an inverse bit is inserted that will be discarded when received to improve the phase synchronization

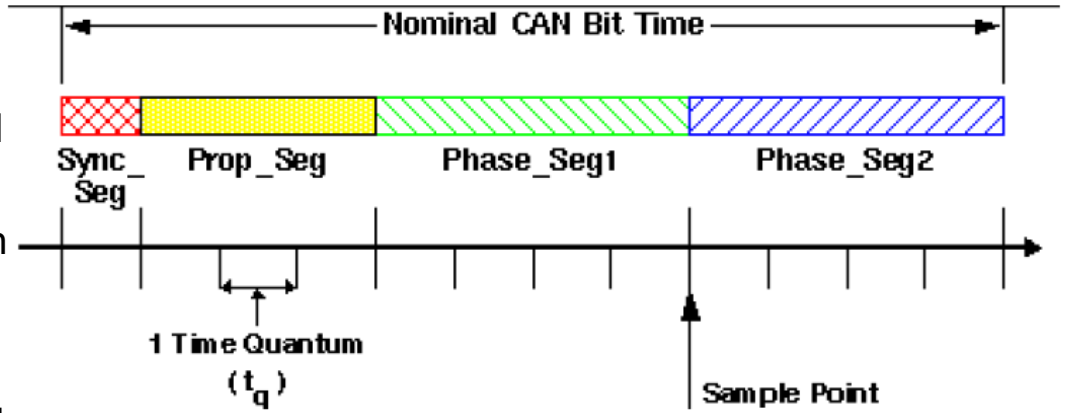


The time duration of a bit (Nominal Bit Time = 1/bit_rate) is divided in 4 intervals each multiple of a suitable TIME QUANTUM (TQ) set through a local oscillator:

Synchronization Seg. = 1 TQ where commutation edges must be inserted so as to synchronize the nodes

Propagation Time Seg. = from 1 to 8 TQ, used to compensate delays in tx/rx

Phase Segment 1 e 2 = with the same duration (1-8 TQ) are tuned so as the transition from one to the other (i. e. the sampling point that is the reading of the level present on the bus) is compliant with the requirements of all the present devices (commutation edges)



http://www.kvaser.com/wp-content/uploads/2014/08/boschcia99-can-bit-timing.pdf

CAN Bus physical layer: bit timing configuration

Every node must be able to sample the bit level within the same bit time: for instance in case of arbitration two nodes must "read" in the same time (bit time) dominant and recessive values put on the bus by the other nodes

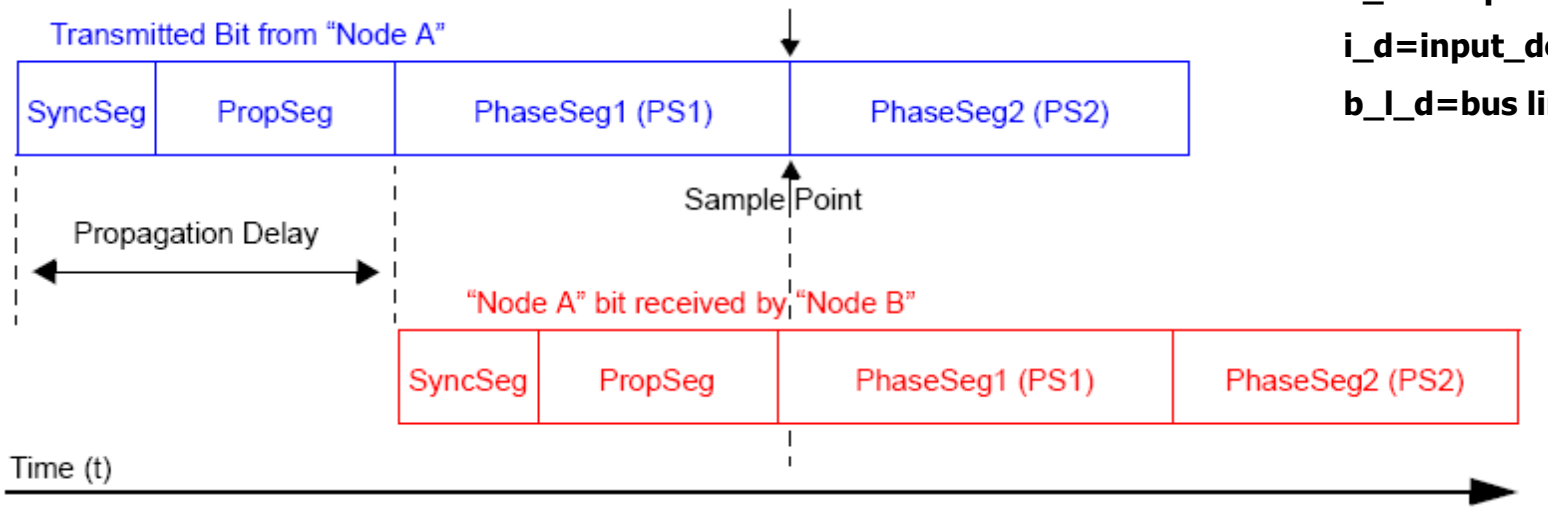
In the example below if propagation delays bring one of the nodes (B) to sample much after the sample point of another one (A) with whom there is a competition, this could make the arbitration unsuccessful

If the bit rate is fixed the length of the connections must be limited: t_{prop}= 2 (o_d+b_l_d+i_d)

o_d=output delay

i_d=input_delay

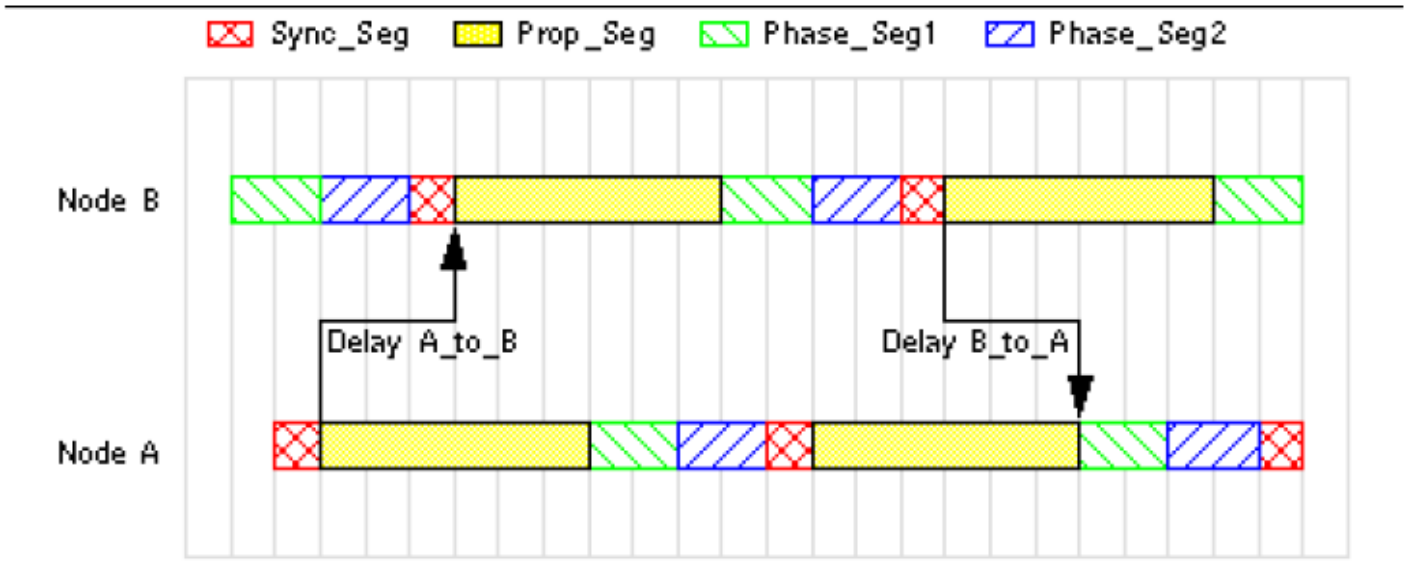
b_l_d=bus line delay



PLS – (partial) compensation of phase shift due to the distance

A e B are transmitting: initially A transmits a recessive level while B receives it after a delay due to the distance that separates them

If B transmits a dominant level on its own turn this level must be received before A continues in transmission otherwise A thinks to be the arbitration winner. Since the Sample Point can happen in every point of the Phase_segment (see later), the bit sent by B must arrive to A before Phase_seg1 starts. This defines the length of the Prop_Seg (in the example 6 TQ)

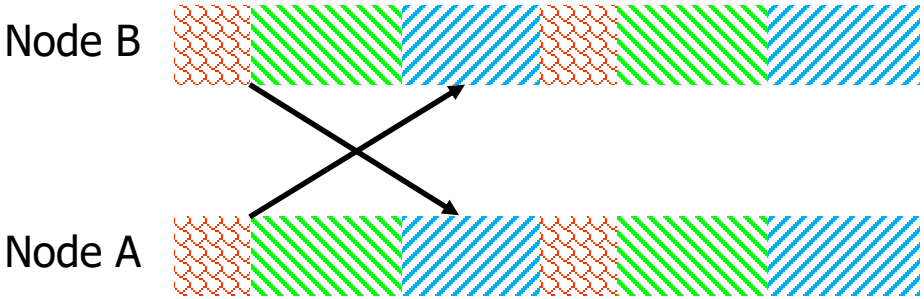


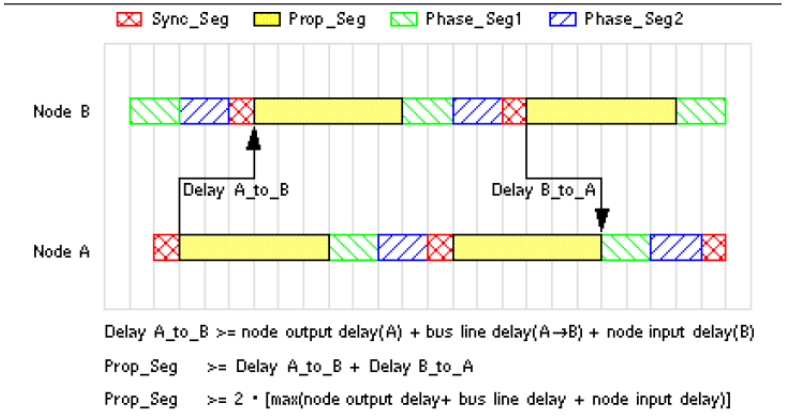
Delay A_to_B >= node output delay(A) + bus line delay(A→B) + node input delay(B)

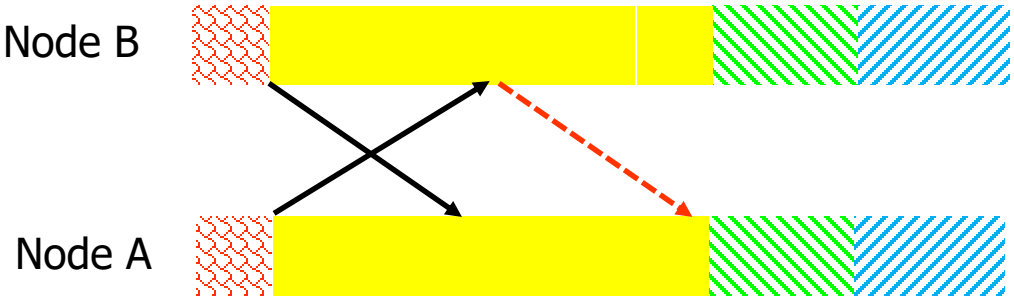
Prop_Seg >= Delay A_to_B + Delay B_to_A

Prop_Seg >= 2 * [max(node output delay+ bus line delay + node input delay)]

PLS – (partial) compensation of phase shift due to the distance







PLS: Phase Segment variation for synchronization

The length of PS is lengthened or shortened to compensate variations in local clocks (synchronization)

The synchronization among nodes is made on a recessive-dominant transition and keeps the distance between the edge and the sample so as to be always the same

The edges are detected on every TQ by comparing the present level of the line with that relative to the last sample point. If the level changes we have 2 possible actions:

Hard Synchronization: the bit time is reset to the moment when the edge arrives. It is done every time the start of frame arrives of the node that begins to transmit.

Bit Resynchronization: the PS are lengthened or shortened considering when the edge arrives; if after the Sync_Seg the phase shift is positive and PS1 is lengthened. If the edge arrives before the Sync_Seg, PS is made shorter.

The maximum shortening/elongation SJW (Synchronization Jump Width) goes from 1 to 4 TQ. Beyond this value the phase shift cannot be completely compensated but only partially and an error equal to (phase error – SJW) remains.

PLS: Phase Segment variation for synchronization

The synchronization is performed not only to compensate clock variations, but also to compensate eventual noise spikes or due to the arbitration.

As already said, the synchronization among the nodes is done when the first node begins to transmit. However, since it is not said that this node will win the arbitration, the other nodes must synchronize with the winner and this operation is continuously performed.

Between two successive Sample Points only one synchronization can be done.

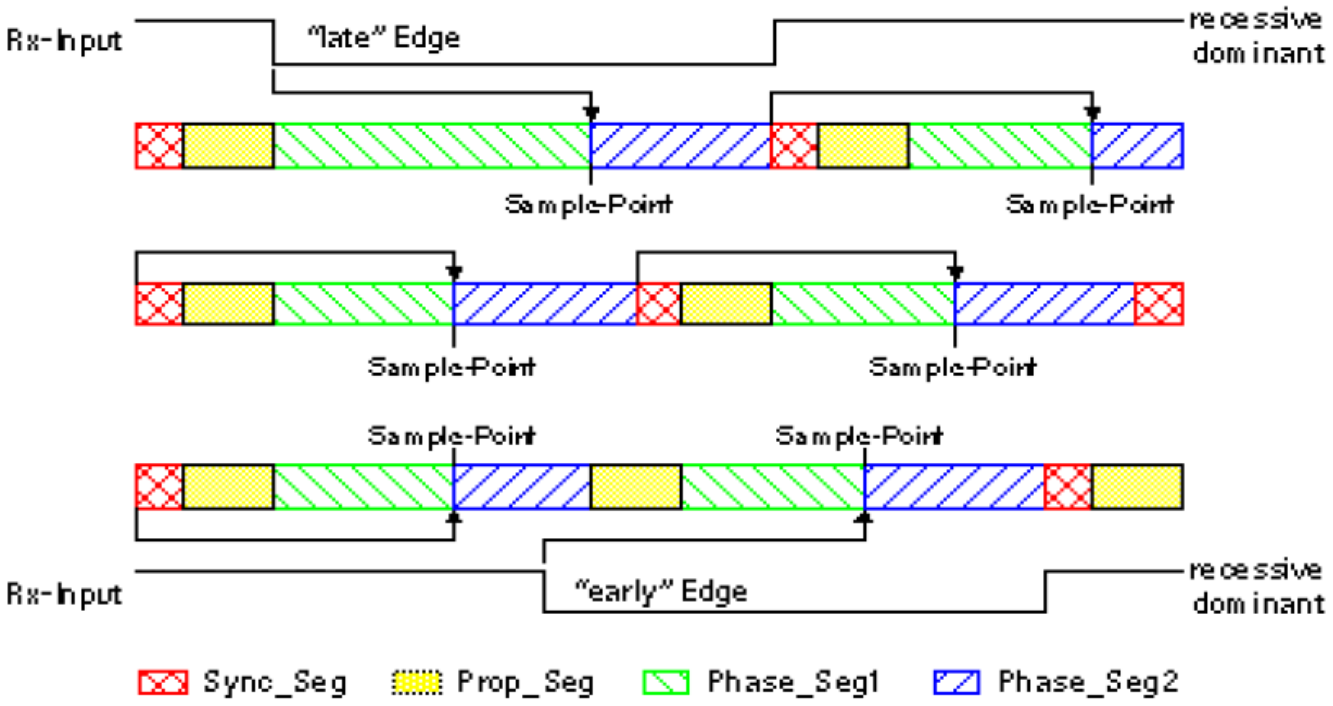
The same operation is done in the acknowledge field where the sender and the receiver synchronize themselves with the bit of «reception happened» (see later).

Phase Segment variation in case of phase shift

In the first stream the edge is successive to the Sync Seg. PS1 is lengthened so as the distance from the Sample Point is the same compared when no delay is present.

In the second stream the edge arrives before the Sync Seg and is in PS2. Now the PS2 will be shortened and Sync cancelled so as to maintain the same distance between edge and SP as without anticipation.

These variations are temporary but they imply a variation of the bit time.



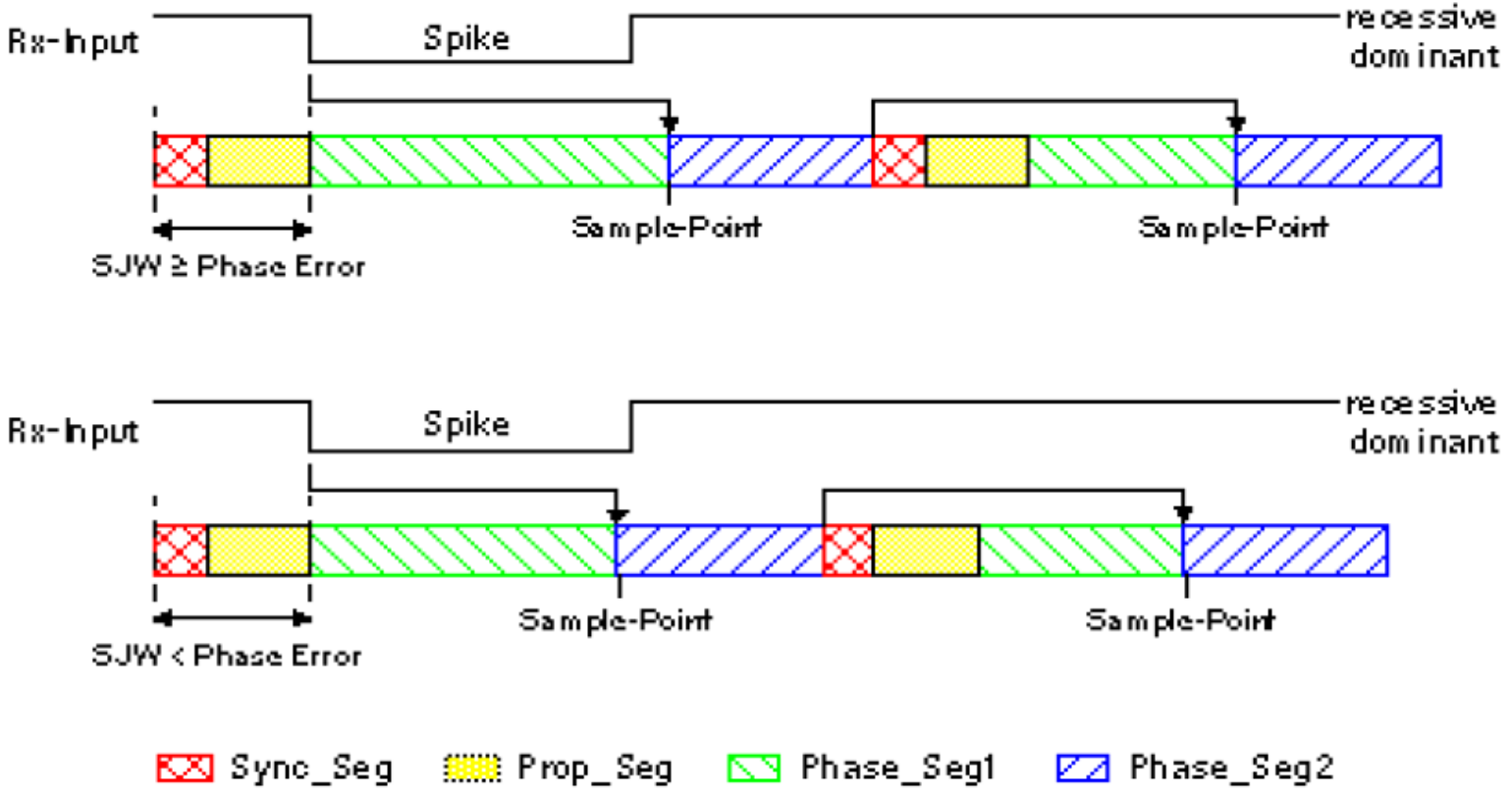
Synchronization due to a delayed edge

Synchronization as it would be

Synchronization due an edge in advance

PLS: Phase Segment variation for synchronization

The picture below shows how a dominant spike will be filtered by the proposed synchronization technique.



CAN Bus: physical level – PMA e MDI

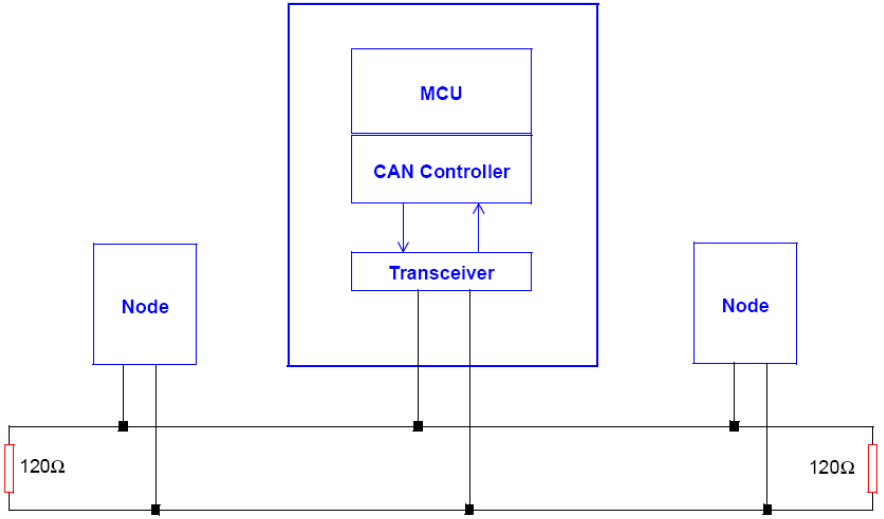
PMA (Physical Medium Attachment)

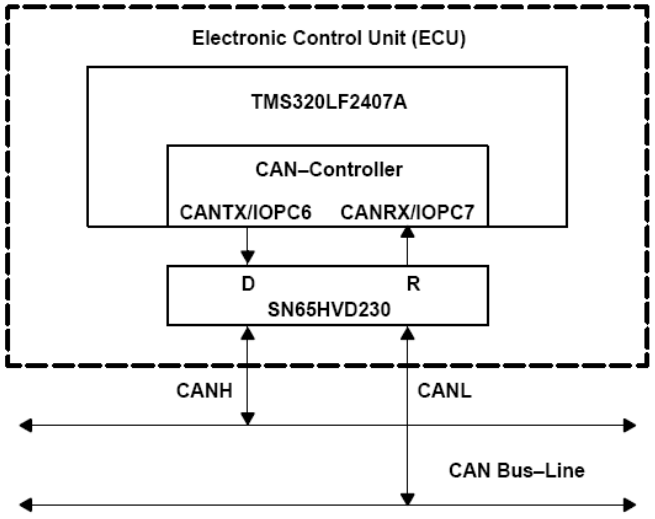
It defines the structure of the connection of every node to the bus through three components:

- Transceiver: bi-directional buffer to make access to the bus (detects the difference between the two lines CAN_H e CAN_L)
- Controller: manages tx/rx from and to the bus
- Microcontroller: "intelligent" part of the node controlling the other two

MDI - Medium Dependent Interface

It defines the physical and electrical characteristics in terms of connection type, adaptation impedance (120 Ω), line delay …





CAN Bus: Data Link Layer

DLL is charged with:

- **Message format definition**
- **Message validation**
- **Error management**
- **Error active nodes confinement**
- **Arbitration**

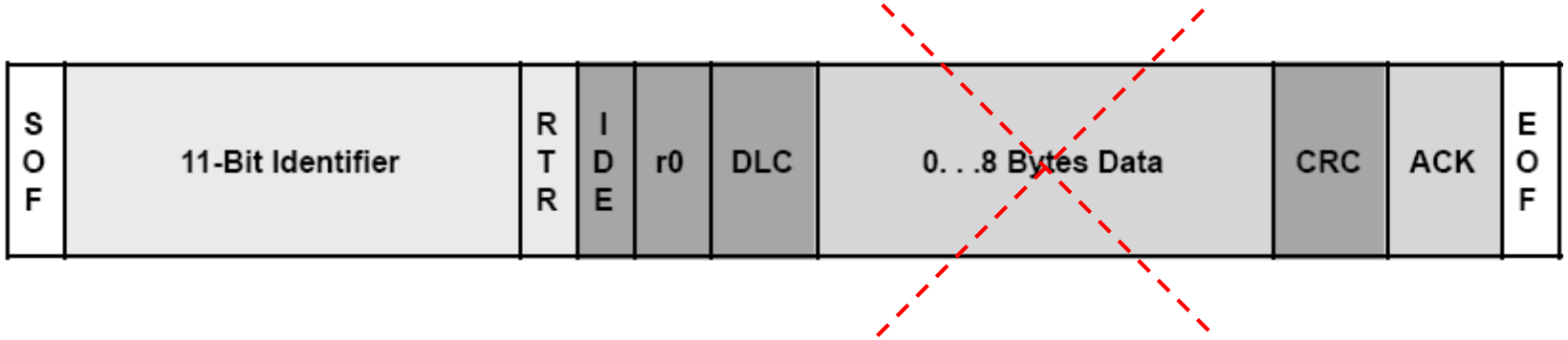
Message format:

- **Data Frame:** messages with data that must be kept by each node or discarded
- **Remote frame:** similar to DF but without data. It means "data request"
- **Error frame:** sent by a node that detects an error and causes the retransmission of a message. Each node calculates its own error rate: if high self-exclusion (fault confinement)
- **Overload:** sent by a "busy" node to ask to delay the next message dispatch
- **Interframe Space:** sent before each DF e RF, to separate them

CAN Bus: Data Link Layer – Data frame

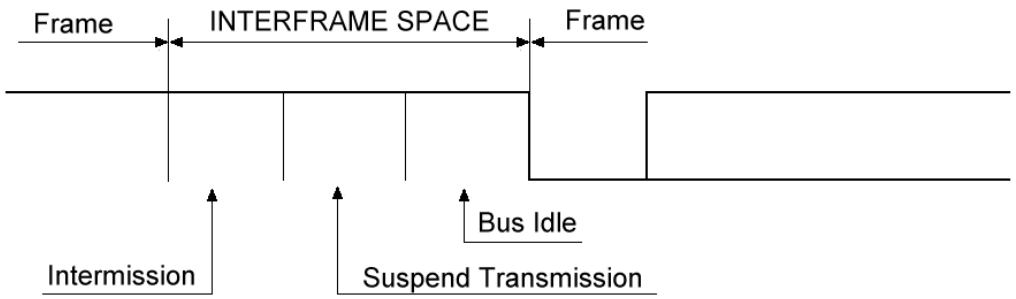
- **Start of frame:** 1 bit "dominant" = synchronism for beginning of DF (or RF)
- **Arbitration field (11/29 bit extended):** identifier of the message content plus bit RTR (Remote Transmission Request) to distinguish between DF and RF: in fact this bit is dominant for DF and recessive for RF. Thus if a request of datum simultaneous with their transmission, the request is aborted
- **Control field (6 bit):** 4 (DLC) specify the length of the Data Field (0-8 byte) while 2 (IDE and r0) are reserved
- **Data frame:** from 0 to 8 data bytes
- **CRC field:** 15 bit for CRC + 1 "recessive". These bits are added to the frame so as the sequence is perfectly divisible by a prefixed polynomial term. If the validation is ok (null remainder), in ACK a dominant bit is written.
- **Ack Field:** 2 bits, ACK Slot e ACK Delimiter. By default they both are recessive: if at least one destination node received the message (although we do not who), it overwrites the ACK Slot with a dominant bit
- **End of frame:** 7 recessive bits

CAN Bus: Data Link Layer – Remote frame



- **Very similar to the Data Frame with 2 important differences:**
  - **no data field**
  - **RTR is forced to be "recessive" so as to specify that this is a Remote Frame**
- **Used to ask for data whose type is specified in the "identifier" field**
- **The transmitter will answer with the information required in the data field**

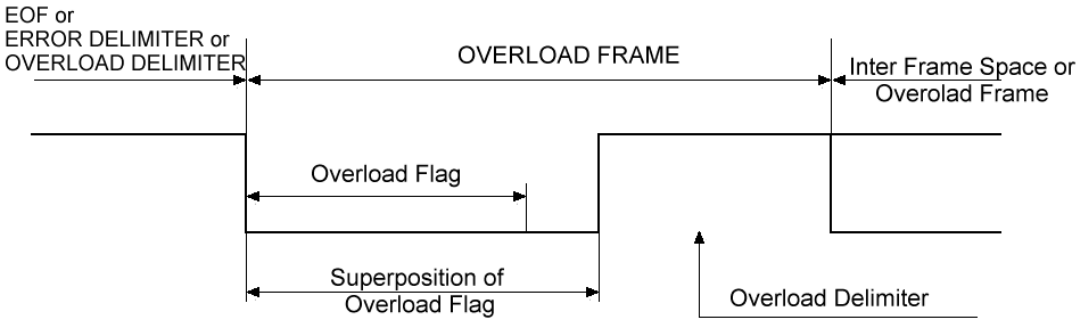
CAN Bus: Data Link Layer – Interframe space



- **Sequence of recessive bits to separate DF and RF**
- **Not used if Overload and Error are sent (they can also be consecutively dispatched)**

- **Intermission: the first 3 bits are "recessive" and it is not allowed to send other DF or RF; the Intermission bits can be interrupted only by an Overload Frame that prevents from sending the remaining ones. The Overload cannot break a DF**
- **Suspend Transmission: 8 recessive bits used by a node featuring transmission error rate not negligible. This node must wait 8 bits before sending again; if, in the meanwhile, another node will start a transmission, the node, if "error passive", can receive the message. In any case if the node is not "error passive" or it only receives, the suspend transmission field does not exist**
- **Bus Idle: sequence of "recessive" bits with not specified length that are interpreted as "free bus" and "ok to transmit" notification. This sequence ends when the SoF is encountered of a new Data or Remote Frame**

CAN Bus: Data Link Layer – Overload Frame



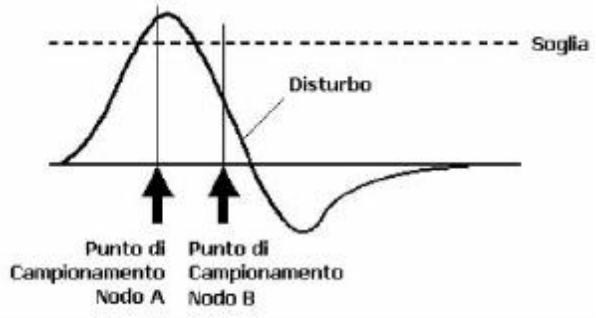
- Sequence of bits that delays the DF and RF emission if a node is "busy" and cannot correctly process other data. It is allowed to send it only correspondingly to the first 3 bits of an Intermission frame

- In other words a "dominant" bit received during the I and II bit of an Interframe is interpreted as condition of "other busy node" that can be acknowledged by sending another Overload
- The Frame consists of two fields: Flag and Delimiter
- Flag: 6 "dominant" bits that can be sent within the Intermission field of an Interframe. In this case all the other stations of the bus detecting this condition answer on their turn with a Overload flag. If the 6 bits are sent after they will be interpreted as SoF of an DF or an RF generating an error condition since they violate the bit stuffing condition
- The Overload frames delay all nodes communication, therefore only 2 consecutive Overload are allowed by the same receiver. A mechanism is given to confine those nodes that due to their slowness are not able to process the received messages
- Delimiter: 8 recessive bits of ending frame

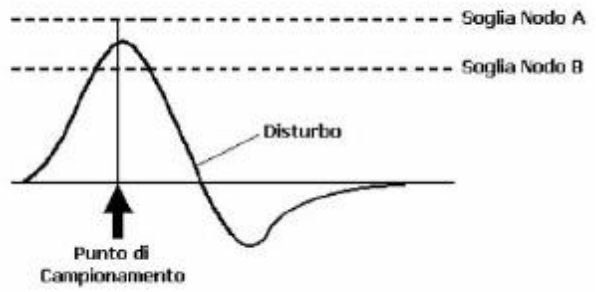
CAN Bus: Data Link Layer – Error Frame

Possible error sources are both due to mechanisms internal to the nodes and to noise

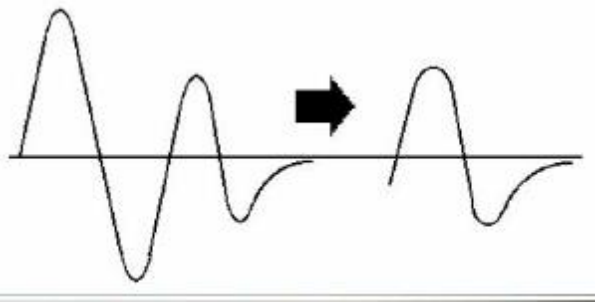
•phase shifts in sampling time,



• different discrimination thresholds



•signal attenuation on the line (reading error)



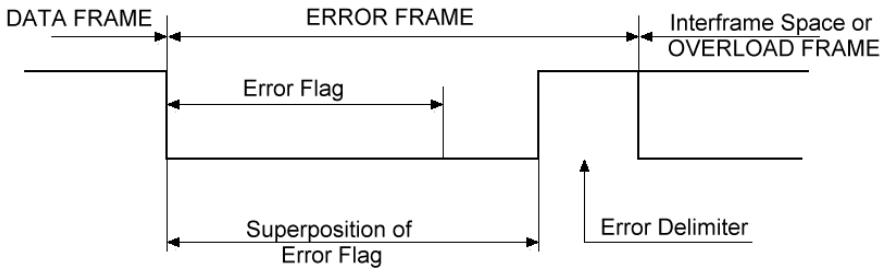
CAN Bus: Data Link Layer – Error management

- Error detection consists of the following steps:
  - The node controller detects an error and transmits an Error Frame
  - All the connected nodes ignore the not valid message
  - The controller is upgraded
  - The message is re-transmitted
- The detectable errors:
  - *Bit monitoring*: if the bits on the bus are different from those that were sent (by the winner of the arbitration) a bit error occurs. This can happen everywhere but in the "identifier" field where it is considered as loosing the bus control (see later: bus arbitration)
  - *Bit stuffing*: if 6 consecutive equal bits are on the bus a Stuff Error is detected
  - *Frame Check*: a format not corresponding to one of the 5 possible triggers a Form Error
  - *CRC*: the received code does not correspond to the received sequence a CRC error occurs
  - *Acknowledgement*: if the sent frame has been received, the ACK field must be overwritten with a "dominant" bit. If the transmitter does not detect this, it notifies an acknowledgement error

CAN Bus: Data Link Layer – Fault confinement

- Each node has two fault counters for reception (REC) and transmission (TEC) that in case of faults add 1 for a receiving error and 8 for transmission error. If the transmission or the reception are ok they are diminished by 1.
- On the basis of the values of its own counters, each node can set itself as:
  - **Error active: both the counters are < 128**. The error rate does not disrupt the normal traffic on the bus so the node can play an active role and send error frames if necessary
  - **Error passive: if at least one counter is > 128**. The node causes faults quite frequently and is allowed to send only ***passive*** error frames (no automatic re-transmission of the messages) but must wait a greater interval than active nodes before sending again (interframe with suspend)
  - **Bus off: if one counter is = 256**. The node is seriously affected by errors so the access to the bus is forbidden.
  - Suitable bit sequences have been conceived in the standard that can be sent by the user code in the application layer, to re-enable a "bus off" node and to reset it as "error active" (with REC=TEC=0). For example a sequence with 128 frames with 11 consecutive recessive bits.
- This error management policy is very efficient so as a not detected fault can be considered a very rare event during the bus activity.

CAN Bus: Data Link Layer – Error Frame



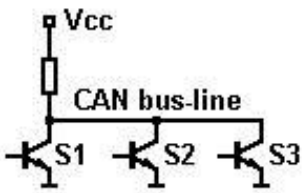
That node detecting an error in a DF or RF sends the Error Frame

The Frame consists of two fields: Flag and Delimiter

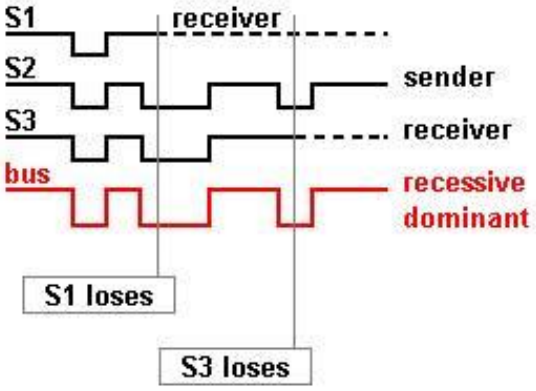
- Flag: 6 bits dominant or recessive depending on the type of node
  - active error flag (dominant): if a node is qualified as "error active" by the fault confinement system detects an error, it sends on the bus 6 "dominant" bits that violate the stuffing rule causing the error detection by the other connected nodes (they will send in turn error flags). The traffic present at that moment is destroyed
  - passive error flag (recessive): they are sent by a node qualified as "passive". They can be overwritten by an Active Error Flag but they can't overwrite the ACK and EOF fields of a DF thus they can't destroy the traffic
- Superposition of error flags: the other 6 bits sent by the other nodes (at maximum 12 dominants bits, all in all)
- Delimiter: 8 recessive bits identifying the End of the Frame

CAN Bus: Data Link Layer – Arbitration

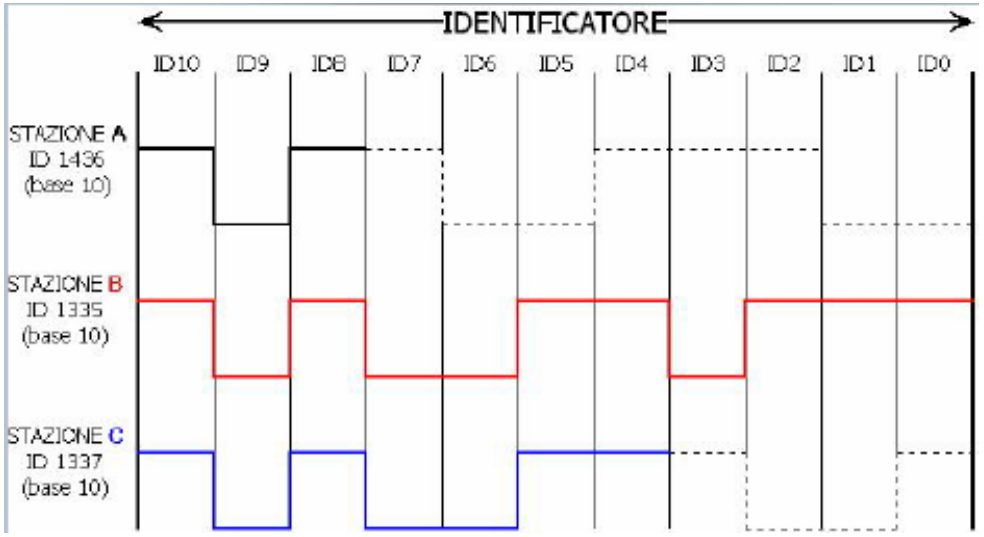
- When a node transmits write the SOF first and then its own identifier: if many nodes try to contemporaneous transmission the winner among them will be the minor value owner
- Pre-emptive priority mechanism
- During the arbitration a node monitors the bus and compares the read level with the transmitted one: if the two bits do not correspond and in particular if the node sent a recessive bit, it stops its own transmission.



Wired nor



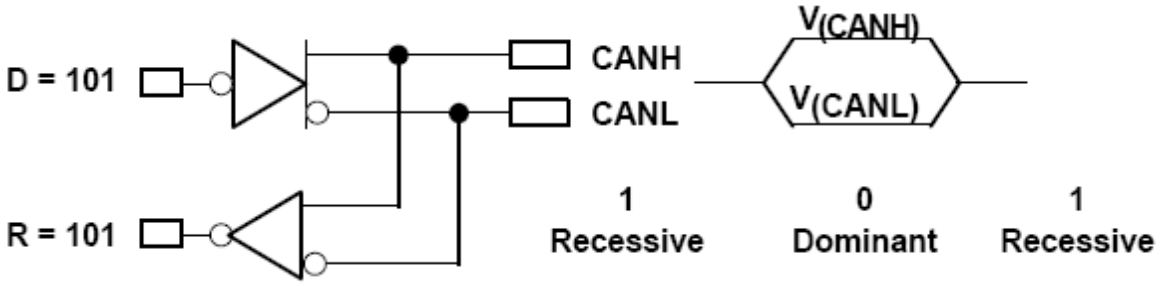
"0" dominant

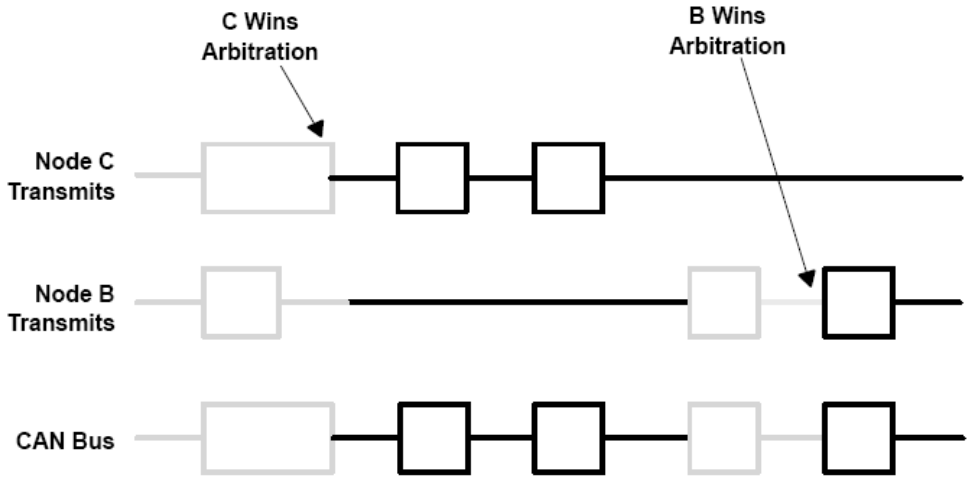


The minor identifier wins

CAN Bus: Data Link Layer – Arbitration

- In a CAN node the logic level "1" is associated to the recessive level while the "0" to the dominant.
- This happens since the line must be set "recessive" by the transceiver when not driven so as to minimize the power consumption (if the level is pulled high no current passage).
- During the arbitration each node monitors the bus and compares the read level with that transmitted: if the two bits are not coincident and in particular if the node sent a recessive bit, it breaks its transmission.





CAN Bus: Data Link Layer

- Object Layer: filtering (the packet received can belong to the node or not), interpretation of the received messages, interfacing with Application level
- Transfer Layer: error detection and management, faults confinement, arbitration, handshake and synchronization, message packing and packet distribution