Lezione 13
Esempi di dispositivi di I/O

http://www.dii.unisi.it/~giorgi/didattica/arcal1

- **Timer**
- **Seriale**

All copyrighted figures are copyright of respective authors.
Figures may be reproduced only for classroom or personal educational use only when the above copyright line is included.
They may not be otherwise reproduced, distributed, or incorporated into other works without the prior written consent of the publisher.

Timer Intel-8254

- A cosa puo' servire:
  - Generare ritardi programmabili in maniera accurata
  - Real time clock
  - Conteggio eventi
  - Generatore di frequenza programmabile
  - Generatore di onda quadra programmabile
  - Generatore di "impulso digitale"
  - Moltiplicatore binario di frequenza
  - Generatore di forma d'onda complessa
  - Controllo di motori

- E' un buon esempio per mostrare la complessita' interna di un'interfaccia di I/O

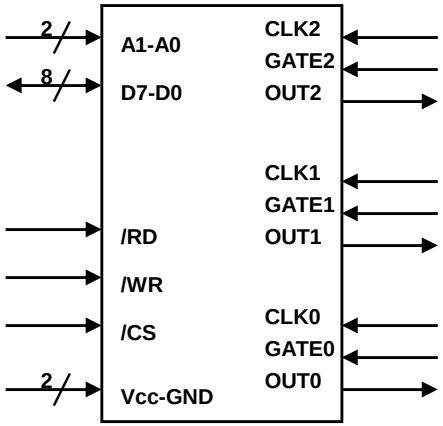
Timer Intel-8254

- Visione Funzionale del chip:
  - 3 contatori indipendenti a 16-bit
  - 6 modalita' di conteggio

- Ogni timer/counter supporta sei modi operativi:
  - MODO 0: Interrupt al termine del conteggio
  - MODO 1: Programmable one shot
  - MODO 2: Rate generator
  - MODO 3: Square wave rate generator
  - MODO 4: Software triggered strobe
  - MODO 5: Hardware triggered strobe

Timer Intel-8254

- Visione Elettrica del chip:

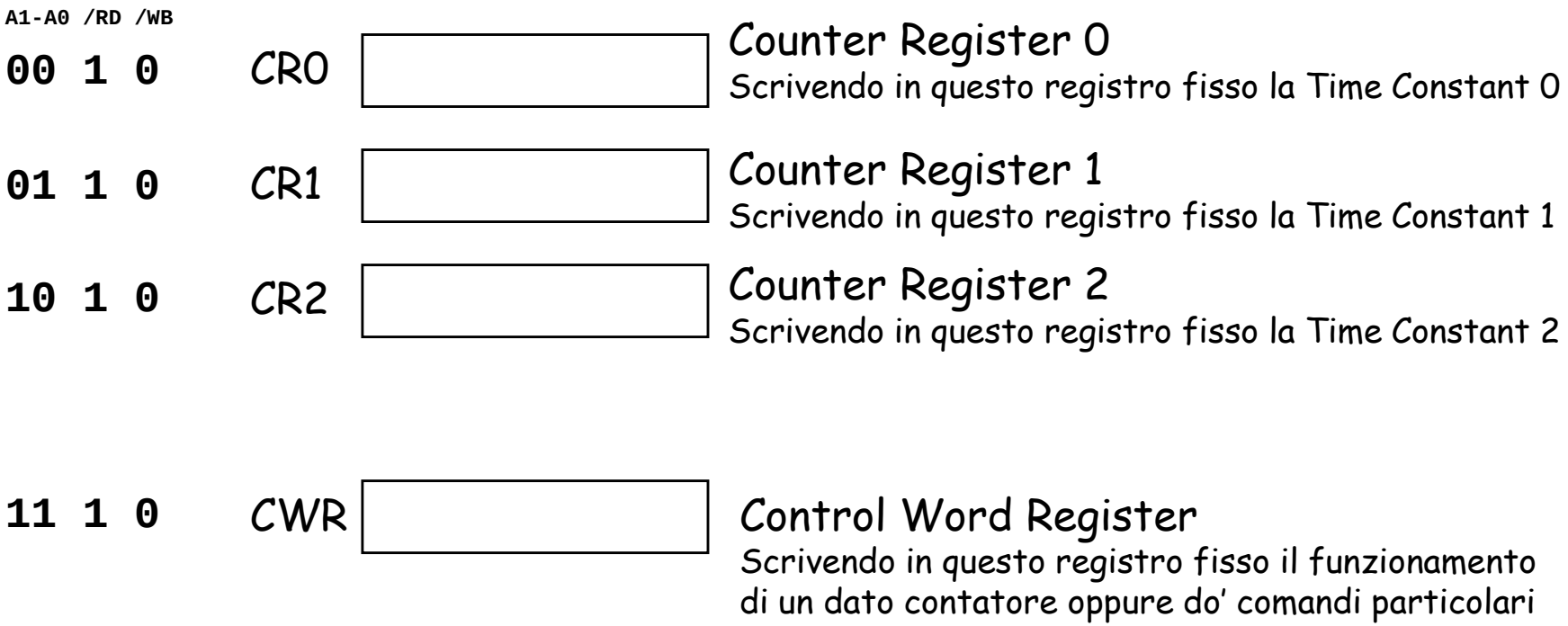


- A1-A0: indirizzamento
- D7-D0: scambio dati su 8 bit
- /RD, /WR, /CS: Read, Write, Chip-Select
- Vcc-GND: alimentazione

- CLKj: clock del timer j
- GATEj: trigger hardware o congelamento conteggio
- OUTj: forma d'onda prodotta dal timer j

Timer Intel-8254 (2)

- **Visione logica del chip (tutti registri a 8 bit):**



Nota: la Time Constant e' a 16 bit. CRj funziona come un "bus" a 8 bit verso un registro interno a 16 bit che contiene tale Time Constant

Programmazione del timer

1) Scrivere in CWR per selezionare uno dei contatori e il *modo*
2) Scrivere in CRj la costante di tempo
3) Abilitare il conteggio con un segnale alto sul pin GATE
4) L'uscita OUT produrra' un segnale di uscita alla fine del (oppure durante il) conteggio a seconda del modo prescelto

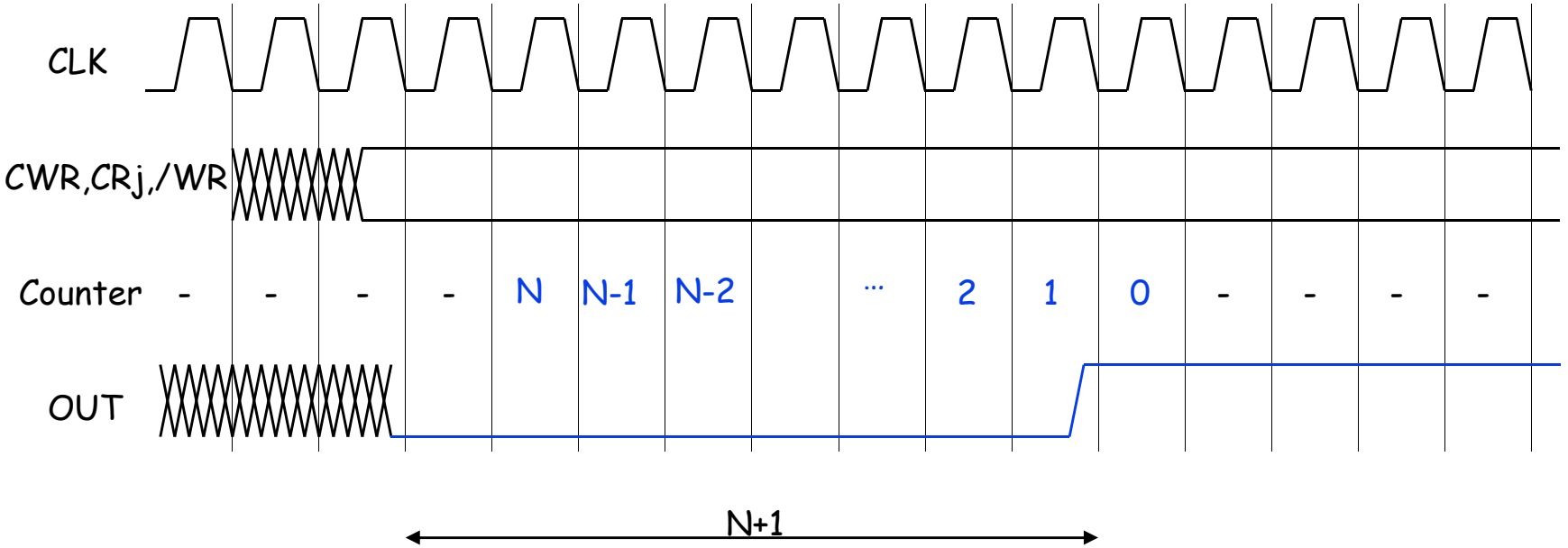
Modalita' di lettura/scrittura

- Poiche' nei dispositivi di I/O si tende ad utilizzare un numero limitato di indirizzi (nel nostro caso 2) si usa un registro per dare i comandi (CWR) e gli altri registri per leggere o scrivere dati (CR0,CR1,CR2)

- I contatori sono a 16 bit ma posso scrivere solo 8 bit per volta
  - Devo percio' specificare, nelle operazioni di lettura/scrittura, se sto lavorando:
    - Solo sul byte meno significativo (LSB)
    - Solo sul byte piu' significativo (MSB)
    - Prima sul byte meno significativo e poi su quello piu' significativo (LSB+MSB)

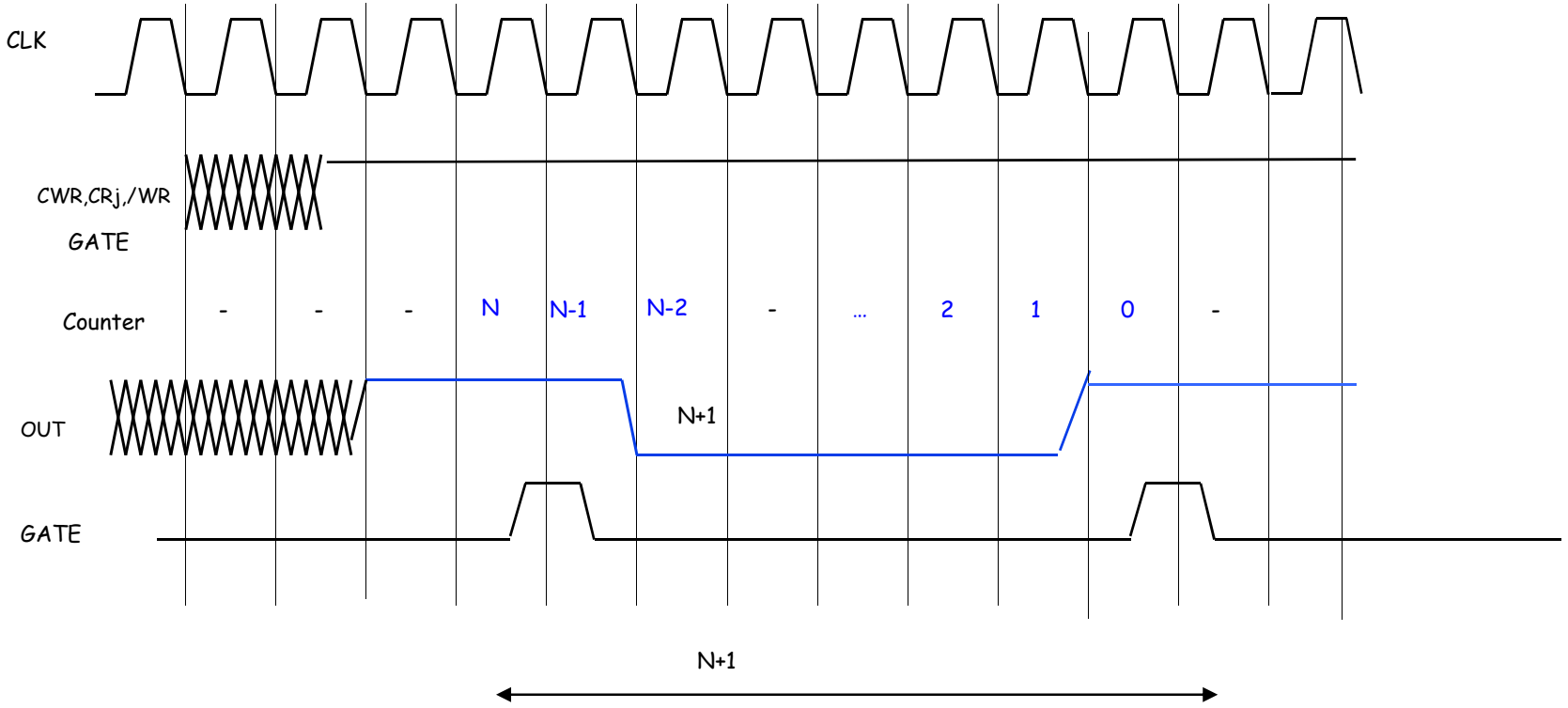
Modi di conteggio

- **Modo 0: interrupt al termine del conteggio**



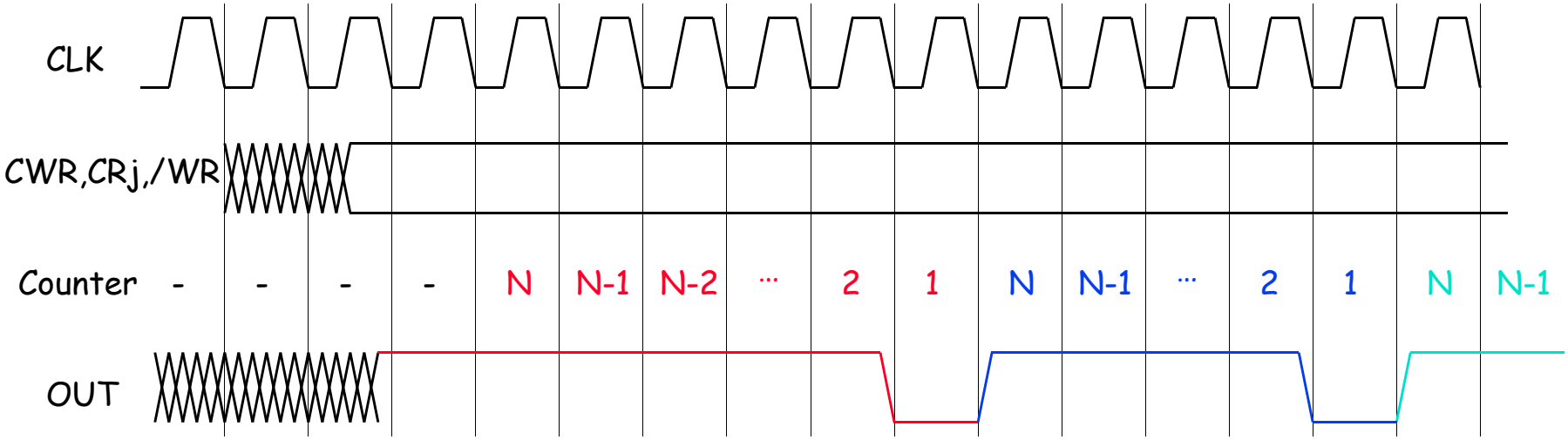
Modi di conteggio

- Modo 1: one shot. L'uscita è inizialmente alta e diventa bassa sull'impulso di clock che segue il trigger (pulsazione one shot), e vi rimane fino a quando il contatore non raggiunge lo zero.



Modi di conteggio

- **Modo 2: rate generator (divisore di frequenza)**



- **Il duty cycle e' (N-1)/N**

- **Se f e' la frequenza del segnale di clock, la forma d'onda d'uscita avra' frequenza f/N.**

Lettura del conteggio (1)

- Ci sono tre maniere per leggere il valore di un contatore
  1) lettura semplice
  2) Counter Latch Command
  3) Read-Back Command

- La lettura semplice consiste nel leggere il valore di CRj previo bloccaggio del conteggio tramite il segnale GATEj
  - Se non si blocca il conteggio il valore letto puo' essere inconsistente

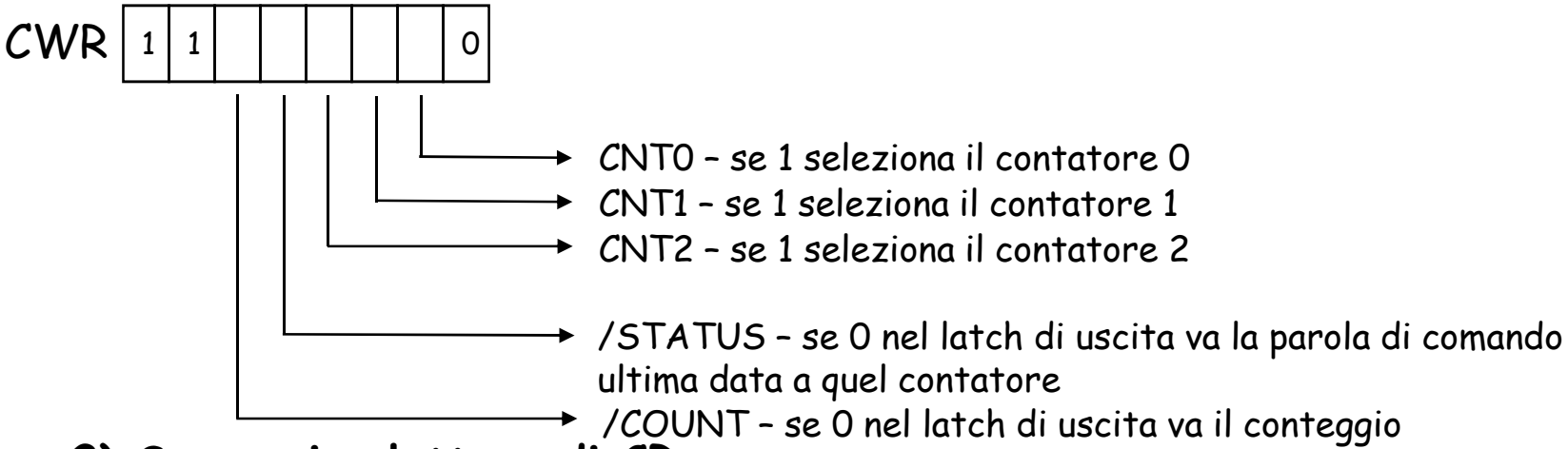
Lettura del conteggio (2)

- **Counter Latch Command**
  - **Serve per leggere il valore corrente del contatore**
  - **Si realizza in questo modo**
    - CWR ← cc00 ' - - - -
      (cc=0, 1 o 2 indica il contatore, '-' indica non specificato)
    - Lettura di 1 o 2 byte da CRcc
      (a seconda dello schema di lettura/scrittura impostato)

Lettura del conteggio (3)

- **Read-Back Command**

  1) Si realizza in questo modo: scrittura in CWR delle seguenti info



  2) Successiva lettura di CRx

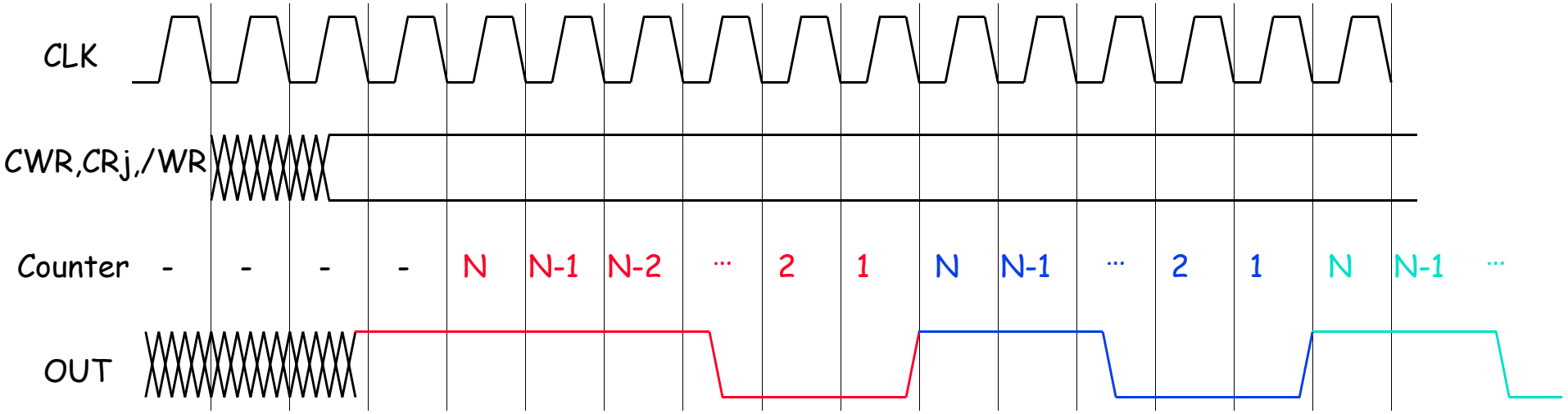
- **Vantaggi:**
  - **Consente di leggere i conteggi di piu' contatori simultaneamente (caso di bit /COUNT attivo)**
  - **Consente di leggere come sono stati programmati i 3 contatori (caso di bit /STATUS attivo)**
    - **In particolare, i 6 bit meno significativi (danno i valori precedentemente scritti in CWR per quel contatore**

Ulteriori dettagli

- E' possibile
  - Programmare vari altri modi (es. modi 3,4,5 – v. slide successive)
  - Leggere lo stato relativo ad altre informazioni interne

Modi di conteggio

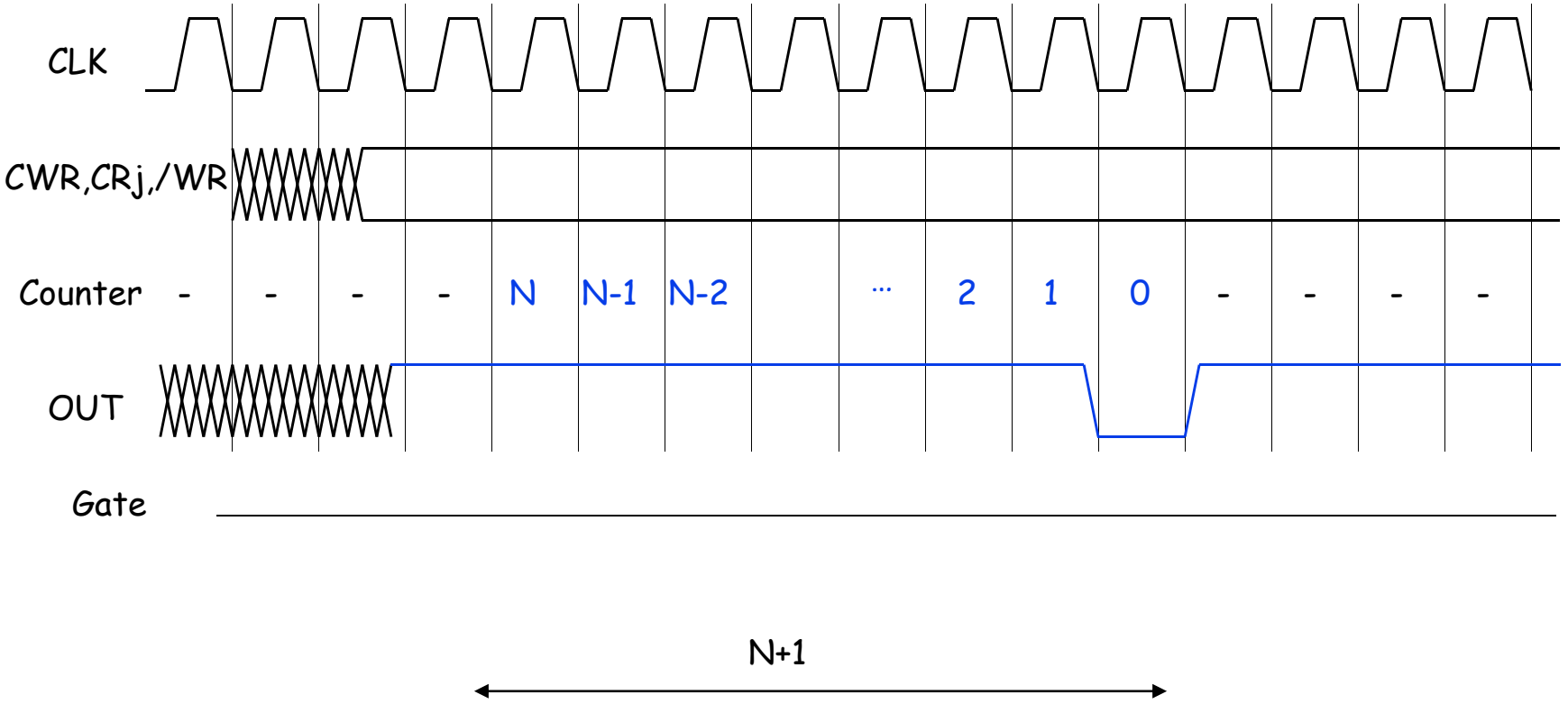
- Modo 3: square wave mode (generatore d'onda quadra)



- OUT rimane alto nei periodi N, N-1, … , (N+1)/2
- OUT rimane basso nei periodi N/2, (N-1)/2, … , 2, 1
- Il duty cycle e' ½ se N e' pari, altrimenti e' ((N+1)/2)/N

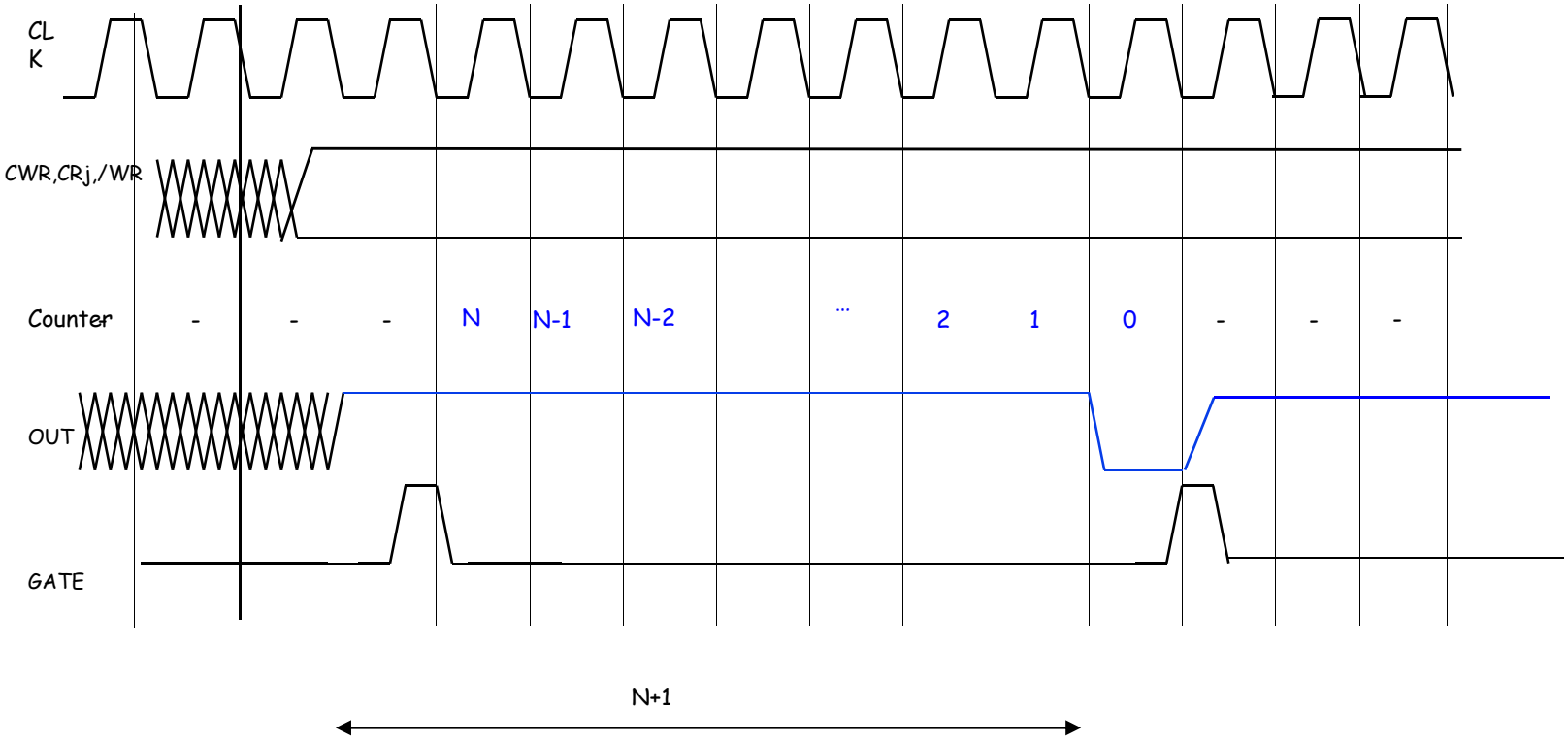
Modi di conteggio

- **Modo 4: software triggered strobe (generatore d'impulso)**



Modi di conteggio

- **Modo 5: Hardware triggered strobe (generatore d'impulso)**



Il Chip 8254 nei Personal Computer

- PORTE DI I/O ASSEGNATE
  - CR0 ≡ 0x0040
  - CR1 ≡ 0x0041
  - CR2 ≡ 0x0042
  - CWR ≡ 0x0043
- CLK0=CLK1=CLK2 = 1.19MHZ
- TIMER-0: timer di sistema
  - GATE0 sempre attivo
  - OUT0 genera IRQ0
- TIMER-1: timer per il rinfresco della memoria
  - GATE1 sempre attivo
  - OUT1 genera impulsi con un periodo di 15ms
- TIMER-2: generatore di beep
  - GATE2 controllabile tramite bit0 di SPR,Speaker Register (porta di I/O 0x0061)
  - OUT2 in AND col bit1 di SPR, l'uscita va poi sul beeper

Riepilogo risorse 8254

Chip name	Register name		I/O port	IRQ
Intel 8254	CR0	Counter Register 0	0x0040	IRQ0
Intel 8254	CR1	Counter Register 0	0x0041	-
Intel 8254	CR2	Counter Register 0	0x0042	-
Intel 8254	CWR	Comtrol Word Register	0x0043	-

Glue logic	SPR	Speaker Register	0x0061	

Esempio di programmazione del Timer 8254

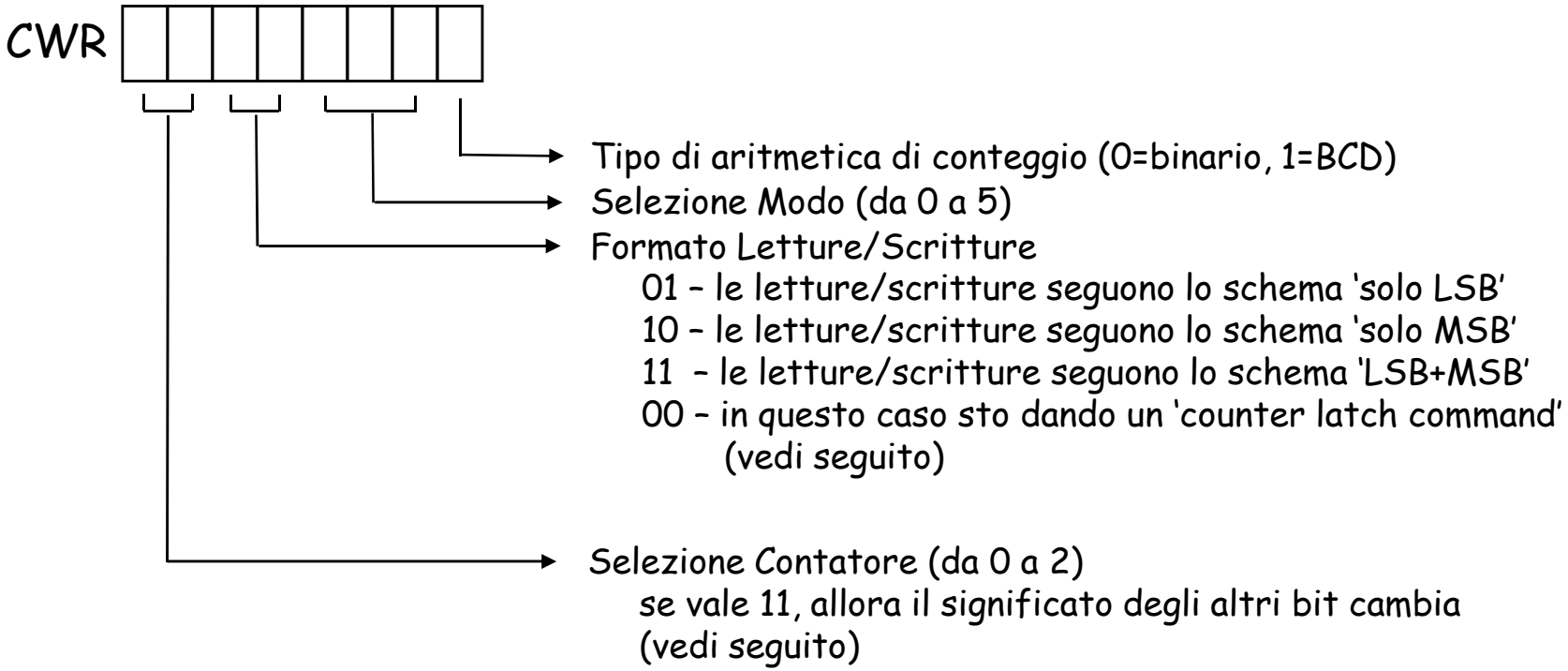
- **Generare una interruzione ogni 50ms**
  - **Tc=1/1.19MHz=840ns**
  - **N+1=50·10^{-3}/Tc → N=59499**

```
int init_8254 (void)
{
    int cr_l, cr_h;


    outp(0x0043, 0x30);  /* contatore 0, costante di tempo a 16 bit,
                                        modo 0, conteggio binario */
    cr_l = 59499 & 0x000000FF;
    cr_h = (59499 >> 8) & 0x000000FF;
    outp(0x0040, cr_l);
    outp(0x0040, cr_h);
}
```

→V. SLIDE SUCCESSIVA

Programmazione del Timer (2)



Es.
CWR ← 1011'0110 significa che il timer #2 deve effettuare
un conteggio binario in modalita' #3,
e le scritture/letture seguono lo schema LSB+MSB

Comunicazione Seriale Asincrona

Comunicazione Seriale Asincrona (ASC)

- **Comunicazione**: ad un certo istante, e' possibile individuare
  - Un dispositivo che agisce da trasmettitore (Master o TX)
  - Un dispositivo che agisce da ricevitore (Slave o RX)

- **Seriale**: i bit che compongono il dato vengono trasmessi sul mezzo trasmissivo in sequenza
  - La trasmissione potra' essere anche wireless!
  - La trasmissione potra' usare 1 filo o piu' fili elettrici (modalita' differenziale, cfr. Ethernet baseT)

- **Asincrona**: la trasmissione dei dati avviene seguendo un protocollo di comunicazione asincrono
  - Nessun segnale di clock e' associato alla comunicazione

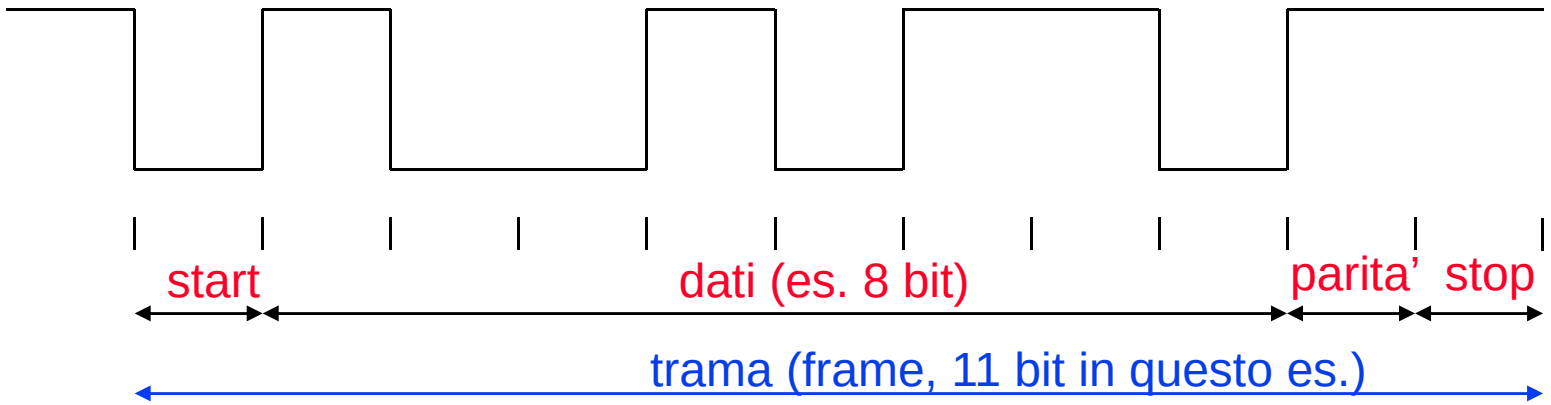
Caratteristiche principali di una ASC

- La velocita' della comunicazione e' caratterizzata dal numero di bit trasmessi per secondo o bit-rate
  - Se T e' il tempo per trasmettere un bit, 1/T e' il bit-rate

- Il dato e' costituito da una trama (frame) di bit
  - L'informazione e' tipicamente racchiusa tra un header o preambolo e un trailer o coda
  - L'header serve a marcare l'inizio della trasmissione
  - Il trailer serve a marcare la fine della trasmissione e a trasmettere un codice di verifica (CRC o parita')

- Ricevitore e trasmettitore si sincronizzano usando l'header
  - Il clock in ricezione cerca cosi' di ottenere stessa fase del clock del trasmettitore: la frequenza deve pero' essere nota!

Trama seriale



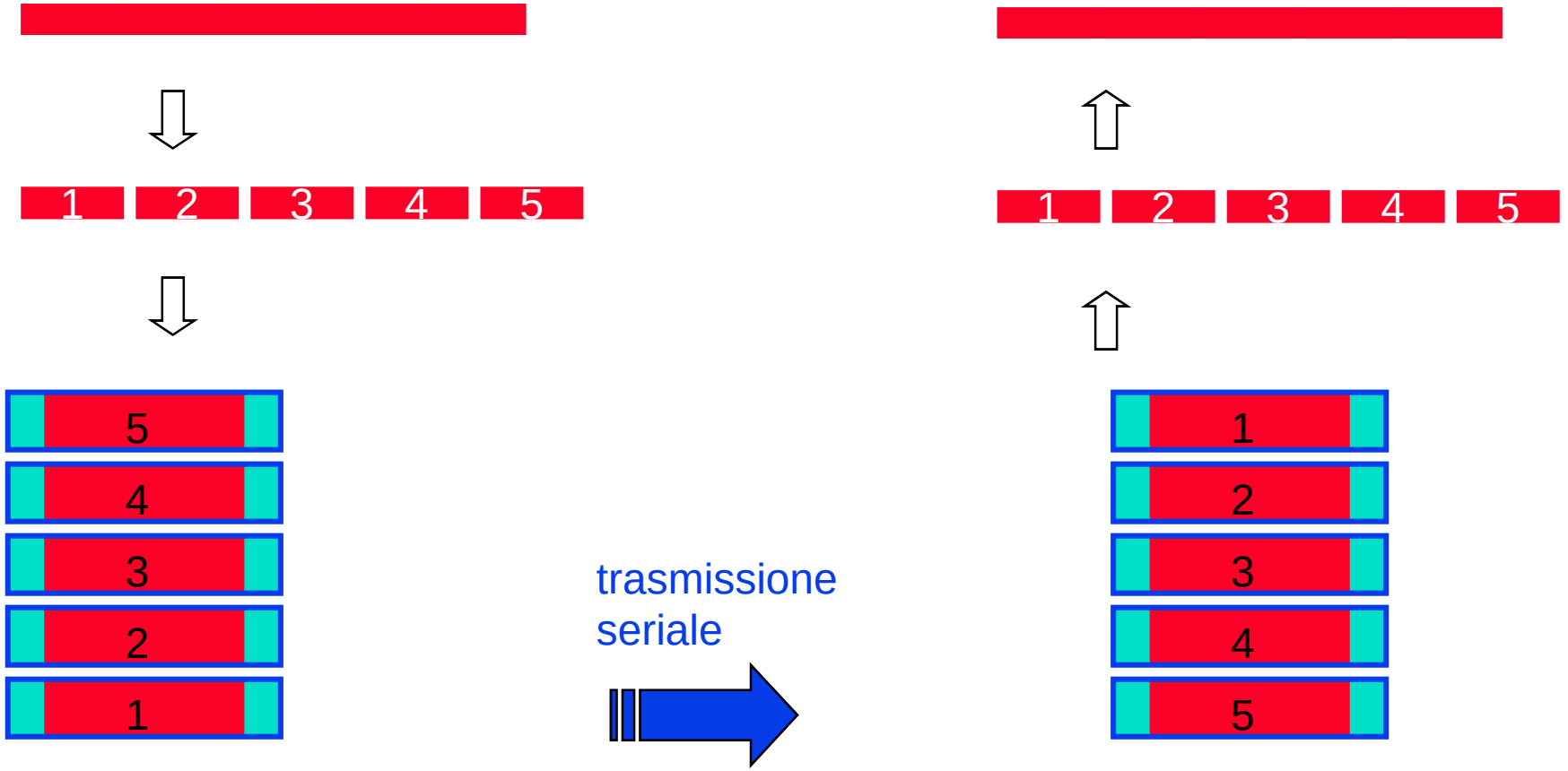
Frame comunicazione seriale

Informazioni che e' necessario sapere a priori

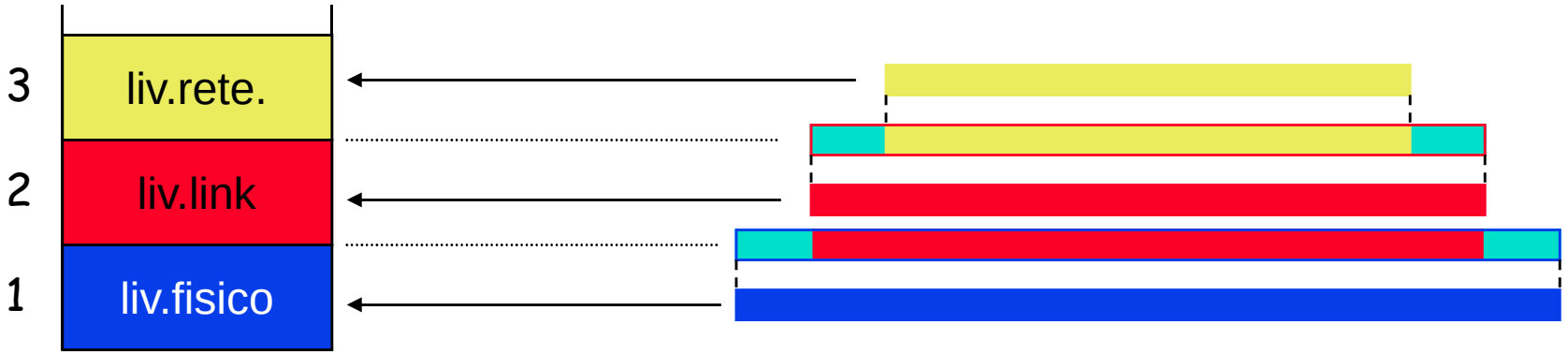
- Velocita' di trasmissione (bit-rate, es. 1200 bit/s)
- Formato della trama (es. 1 bit di start, 8 bit dati, 1 bit di parita', parita' positiva, 1 bit di stop)

Pacchettizzazione

Es. Trasmissione di 5 byte



Stratificazione (Stack di Comunicazione)



Stack di comunicazione:

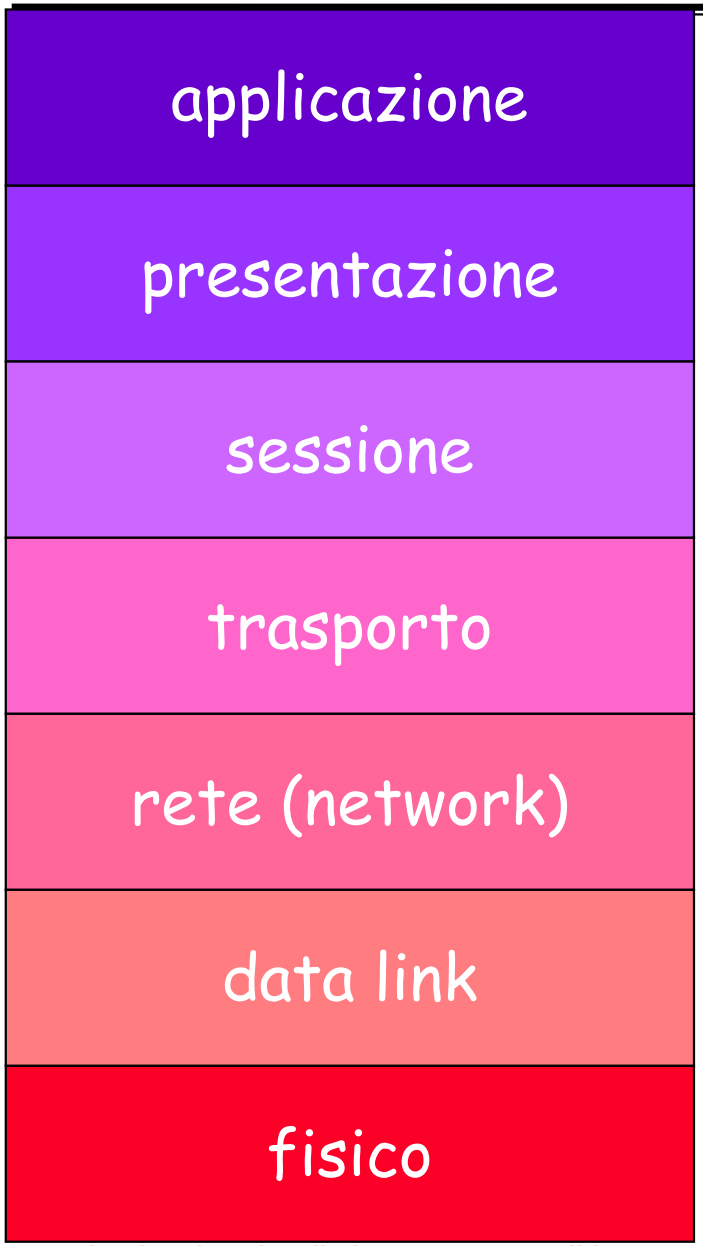
- **verso l'alto, ogni livello fornisce una comunicazione con piu' proprieta'**
- **verso il basso i dati vengono 'arricchiti' con ulteriori info di controllo**

Gestione delle reti per astrazioni successive

- International Standards Organization (ISO) ha sviluppato un modello per descrivere le reti chiamato Open Systems Interconnection (OSI)
  - Modello a 7 livelli
  - Standard noto come ISO/OSI

- Fornisce uno standard per classificare i componenti delle reti e le operazioni coinvolte

Modello OSI



interfaccia utente

formato dati
(tipi dei dati, trasformazione dei servizi)

comunicazione fra le applicazioni ai due estremi (raggruppamento dati, checkpointing)

connessione
(gestione connessione, ottimizzazione gestione di rete)

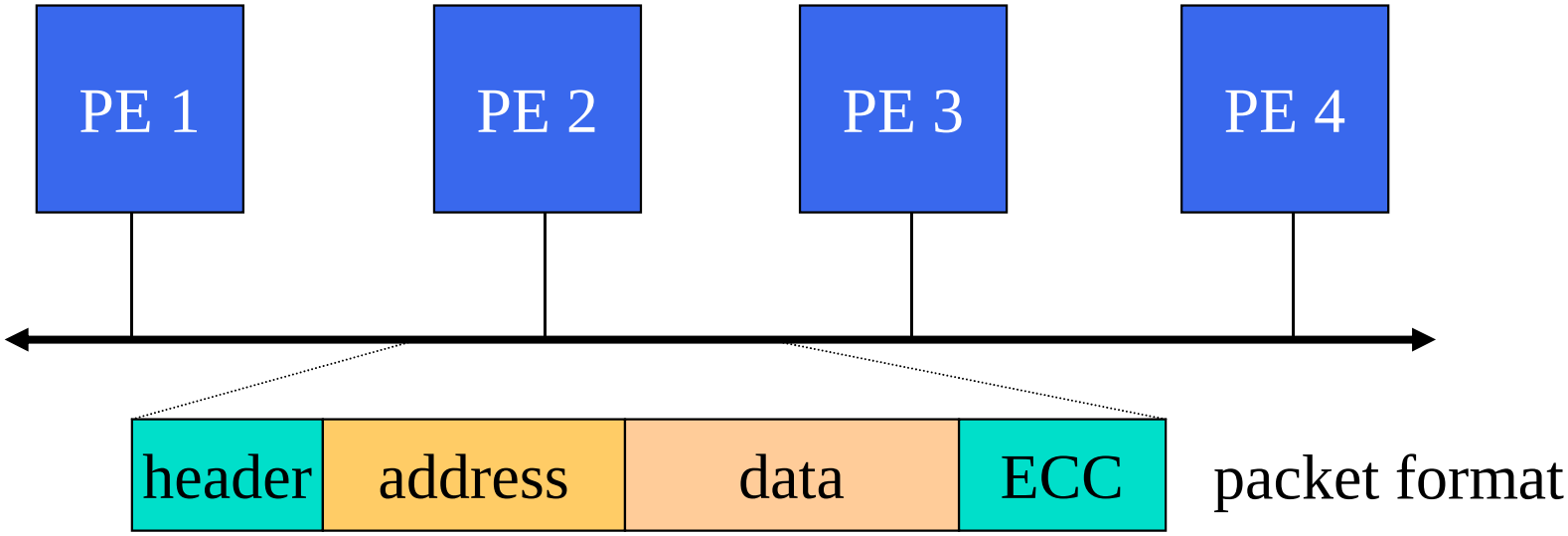
servizio end-to-end
(comunicazione dati multi-hop end-to-end)

trasporto dati affidabile
(rilevazione errori e gestione singolo link (single-hop))

caratteristiche meccaniche/elettriche
(connettori, codifica dei bit,...)

Livelli 1 & 2: Reti Seriali a Bus

- Una unica connessione fisica comune



- I2C, USB, FireWire, Ethernet, …

Standard Ethernet

Standard Ethernet – Livelli 1 & 2

- Risale ai primi anni'80
- Standardizzato come IEEE 802.3
- Dominante nelle reti locali (LAN)
- Versioni:
  - 10 Mb/s (IEEE 802.3, o Ethernet)
  - 100 Mb/s (IEEE 802.3u, o Fast Ethernet)
  - 1 Gb/s (IEEE 802.3ab, o Gigabit Ethernet)
  - 10 Gb/s (IEEE 802.3ae, o 10-Gigabit Ethernet)
  - 100 Gb/s (IEEE 802.3ba, o 100-Gigabit Ethernet)

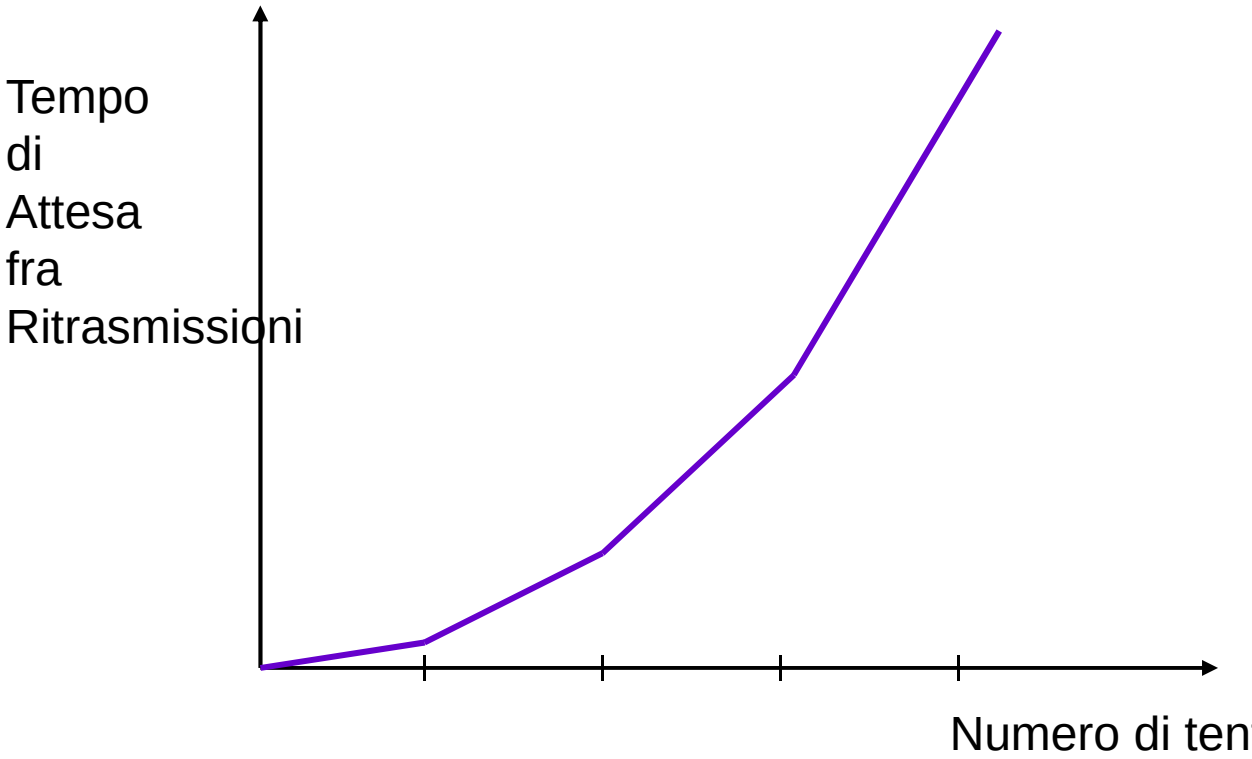
  - Wireless LAN (IEEE 802.11, o Wi-Fi)
  - Low-Power WLAN (IEEE 802.15, o Bluetooth)
- Topologia: "a bus"
- Obiettivo: ottenere una comunicazione affidabile usando un mezzo trasmissivo eventualmente non affidabile

CSMA/CD, Carrier Sense Multiple Access with Collision Detection

- Un nodo "ascolta prima di parlare" (carrier sense)
- Se il mezzo e' libero si attende un tempo "Defer Time"
- Se ci sono collisioni (multiple access) l'energia sul mezzo aumenta
- Un nodo cerca di rilevare collisioni durante la trasmissione (collision detect)
- Se un nodo rileva una collisione, continua a trasmettere 4-6 bytes per essere sicuro che la collisione venga rilevata dagli altri nodi e poi "lascia" il mezzo
- Se un nodo ha rilevato collisione, la trasmissione viene ripetuta dopo R*Δt per altri 15 tentativi
- Dopo 16 tentativi l'errore viene segnalato ai livelli superiori
- Δt e' fisso mentre R viene calcolato secondo un "algoritmo di back off"

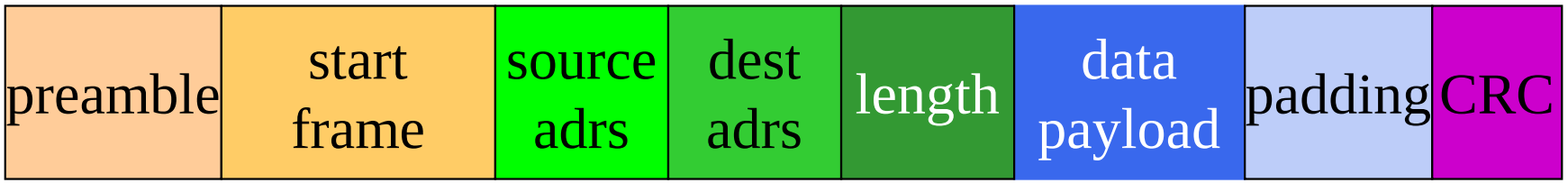
Algoritmo di back off per il calcolo di R

- Sia n l'n-esimo tentativo di trasmissione (n=1,2,…, 15)
- K= min(n,10)
- R e' un numero casuale t.c. 0 <= R <= 2^K



Risolve in maniera probablisticamente accettabile i conflitti fino a 1024 nodi…

Formato del Pacchetto Ethernet



- In dettaglio
  - Preamble — 10101010101010… — 8 Bytes
  - Start frame — 11 — 8 Bytes
  - Source address — 6 Bytes — Ethernet FRAME
  - Destination addres — 6 Bytes — Ethernet FRAME
  - Length — 2 Bytes — Ethernet FRAME
  - Data Payload — i dati veri e propri! — 46-1500 Bytes — Ethernet FRAME 64-1518 Bytes
  - Padding — Eventuali byte in coda al payload — 46-1500 Bytes — Ethernet FRAME 64-1518 Bytes
  - CRC — 4 Bytes — Ethernet FRAME 64-1518 Bytes

- Segue un IPG (Inter Packet Gap) di almeno 96 bit…

Prestazioni della rete Ethernet

- La Qualita' del servizio (Quality-of-Service, o QoS) tende a decrescere in maniera non-lineare ad alti livelli di carico
- Una conseguenza dell'algoritmo di back off e' che non e' possibile stabilire il tempo massimo entro cui verra' trasmessa con successo una trama
  - Non si possono garantire deadline real-time!
  - Si possono ottenere livelli molto buoni di servizi purche' si riesca a mantenere il carico ad un livello appropriato

Porta Seriale

Porte Seriali

- Sui Personal Computer e' presente la cosiddetta porta seriale

- Metodo efficiente di comunicazione tra dispositivi
  - Tipicamente denominata UART (Universal Asynchronous Reciever and Transmitter)
  - Es. 16550A, 16450, 8250, 8251A, MC68681

- Semplice porta seriale costituita da:
  - 2 registri a scorrimento (shift register)
    - Uno trasmettitore (Tx)
    - L'altro ricevente (Rx)
  - Uscita del Tx collegata all'ingresso del Rx
  - Trasmissione e ricezione guidate dallo stesso clock
  - Quando Tx è vuoto o Rx è pieno (ricevuto 1 byte) viene generato un interrupt

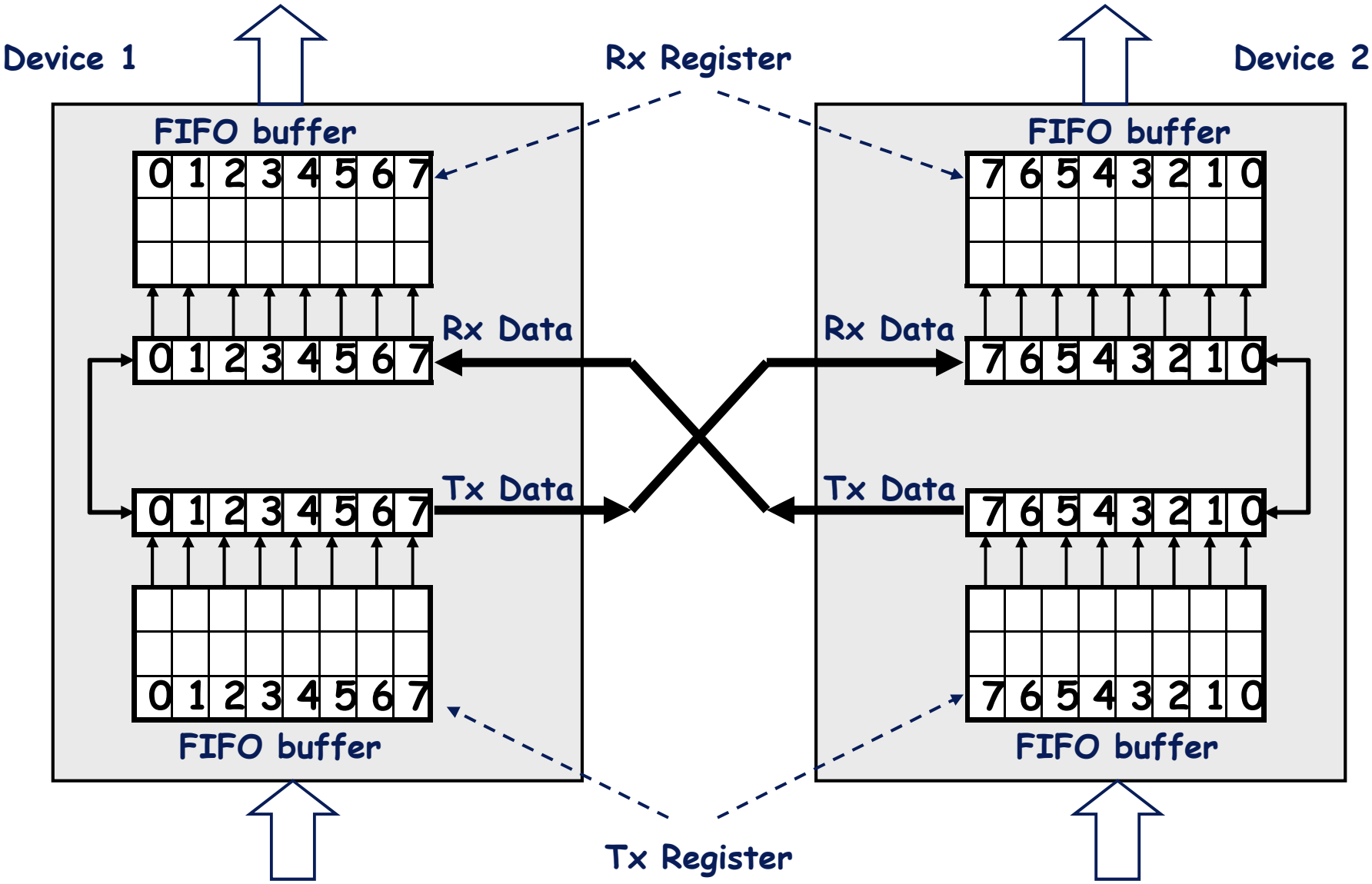
Struttura fisica della porta seriale

- Trasmissione seriale asincrona full-duplex
  - trasmissione contemporanea in entrambi i versi: i due dispositivi fungono sia da Tx che da Rx
  - Sono sufficienti 3 fili di collegamento (Tx,Rx,GND)
- Trasmissione seriale asincrona half-duplex
  - Ad un certo istante, trasmissione in una sola direzione: uno dei due dispositivi funge da Tx l'altro da Rx
  - Sono sufficienti 2 fili (Tx, GND)
- Fisicamente costitita da
  - 2 registri a scorrimento (shift register)
    - Uno per trasmettere (Tx)
    - Uno per ricevere (Rx)
  - Uscita del Tx collegata all'ingresso del Rx
  - Quando Tx è vuoto o Rx è pieno (ricevuto 1 byte) viene generato un interrupt

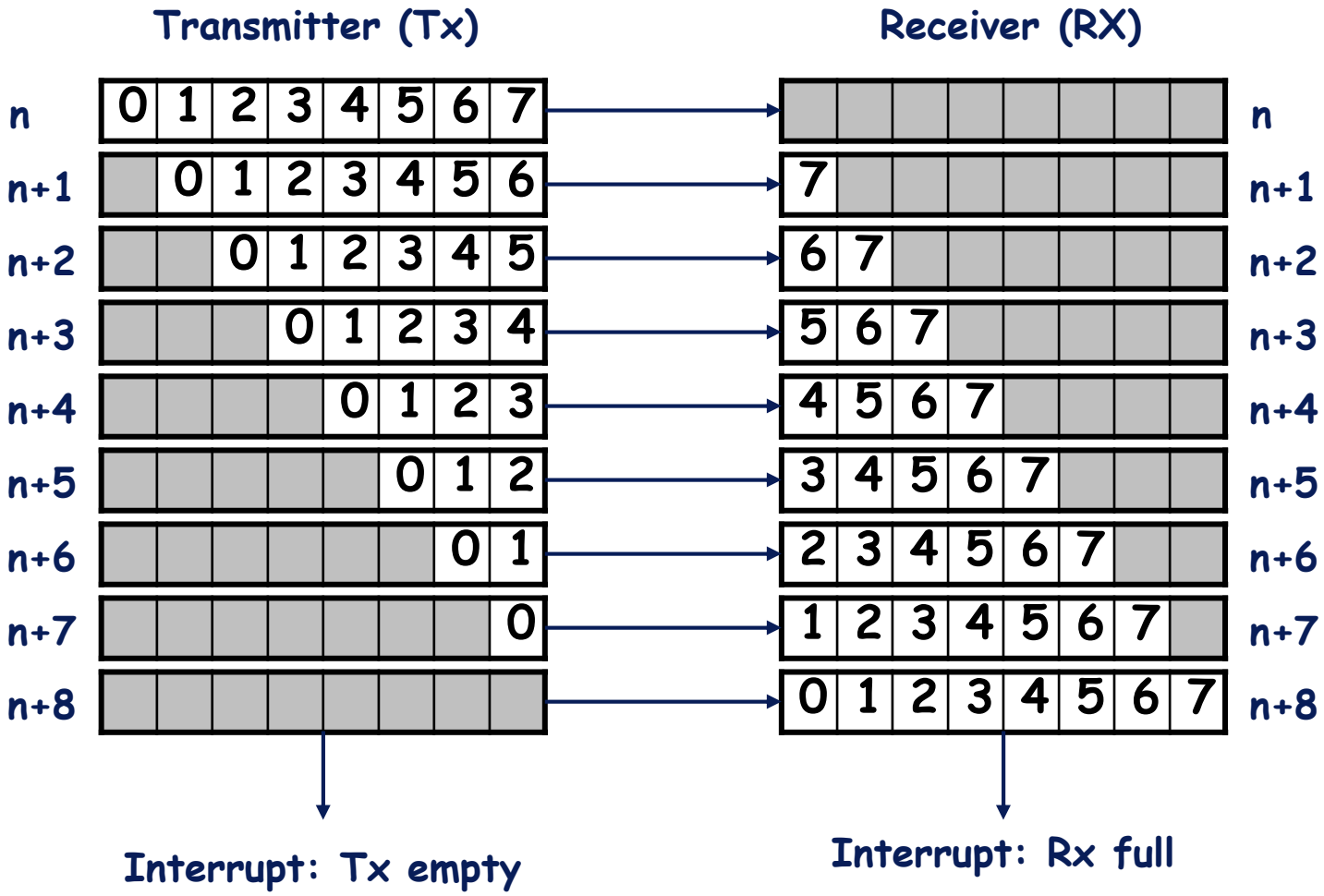
Uso di un buffer FIFO

- Utilizzo di un buffer FIFO per evitare la perdita di dati
  - in Rx:
    dato ricevuto prima che sia stato letto quello precedente
  - in Tx:
    dato inserito prima che sia stato trasmesso il precedente
- Dimensione del buffer FIFO:
  - Aumenta il throughput della porta
  - Riduce l'overhead della CPU (attesa della Tx o Rx)

Interfaccia Seriale Asincrona con Buffer FIFO



Es: trasmissione di un byte



Esempio di trasmissione su porta seriale

- Lato Tx:
  - Appena il dato è nel Tx-Register, viene trasmesso
  - Viene generato un interrupt per segnalare che il byte è stato trasmesso
- Lato Rx:
  - Il buffer riceve bit dopo bit il dato
  - Appena viene ricevuto un intero byte, viene trasferito nell' Rx-register
  - Viene generato un interrupt ed il byte viene letto
- In entrambi i casi
  - Viene messo a 1 un bit nel registro di stato (a fine trasmissione o ricezione)
  - Per poter proseguire, tale bit deve essere azzerato
    - Es. dalla routine di servizio

Controllo del flusso

- Necessario per impedire che Tx invii più dati di quelli che Rx può leggere

- Due metodi
  - Hardware:
    - il chip UART del Rx individua un potenziale intasamento e attiva una linea apposita per avvertire il Tx
    - Tx interrompe la trasmissione
    - Appena Rx può ricevere dati, la linea viene disattivata
  - Software:
    - Il Rx invia caratteri speciali di controllo (XON e XOFF) al Tx
    - XOFF interrompe una trasmissione
    - XON riprende la trasmissione

Standard EIA-RS232C

- Segnali:
  - In trasmissione, 1 logico=[-15V,-5V], 0 logico=[+5V,+15V]
  - In ricezione, 1 logico=[<-3V], 0 logico=[>+3V]
  - Il Tx deve vedere una capacita' <2.5nF, una resistenza <7KOhm
  - Il Tx deve poter supportare un corto circuito con corrente <0.5A
  - Massima lunghezza cavo:
    per 20Kb/s → 16m,
    per 9.6Kb/s → 75m,
    per 1.2Kb/s → 1000m

Standard EIA-RS232C

- I connettori più comuni sono due:
  - DB-25: connettore D type a 25 pin
  - DB-9 : connettore D type a 9 pin

DB-25	Sigla	Segnale	DB-9
1	GND	Chassis Ground	Not Used
2	TXD	Trasmit Data	3
3	RXD	Receive Data	2
4	RTS	Request To Send	7
5	CTS	Clear To Send	8
6	DSR	Data Set Ready	6
7	GND	Signal Ground	5
8	DCD	Data Carrier Detect	1
20	DTR	Data Terminal Ready	4
22	RI	Ring Indicator	9

Segnali (1)

- TXD (Trasmit Data):

  linea di trasmissione dei dati
  connessa alla linea RXD del Rx

- RXD (Receive Data):

  linea di ricezione del dato,
  connessa alla linea TXD del Tx

- RTS (Request to send):

  utilizzata in congiunzione con CTS
  segnala che Tx è pronto ad inviare dati

- CTS (Clear To Send):

  utilizzata in congiunzione con RTS

  segnala che Rx è pronto a ricevere i dati

Segnali (2)

- DSR (Data Set Ready):

  indica che il data-set (modem) è pronto a stabilire la comunicazione

- DTR (Data Terminal Ready):

  indica che il data-terminal (UART) è pronto a stabilire la comunicazione

- DCD (Data Carrier Detect):

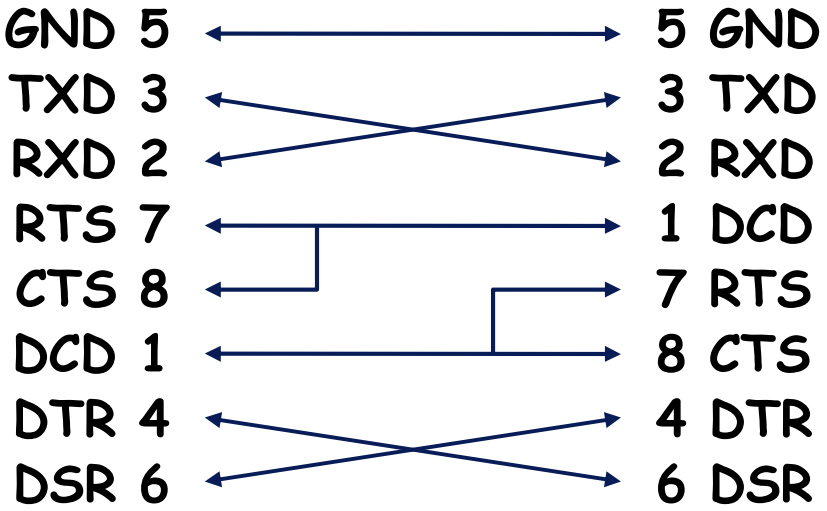
  viene usato quando è connesso un modem indica che il data-set (modem) ha rivelato la portante sulla linea telefonica

- RI (Ring Indicator):

  viene usato quando è connesso un modem e indica che ci sono degli squilli in arrivo sulla linea telefonica

Modem Cables

- Molte funzionalità di questi segnali derivano dal fatto che inizialmente furono pensati per connettere modem
- Per connessioni con altre periferiche, come stampanti, i segnali relativi ai modem devono essere "annullati" con l'uso di
  - cavi diretti oppure
  - cavi null-modem, es. per connettere due PC



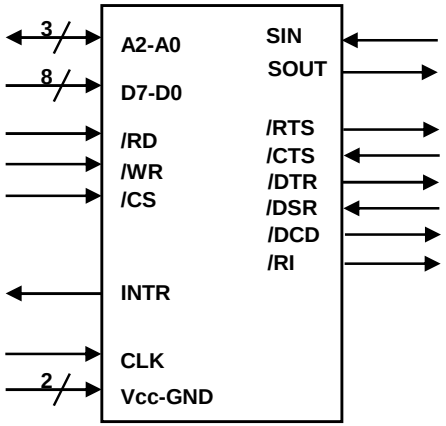
Connessione NULL-MODEM

La Porta Seriale nei Personal Computer

- Tipicamente implementata con un chip tipo 16550A
  - Bit-rate: compreso fra 50 e 115000
  - Formato trama:
    - 1 bit di start
    - da 5 a 8 bit di dati
    - con o senza parita'
    - 1, 1.5, 2 bit di stop
  - Riconoscimento dell'errore
    - Di parita' (numero di 0 o 1 non corrispondente)
    - Di framing (bit di stop non ricevuti correttamente)
    - Di overrun (arrivo di un nuovo dato prima di aver prelevato il precedente)
  - Emissione/ricezione di break
    - Persistenza in stato attivo per un tempo piu' lungo della trama
  - Gestione di 4 pin di ingresso e 4 di uscita, utili per rispettare lo standard EIA-RS232C
  - Buffer dati da 16 byte

UART Intel-16550A

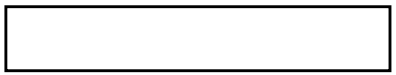
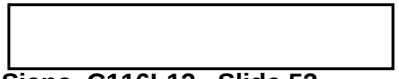
- Visione Elettrica del chip:



- A2-A0: indirizzamento
- D7-D0: scambio dati su 8 bit
- /RD, /WR, /CS: Read, Write, Chip-Select
- Vcc-GND: alimentazione
- CLK: clock locale per derivare le temporizzazioni
- SIN, SOUT: segnali Tx e Rx
- /RTS, /CTS, /DTR, /DSR, /DCD, /RI: segnali per l'interfacciamento secondo EIA-RS232C
- Altri pin possono essere presenti per usi generali e in dipendenza delle versioni specifiche del chip

UART Intel-16550A

- **Visione logica del chip (tutti registri a 8 bit):**

A2-A0	access	DLAB bit	mnemonic		
000	R	0	RBR		Receive Buffer Register Dove si trova il dato ricevuto
000	W	0	THR		Transmitter Holding Register Dove si mette il dato da trasmettere
001	RW	0	IER		Interrupt Enable Register
010	R	0	IIR		Interrupt Identification Register
010	W	0	FCR		FIFO Control Register
011	RW	-	LCR		Line Control Register
100	RW	-	MCR		Modem Control Register I bit 0 e 1 sono dei latch su /RTS e /DTS (in scrittura gli altri bit possono essere messi a 0)
101	RW	-	LSR		Line Status Register
110	RW	-	MSR		Modem Status Register
111	RW	-	SCR		Scratchpad Register Registro di appoggio da usare liberamente
001	RW	1	DLM		Divisor Latch MSB
000	RW	1	DLL		Divisor Latch LSB

Visione logica: DLM e DLL

- Avendo posto a 1 il bit di accesso ai Divisor Latches (DLAB=1) posso scrivere/leggere in DLM,DLL il valore della costante C che determina il bit-rate
  - DLM,DLL vengono interpretati come un intero senza segno a 16 bit
  - Detta FCK la frequenza del clock esterno e BR il bit-rate desiderato, si ha

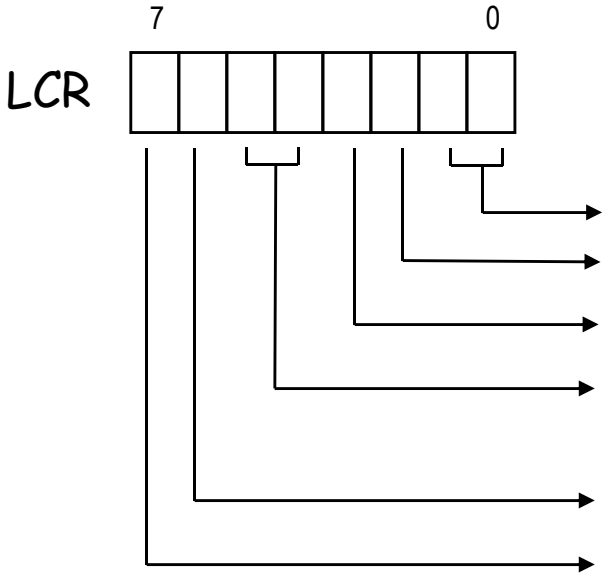
$$C = \frac{Fc}{16 \cdot BR}$$

Nota: nei PC Fc=1.8432MHz
Es. per ottenere un BR=9600
→C=12

BR	C
300	384
1200	96
4800	24
9600	12
115200	1

- DLAB non e' altro che il bit 7 di LCR
- I registri RBR, THR, IER, IIR, FCR si accedono ponendo DLAB=0
- Per i restanti registri il valore di DLAB e' ininfluente

Visione logica: LCR, Line Control Register



n. di bit del dato 00→5, 01→6, 10→7, 11→8

n. di bit di stop: 0→1, 1→1.5 se il dato ha 5 bit, 2 altrimenti

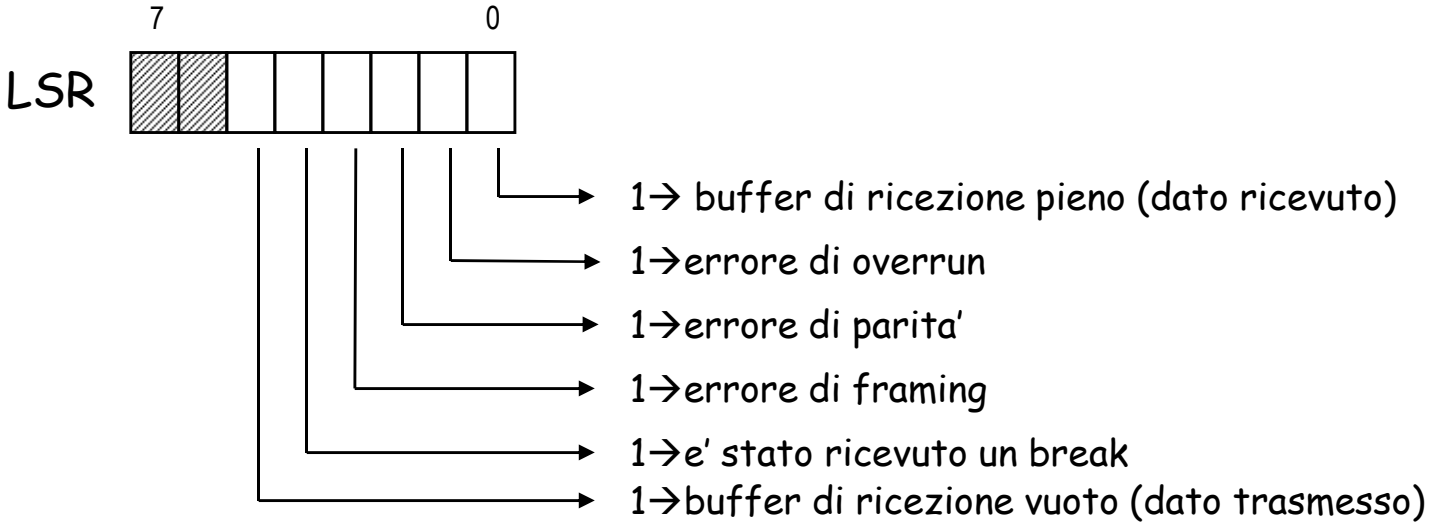
1→bit di parita' presente, 0→bit di parita' assente

00→parita'=#dispari di 1, 01→parita' #pari di 1,
10→parita'=#dispari a 0, 11→parita'=#pari di 0

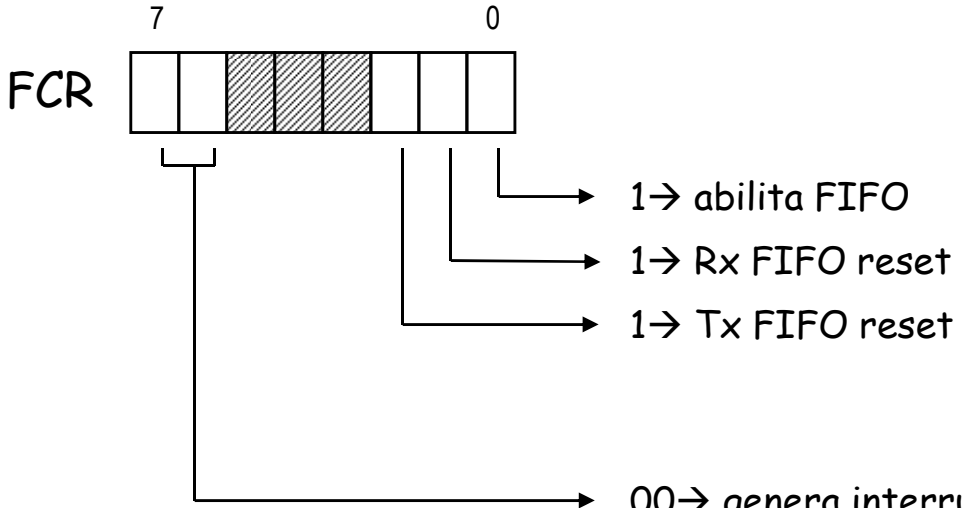
1→trasmetto un break (uscita a 0 logico)

Bit DLAB, se 1→posso leggere/scrivere la costante che detrmina il bit-rate

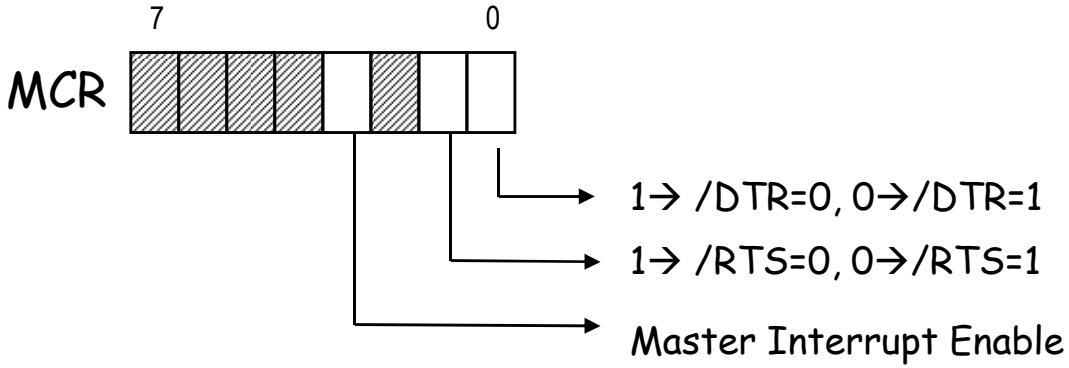
Visione logica: LSR, Line Status Register



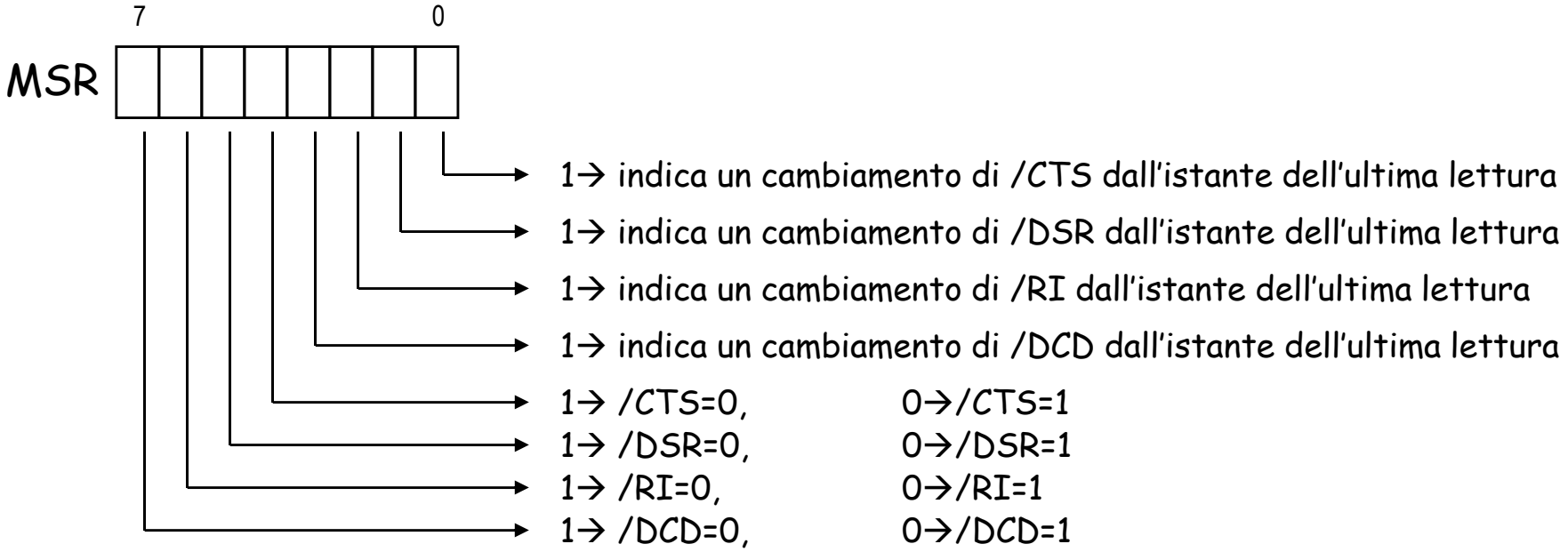
Visione logica: FCR, FIFO Control Register



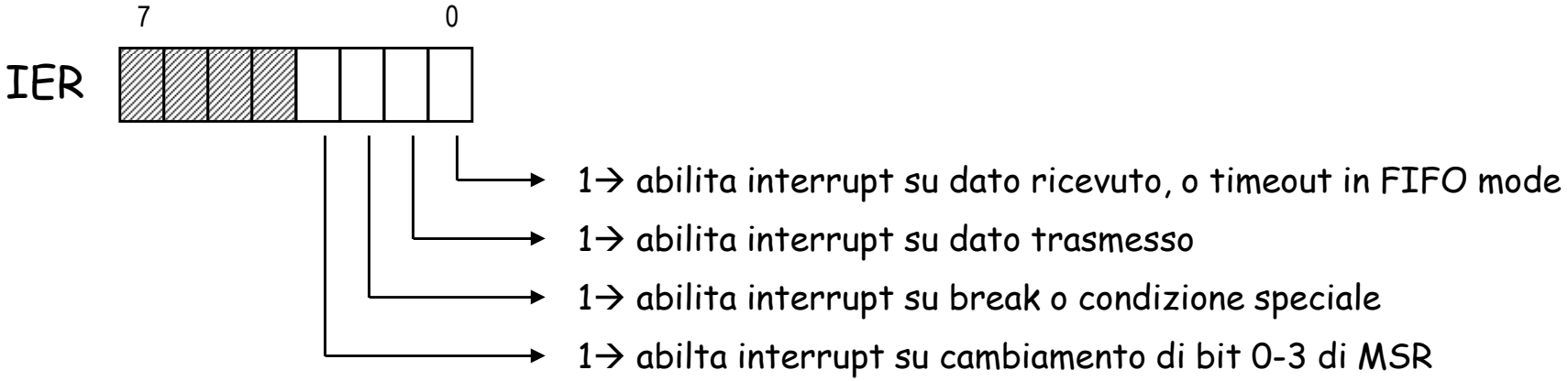
Visione logica: MCR, Modem Control Register



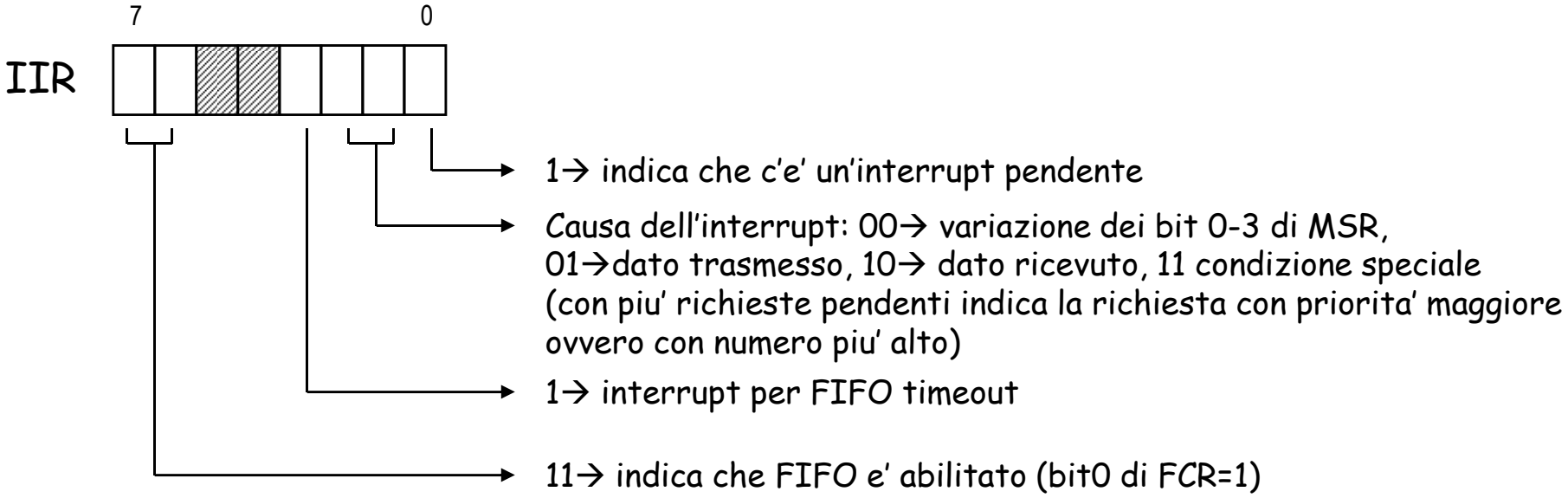
Visione logica: MSR, Modem Status Register



Visione logica: IER, Interrupt Enable Register



Visione logica: IIR, Interrupt Identification Register



- **Si rimanda ai data sheet per ulteriori dettagli e varianti**

Il chip 16550 nei Personal Computer

- PORTE DI I/O ASSEGNATE e INTERRUPT (Risorse)
  - 0x03F8-0x3FF porta seriale #1 (COM1 o ttyS0) con IRQ4
  - 0x02F8-0x2FF porta seriale #2 (COM2 o ttyS1) con IRQ3
  - 0x03E8-0x3EF porta seriale #3 (COM3 o ttyS2) con IRQ4
  - 0x02E8-0x2EF porta seriale #4 (COM4 o ttyS3) con IRQ3

- Nota: poiche' i pin SIN, SOUT, /RTS, /CTS, /DSR, /DCD, /RI sono a livelli TTL (0-5V), si rendono necessari dei chip 75188, 75189 per adattare i livelli di tensione a quelli richiesti dallo standard RS-232C.

Esempio di programma che utilizza la porta seriale (1)

```
#include <sys/types.h>
#include <sys/stat.h>   /* open */
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>      /* stdin, stdout, stderr */
#include <sys/ioctl.h>
#include <termios.h>
#include <stdlib.h>
#include <sys/time.h>   /* timeval */
#include <stdlib.h>
#include <string.h>     /* memset used by FD_ZERO */

#define DEFAULT_DEV "/dev/ttyS0"
#define DEFAULT_SPEED 115200

int translate_speed(int spd);
void fd_setup(int fd, int speed, int flow);

int main(int argc, char **argv)
{
  int serfd, i, c=0, error=0, len, do_echo = 0;
  char txdata[127], rxdata[256];

  if (argc > 2) {
    printf("wrong syntax ! ./sertest [-e]");
    exit(1);
  }

  if (argc == 2) {
    if ((strcmp("-e", argv[1]))==0) do_echo=1;
    else printf("wrong syntax ! ./sertest [-e]\n");
  }

  printf("Opening %s\n", DEFAULT_DEV);
  serfd = open(DEFAULT_DEV, O_RDWR);

  if (serfd < 0) exit(1);

  fd_setup(serfd, DEFAULT_SPEED, 1);

  for (i=0;i<127;i++) txdata[i]=i+64;
```

```
if (!do_echo)
  {
    while (!error){
      len = 0;
      i=0;
      printf("-%d- Sending data...", c++);
      write(serfd, txdata, 127);
      printf("done.\n");

      /* wait for 127 chars */
      while (len < 127)
      {
        len += read(serfd, rxdata+len, 127);
        printf("Got [%d/127]...\r", len);
        fflush(stdout);
      }

      /* now compare the received data */
      printf("Got response, comparing... ");

      if (memcmp(txdata, rxdata, 127) == 0)
        printf("OK.\n");
      else
        error = 1;
    }
  }
```

Esempio di programma che utilizza la porta seriale (2)

```
else
  {
    printf("Running as echo server.\n");
    while (!error) {
      len = 0;

      /* wait for 127 chars */
      while (len < 127)
      {
        len += read(serfd, rxdata+len, 127);
        printf("Got [%d/127]...\r", len);
        fflush(stdout);
      }

       /* now compare the received data */
      printf("-%d- Checking RX data... ", c++);

      if (memcmp(txdata, rxdata, 127) == 0)
      {
        printf("OK...echoing.\n");
        write(serfd, rxdata, 127);
      }
      else
      {
        printf("ERROR\n");
        error = 1;
      }
    }
  }

  printf("Not OK !\n");
  for (i=0;i<127;i++)
    printf("%c", txdata[i]);

  printf("\n");

  exit(1);
}
```

```
int translate_speed(int spd)
{
  int speed_c = 0;

  switch(spd) {
   case 9600:   speed_c = B9600; break;
   case 19200:  speed_c = B19200; break;
   case 38400:  speed_c = B38400; break;
   case 57600:  speed_c = B57600; break;
   case 115200: speed_c = B115200; break;
   case 230400: speed_c = B230400; break;
   case 460800: speed_c = B460800; break;
   default:     printf(
     "Bad baudrate %d is not valid.\n", spd); break;
  }
  return speed_c;
}

void fd_setup(int fd, int speed, int flow)
{
  struct termios t;

  if (fd < 0) { perror("fd_setup"); exit(1); }

  if (tcgetattr(fd, &t) < 0)
     { perror("tcgetattr"); exit(1); }

  cfmakeraw(&t);

  t.c_cflag &= ~CBAUD;
  t.c_cflag |= translate_speed(speed) | CS8 | CLOCAL;
  t.c_oflag = 0; /* turn off output processing */
  t.c_lflag = 0; /* no local modes */

  if (flow) t.c_cflag |= CRTSCTS;
  else t.c_cflag &= ~CRTSCTS;

  if (tcsetattr(fd, TCSANOW, &t) < 0)
     { perror("fd_setup : tcsetattr"); exit(1); }
  return;
}
```