Lezione 21

Sommario

- Trasmissione seriale sincrona e asincrona.
- Standard di livello fisico:
  - RS232;
  - RS422/RS485;
  - Loop di corrente 20 mA.
- Il bus I2C.
- Il bus CAN.
- Programmazione delle periferiche di trasmissione seriale.

Lezione 21

Materiale di riferimento

1. J. B. Peatman, "Design with Microcontrollers", McGraw - Hill, 1988.
2. A. Clements, "The principles of computer hardware", Oxford, 2000.
3. Datasheet del microcontrollore Philips LPC2129, disponibile sul sito web del corso in formato pdf.

Trasmissione seriale

Lo scambio di dati tra un mC o DSP e una unità periferica esterna può essere semplicemente realizzato attraverso un sistema di trasmissione seriale.

Questo permette la trasmissione e la ricezione dei dati (anche simultanee) un bit alla volta, richiedendo quindi un supporto fisico (linea o bus di trasmissione) di complessità minima (i.e. composto di pochi conduttori, di solito 3 o 4).

La trasmissione dei dati in parallelo, invece, richiede bus molto "larghi".

Trasmissione seriale

Nei sistemi di trasmissione seriale, a livello di collegamento, si distingue tra trasmissione sincrona e asincrona.

La trasmissione sincrona permette velocità più elevate, ma richiede una maggiore complessità del supporto fisico.

La trasmissione asincrona, invece, richiede un collegamento di complessità minima (al limite 2 fili), ma consente velocità minori.

La modalità asincrona è molto usata nelle comunicazioni dei microcontrollori, che spesso includono moduli appositi (UART).

Trasmissione seriale

La modalità di trasmissione sincrona è invece più comune nei DSP, dove permette il collegamento con unità periferiche esterne ad alta velocità.

Nei microcontrollori, la trasmissione seriale sincrona è usata nella gestione del bus di comunicazione I2C (Inter-Integrated Circuit), originariamente introdotto dalla Philips (a metà anni '80) e diventato ormai uno standard industriale di largo uso.

Molti microcontrollori incorporano unità periferiche per la gestione del bus I2C.

Trasmissione seriale asincrona

La trasmissione seriale asincrona avviene senza che i clock del trasmettitore (Tx) e del ricevitore (Rx) siano sincronizzati.

Il ricevitore deve perciò estrarre il sincronismo dalla stessa linea riservata al trasferimento dei dati.

La convenzione che si adotta per rendere possibile questa operazione è di far precedere alla trasmissione di un dato un bit di start e di far seguire uno o più bit di stop. Tra un dato e il successivo la linea può restare inattiva anche per tempi lunghi.

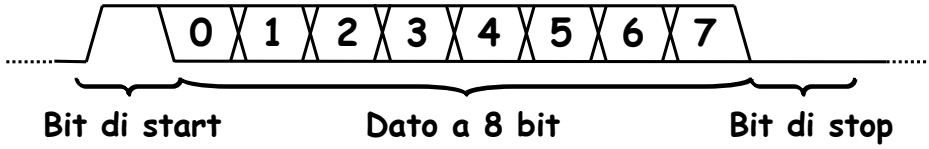
Trasmissione seriale asincrona

Questa modalità di trasmissione era stata inizialmente concepita per trasferire informazioni che sopraggiungono in modo "sporadico", con pause anche lunghe tra un dato e l'altro.

Essendo molto semplice, tende però ad essere usata anche per trasferire sequenze continue di dati, purchè non sia richiesta elevata velocità di trasferimento.

In pratica, la modalità asincrona è un vero e proprio standard per le trasmissioni da punto a punto a bassa velocità.

Trasmissione seriale asincrona



Il bit di start è sempre segnalato da una transizione della linea dallo stato inattivo (e.g. basso) a quello opposto. I bit di stop hanno invece la stessa polarità della linea inattiva (quindi bassa in questo esempio).

Il ricevitore, al momento della prima transizione basso – alto, comincia a campionare i diversi bit fino al bit di stop.

Trasmissione seriale asincrona

I parametri tipici della trasmissione seriale asincrona sono i seguenti:

bit di start:	1;
bit dei dati:	da 5 a 8;
bit di parità:	nessuno oppure 1;
bit di stop:	1, 1.5, 2;
velocità:	1200, 2400, 4800, 9600, 19200 bit al secondo (bps).

L'impostazione di questi parametri avviene in apposite periferiche denominate appunto Universal Asynchronous Receiver/Transmitter.

Trasmissione seriale asincrona

Il bit di parità è un bit aggiuntivo (opzionale) che può essere trasmesso subito dopo ciascun dato per introdurre un semplice sistema di controllo sugli eventuali errori di ricezione.

Se la somma dei bit trasmessi è pari, il bit è posto a 1, altrimenti a 0 (o viceversa).

Il ricevitore calcola il bit di parità sui bit ricevuti. Se questo risulta uguale a quello trasmesso considera la ricezione esente da errore (due errori sullo stesso dato sono assai meno probabili di uno) altrimenti segnala la situazione di errore.

Trasmissione seriale asincrona

Il fatto di usare alcuni tra i bit trasmessi per la sincronizzazione e la correzione di eventuali errori abbassa la velocità effettiva di trasferimento dei dati.

Nel caso migliore, per trasmettere 8 bit di dati è necessario in realtà trasferire 10 bit.

Ne consegue un sotto-utilizzo del canale del 20%. Tale fattore peggiora se si considerano dati a meno di 8 bit o se si aggiungono bit di stop e/o il bit di parità.

Questo è purtroppo un limite strutturale della trasmissione seriale asincrona.

Trasmissione seriale asincrona

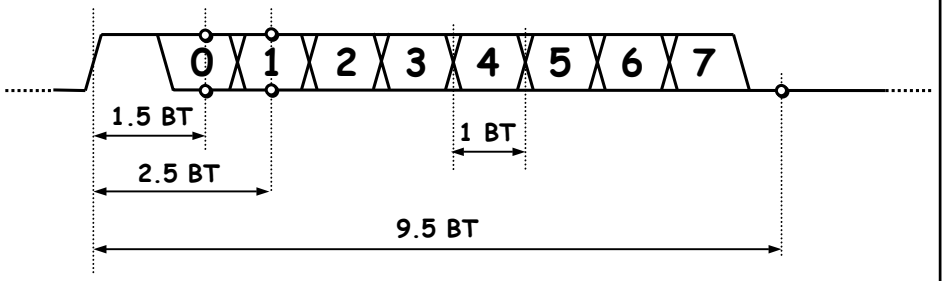
Affinché la trasmissione abbia successo, è necessario che il ricevitore campioni i bit dei dati a metà dell'intervallo di bit.

L'intervallo di bit ha una durata BT pari a 1/bps dove bps (detta anche baud rate) è la velocità di trasmissione impostata.

Nel caso di dati a 8 bit, gli istanti di campionamento ideali per il ricevitore sono quindi dopo 1.5 BT, 2.5 BT, …, 8.5 BT dalla transizione di start.

Bisogna ricordare però che i due clock (Tx e Rx) non sono sincronizzati.

Trasmissione seriale asincrona



Transizione di start

Se il sistema funziona correttamente, dopo 9.5 BT dalla transizione di start il ricevitore deve trovare la linea ad un livello basso, corrispondente al bit di stop. Altrimenti si è verificato un errore di sincronismo, che viene segnalato.

Trasmissione seriale asincrona

È interessante saper valutare il massimo scostamento relativo tra le frequenze di Tx e Rx che non produce un errore di sincronismo. Nell'ipotesi di rilevamento corretto della transizione di start, il massimo errore consentito è pari a ±0.5 BT, accumulato in un intervallo di 9.5 BT.

Ciò corrisponde ad un errore relativo tra le due frequenze pari a:

$$\frac{\Delta f}{f_{Tx}} = \frac{|f_{Tx}-f_{Rx}|}{f_{Tx}} = \frac{\pm 0.5}{9.5} = 0.053$$

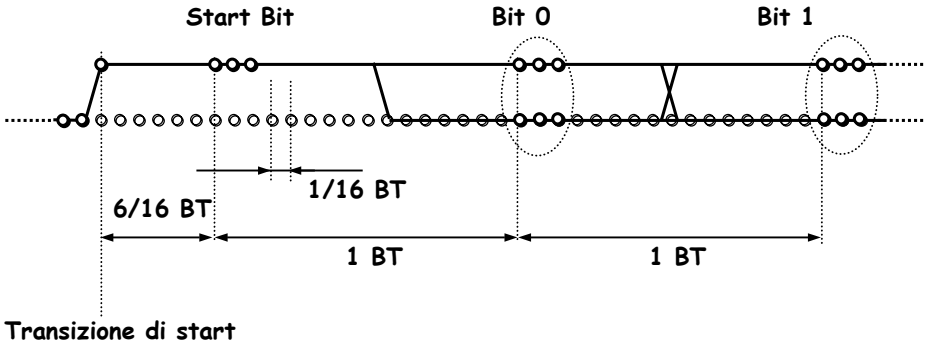
Trasmissione seriale asincrona

La frequenza di campionamento della linea da parte del ricevitore è di solito pari a 16 volte la frequenza di trasmissione.

La procedura di sincronizzazione si attiva al primo campione "alto" successivo ad una serie di campioni "bassi". Dopo 6 periodi di campionamento vengono acquisiti 3 campioni in successione, per garantire il corretto rilevamento della transizione di start.

Da questo istante in poi, il ricevitore legge sempre blocchi di 3 campioni, distanti l'uno dall'altro 16 periodi di campionamento.

Trasmissione seriale asincrona



Il valore logico assegnato al bit corrente è deciso con logica di maggioranza all'interno del cluster dei 3 campioni acquisiti. Questo abbassa la probabilità di errore.

Trasmissione seriale asincrona

Un errore di sincronismo può però insorgere anche se le frequenze di Tx e Rx differiscono di meno del 5.3%, a causa dell'incertezza nel rilevamento della transizione di start. Questa può essere rilevata con un ritardo massimo di 1/16 BT, quindi la condizione di errore è che in 9.5 BT si sia accumulata una differenza sui tempi di ±(0.5 – 1/16) BT. Lo stesso calcolo eseguito in precedenza richiede allora che sia:

$$\frac{\Delta f}{f_{Tx}} = \frac{|f_{Tx}-f_{Rx}|}{f_{Tx}} = \frac{\pm 0.4375}{9.5} = 0.046$$

Trasmissione seriale sincrona

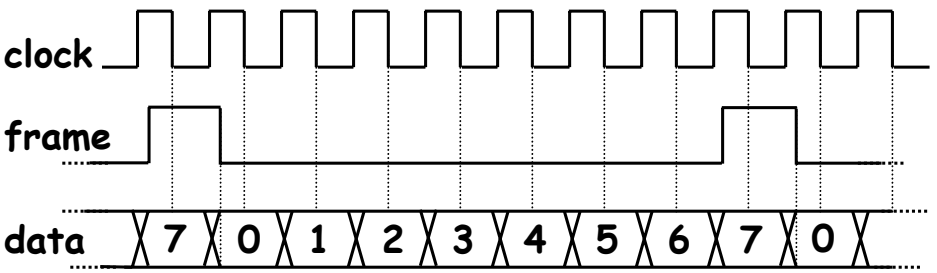
La modalità sincrona di trasmissione seriale è basata sull'inerente sincronizzazione dei clock del trasmettitore e del ricevitore.

Per ottenere questo effetto il trasmettitore funge da master e invia al ricevitore, su una linea dedicata, anche il suo clock.

Inoltre, sono spesso presenti linee di controllo aggiuntive che permettono di trasmettere segnali di sincronismo a livello di parola (frame sync).

In questo modo, le linee di trasmissione operano sui dati per il 100% del tempo.

Trasmissione seriale sincrona



Esempio di trasmissione seriale sicrona a 8 bit. Il fronte di discesa del clock indica al ricevitore quando il dato è stabile e può essere campionato. Il segnale di frame indica al ricevitore l'inizio di una nuova parola.

Trasmissione seriale sincrona

Le periferiche per la trasmissione seriale sincrona (USRT) presenti a bordo di mC e DSP permettono in realtà di configurare molti parametri della trasmissione.

Sono ad esempio configurabili: la polarità del clock (associazione tra i fronti del clock e il campionamento dei dati), la polarità dei dati (associazione tra tensioni sulla linea e livelli logici), l'ordine di trasmissione dei bit (MSB first o LSB first), la dimensione in bit dei dati (8 oppure 16 o altro).

Trasmissione seriale sincrona

Anche il segnale di sincronismo di frame può essere variamente configurato. Ad esempio può avere durata pari a un BT, oppure durata pari ad una intera parola.

La maggior parte delle periferiche per la trasmissione seriale sincrona supportano però solo un segnale di sincronismo di frame di durata BT.

Inoltre è talora possibile trasmettere il sicronismo di frame ogni 2 parole anziché dopo ogni parola (e.g. nel trasferimento dei dati relativi a due canali di acquisizione).

Trasmissione seriale sincrona

Infine, in molte USRT è possibile per l'utente definire la frequenza del clock di trasmissione in modo molto flessibile, essendo a disposizione un divisore di frequenza programmabile con pre-scaler.

In altri casi, le possibili frequenze di trasmissione sono predefinite e l'utente può soltanto scegliere quale usare.

In alcuni casi è possibile che la generazione del clock sia a cura di un circuito esterno, il cui segnale viene usato dai due dispositivi come base dei tempi della comunicazione.

Trasmissione seriale

La trasmissione seriale, sia di tipo asincrono che di tipo sincrono, avviene spesso secondo uno degli standard di livello fisico seguenti:

1. RS232 in una delle numerose versioni;
2. RS422/v11 o v12;
3. Loop di corrente a 20 mA;

che definiscono, tra le altre cose, il numero di conduttori richiesti per la realizzazione della comunicazione, i livelli di tensione e le modalità di pilotaggio delle linee.

In altri casi (e.g. nei sistemi chiusi) non si usa invece alcuno standard.

Standard RS232

Si tratta di uno standard, introdotto nel 1962 e successivamente modificato molte volte, pensato per definire le modalità di comunicazione tra un DTE (Data Terminal Equipment, cioè un computer) e un modem.

E' stato poi usato in molti altri tipi di collegamento punto a punto.

Lo standard definisce 54 linee, stabilendone le funzioni. Il connettore standard RS232 ha però 25 poli, o, in alcuni casi, soltanto 9.

Nelle comunicazioni asincrone sono necessarie solo 3 linee (trasmissione, ricezione, massa).

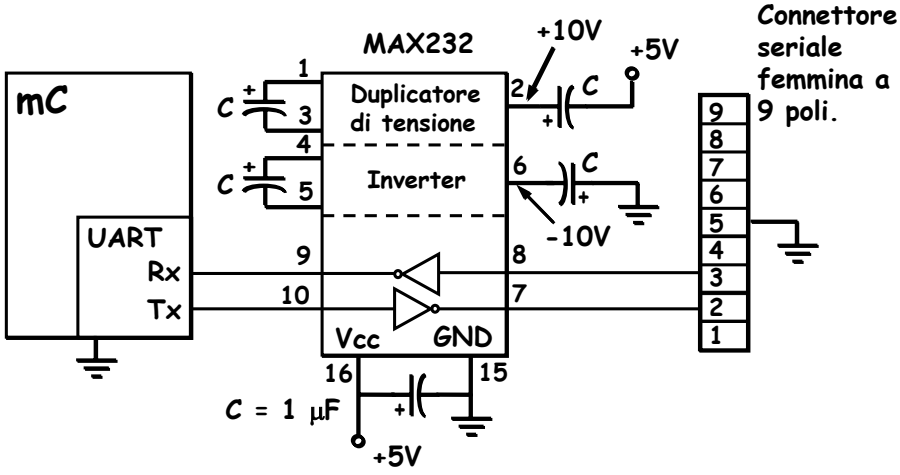
Standard RS232

I livelli di tensione stabiliti dallo standard sono due: +12V, -12V. La tensione di –12 V è associata al livello logico 1, quella di +12V al livello logico 0 (logica negata).

Molti dispositivi lavorano però anche con tensioni minori, e.g. di +10V e -10V, conservando comunque la stessa logica.

La comunicazione secondo questo standard può quindi richiedere l'uso di dispositivi di interfacciamento (driver) capaci di pilotare le linee alle tensioni prescritte.

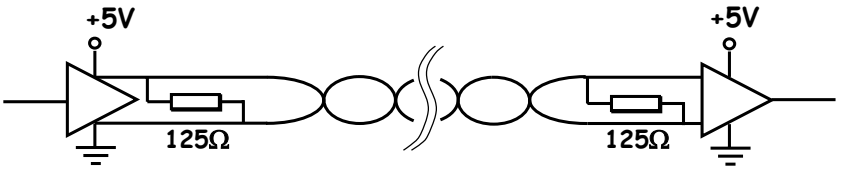
Standard RS232



Esempio di collegamento seriale asincrono secondo lo standard RS232 tra un PC e un mC.

Standard RS422

Lo standard RS422 prevede la trasmissione dei dati in modo differenziale tra due conduttori che sono di norma intrecciati per minimizzare la possibilità di interferenza.



La maggiore immunità al rumore consente di supportare velocità di trasmissione anche molto elevate (10 Mbps su distanze inferiori a 15 m)

Standard RS485

Dal punto di vista elettrico lo standard RS485 è del tutto simile al precedente RS422, prevedendo la trasmissione dei dati in modo differenziale tra due conduttori intrecciati.

La principale differenza consiste nel fatto che lo standard RS485 prevede la possibilità di realizzare una linea di trasmissione a molti trasmettitori. Questo impone la presenza nei driver RS485 di un comando di abilitazione per i trasmettitori, che possono essere posti in uno stato ad alta impedenza.

Standard RS422/RS485

La trasmissione differenziale è molto più robusta ai disturbi elettromagnetici rispetto a quella single-ended. Per questo motivo, gli standard RS422 e RS485 sono molto utilizzati in ambito industriale.

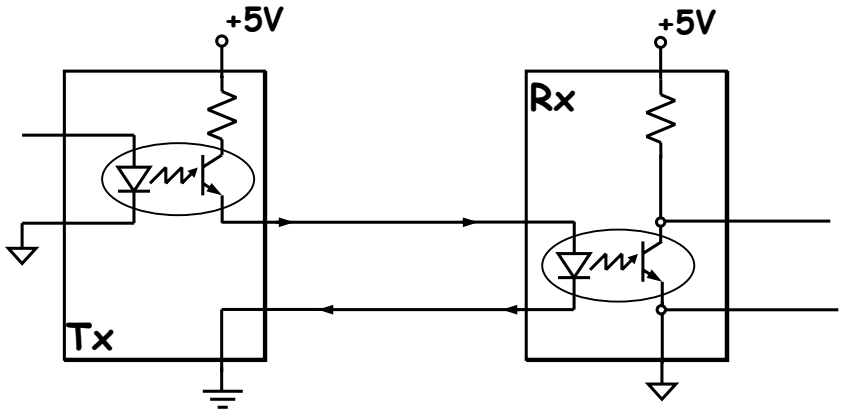
Gli accoppiamenti capacitivi sono resi ininfluenti dalla insensibilità del ricevitore al modo comune della tensione presente sulle linee dati. Gli accoppiamenti induttivi sono invece minimizzati dalla disposizione intrecciata dei conduttori (doppino). Inoltre, spesso sono impiegati conduttori schermati.

Standard Current Loop 20 mA

Il collegamento in loop di corrente associa il passaggio di una corrente (di 20 mA) al livello logico 1, l'assenza di corrente allo 0. Si tratta di un metodo di trasmissione molto usato a livello industriale in quanto molto poco sensibile al rumore elettromagnetico ambientale (massima immunità).

Per la sua realizzazione si impiegano spesso driver opto-isolati, che permettono quindi di mantenere galvanicamente separati il trasmettitore ed il ricevitore, fino a differenze di potenziale di migliaia di Volt.

Standard Current Loop 20 mA



Esempio di collegamento in loop di corrente a 20 mA, con optoisolatori.

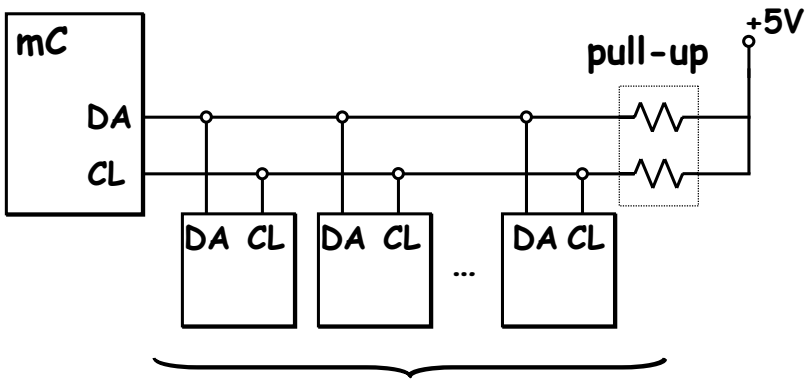
Il bus I2C

Il protocollo di comunicazione I2C è uno standard, industrialmente molto diffuso, per la connessione seriale sincrona dei circuiti integrati ad esso predisposti.

Esso permette il collegamento di molti dispositivi ad un solo master, richiedendo la disponibilità di due linee di trasmissione, una per i dati e l'altra per il segnale di clock.

Molti tra i mC più recenti includono un'apposita unità periferica per la gestione di questo tipo di comunicazione, che ne facilita l'espansione con periferiche esterne.

Il bus I2C



Configurazione tipica del bus di comunicazione I2C.

Il bus I2C

Nello standard I2C sono disponibili due modi di funzionamento: normale e fast.

Il primo modo (normale) consente di operare con velocità comprese tra 0 e 100 kbps.

Il modo fast consente di estendere la velocità di trasferimento fino a 400 kbps.

Può accadere che al bus I2C vengano collegati dispositivi che supportano il modo fast insieme ad altri che invece non lo supportano.

In questo caso il bus dovrà essere usato nel modo normale, limitando la velocità a 100 kbps.

Il bus I2C

Il mC, che gestisce il bus (i.e. funge da master), invia sulla linea clock il segnale di sincronismo per tutte le unità collegate. Il trasferimento dei dati avviene 9 bit per volta.

I primi 8 bit trasferiti sul bus dei dati sono sempre a cura dell'unità che in quel momento sta trasmettendo, mentre il nono è a cura dell'unità cui il dato trasmesso è destinato.

Se il ricevitore riconosce il dato, deve portare la linea dati al valore basso nel BT riservato al nono bit. Altrimenti la linea resta al livello alto.

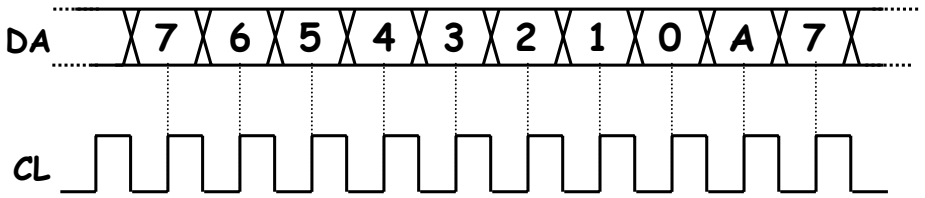
Il bus I2C

Ad esempio, il mC può indirizzare una periferica trasmettendo sul bus dati l'indirizzo della stessa. Quando la periferica vede transitare il proprio indirizzo, risponde abbassando la linea dati nel nono BT. Le altre periferiche restano invece inerti.

Il mC, leggendo un valore logico basso per il nono bit, apprende che la periferica indirizzata è pronta a ricevere i dati e comincia a inviarli.

Allo stesso modo, una periferica può inviare i dati al mC agendo da trasmettitore.

Il bus I2C



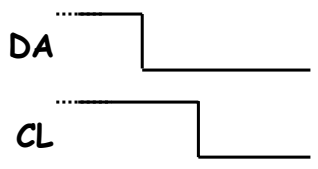
Modalità di trasmissione di un dato sul bus I2C. La scrittura del bit inizia sul fronte di discesa del clock. Il bit è mantenuto stabile fino al fronte di discesa seguente.

La scrittura del byte inizia sempre con il bit più significativo (MSB). L'ultimo BT (il nono) è riservato all'acknowledgement (A).

Il bus I2C

In realtà la comunicazione tipica tra il mC e le periferiche coinvolge più di un byte. E' quindi necessario un protocollo per la gestione della comunicazione.

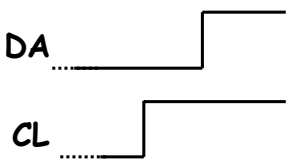
Il bus I2C è pensato per trasferire "messaggi". Ogni messaggio si apre con la generazione, da parte del mC, che controlla la trasmissione, di un segnale di start.



Condizione di start: la linea dati DA viene abbassata prima della linea del clock.

Il bus I2C

Ogni messaggio si chiude poi con la generazione, da parte del mC, di un segnale di stop.



Condizione di stop: la linea dati DA viene alzata dopo la linea del clock.

Le condizioni di start e stop possono essere distinte da un qualunque altro dato perché la linea dati DA cambia stato durante la fase positiva del clock, al contrario di quanto accade nella trasmissione di un bit.

Il bus I2C

I messaggi sono quindi costituiti da sequenze di byte incluse tra una condizione di start e una di stop. Il primo byte del messaggio che il mC invia è sempre rappresentato da un indirizzo a 7 bit. Questo deve individuare univocamente una periferica tra quelle interfacciate al bus. L'ottavo bit del primo byte è usato dal mC per predisporre un trasferimento dati verso la periferica (in tal caso è basso) oppure per prelevare dati dalla periferica (in tal caso è alto).

Si tratta di un bit di R/$\overline{W}$.

Periferiche I2C

In commercio sono disponibili molti tipi di dispositivi predisposti per il collegamento ad un bus I2C. Si trovano espansioni per le porte di uscita, ma anche sensori (di temperatura, pressione, …) o ancora convertitori DA. Tutti permettono all'utente di definirne l'indirizzo, fissando il livello logico su alcuni piedini a questo dedicati. Di norma, non tutti i 7 bit dell'indirizzo sono configurabili dall'utente: è comune che siano variabili solo i 3 bit più bassi dell'indirizzo. Raramente, infatti, sono richieste più di 8 periferiche dello stesso tipo.

Messaggi sul bus I2C

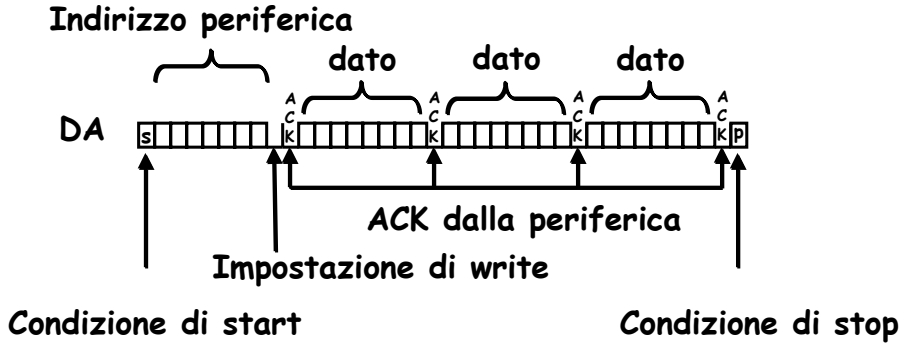
I byte successivi al primo hanno funzioni diverse a seconda del tipo di periferica coinvolta.

Nel caso di periferiche complesse, il mC spesso invia più di un byte con funzioni di controllo e/o configurazione.

Solo successivamente avviene la trasmissione dei dati veri e propri.

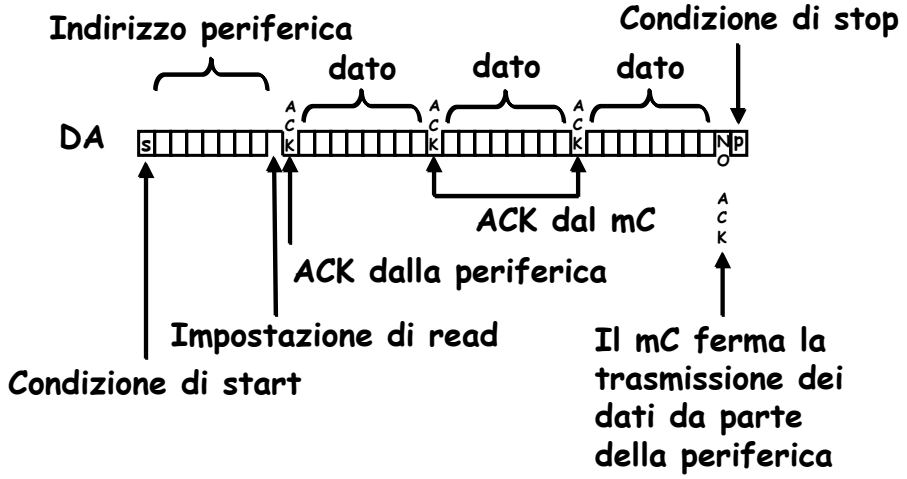
Nel caso di lettura di dati da una periferica il mC può mettere fine alla trasmissione da parte della periferica stessa, semplicemente non dando il segnale di acknowledgement.

Messaggi sul bus I2C



Trasmissione di dati dal mC ad una periferica. Il mC avvia e conclude la trasmissione. Il primo byte indirizza la periferica e imposta la condizione di write.

Messaggi sul bus I2C



Trasmissione di dati da una periferica al mC.

Il bus CAN

Il bus CAN (Controller Area Network) è uno standard di comunicazione seriale asincrona pensato per la trasmissione dati in ambiente industriale (applicazioni real-time).

Sviluppato dalla Bosch (1986) su richiesta della Mercedes, il bus CAN si caratterizza per l'elevata velocità di trasmissione, i bassi costi e i numerosi meccanismi di rilevazione e correzione degli errori.

E' diventato rapidamente uno degli strumenti più usati nell'ambito dell'automazione industriale e del controllo di processo.

Il bus CAN

A livello fisico, il bus CAN praticamente si uniforma allo standard RS422, prevedendo l'uso di doppino intrecciato con impedenza controllata di 120 Ω. La velocità di trasmissione è variabile in funzione della lunghezza del collegamento.

Lunghezza del bus (metri)	Max bit rate (bit/s)
40	1 Mbit/s
100	500 Kbit/s
200	250 Kbit/sec
500	125 Kbit/s
6000	10 Kbit/s

Il bus CAN

A livello di collegamento, il bus CAN si caratterizza per la particolare struttura dei messaggi, il modo di gestire il mezzo di trasmissione e i meccanismi di rilevamento e correzione degli errori.

I messaggi CAN sono privi di indirizzi del mittente e del destinatario. I messaggi sono invece "etichettati" in base al loro contenuto.

Le varie unità interfacciate al bus recepiscono tutti e soli i messaggi il cui contenuto sia rilevante per la funzione che svolgono.

Il bus CAN

Prerogativa del CAN bus è che l'identificatore del contenuto di un messaggio sia unico per tutta la rete, in modo da fissare una corrispondenza certa tra etichette e contenuti dei messaggi.

Non lavorando su indirizzi, il bus CAN risulta estremamente flessibile, permettendo l'aggiunta di unità riceventi senza alcuna modifica dell'hardware o del software di gestione.

Un'altra conseguenza dell'approccio illustrato è che non c'è alcun gestore del bus (multicast).

Il bus CAN

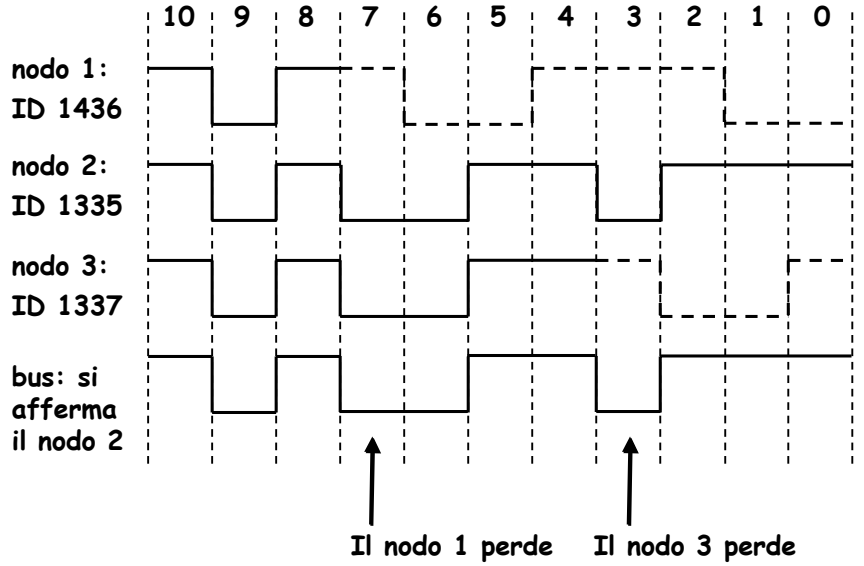
Quando due o più unità si contendono il bus per la trasmissione di un messaggio, l'assegnazione avviene in modo automatico sulla base della priorità assegnata implicitamente ad ogni identificatore.

I valori più bassi prevalgono su quelli più alti in base ad una logica di tipo wired-and.

Senza nessun intervento esterno, si afferma sul bus il messaggio il cui identificatore ha il codice numericamente più basso.

Questo arresta la trasmissione da parte di tutte le unità "perdenti".

Il bus CAN



Il bus CAN

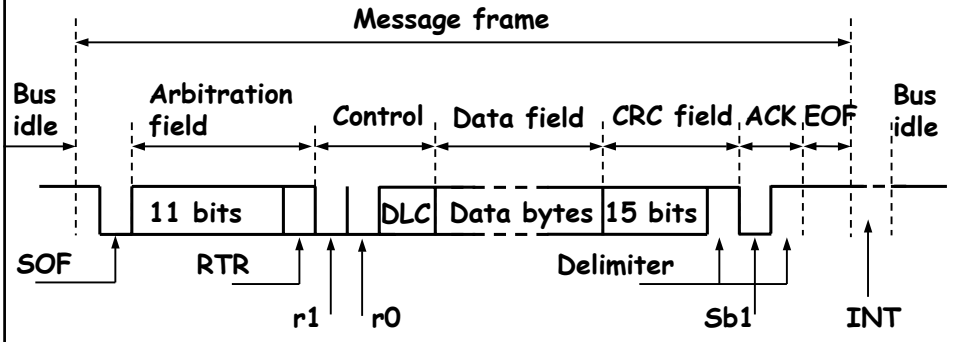
Il bus CAN prevede 4 tipi di messaggi:

1. Data Frame
2. Remote Frame
3. Error Frame
4. Overload Frame

I primi due possono avere formati diversi a seconda che si abbia a che fare con CAN standard oppure extended.

Nel secondo caso l'identificatore è formato da 29 bit, anziché da 11.

Messaggi sul bus CAN



Formato di un messaggio nello standard CAN 2.0A. Lo start of frame è un bit dominante, i.e. uno zero, che segue uno stato di idle del bus, nel quale questo si trova nello stato alto.

Messaggi sul bus CAN

Il messaggio si apre con un campo di 11 bit che rappresenta l'ID del messaggio, seguito da un bit (Remote Transmission Request) per qualificare il messaggio come data frame (se basso: invio di dati) oppure come remote frame (se alto: richiesta di dati).

Il campo seguente, di 6 bit, è un campo di controllo. I primi due bit sono riservati (i.e. sempre bassi), i quattro seguenti servono per indicare il numero di byte che verranno trasmessi (solo nel caso di un data frame: da 0 a 8 byte).

Messaggi sul bus CAN

Il campo successivo (CRC) è riservato a 15 bit di cyclic redundancy check code (codice a ridondanza ciclica per la correzione degli errori).

Dopo un bit di delimitazione, segue il campo di acknowledgement di due bit. Tutte le periferiche, che hanno correttamente ricevuto il messaggio, portano il primo bit a 0. Il secondo bit è ancora un delimitatore.

Il campo EOF, infine, consiste di 7 bit di tipo recessive (i.e. alti).

Messaggi sul bus CAN

Il bus è considerato libero solo dopo un intervallo di almeno tre bit "alti". In questo intervallo di tempo, le diverse unità possono inviare sul bus un overload frame.

L'overload frame viene inviato da un nodo quando questo non è in grado di ricevere altri dati, perché impegnato nelle elaborazioni dei precedenti. Consiste in un bit basso, che viene trasmesso subito dopo l'EOF.

Le altre unità, rilevando questo bit, rispondono a loro volta con un bit basso sul bus. Ciò impedisce l'inizio di un nuovo frame.

Errori sul bus CAN

Il bus CAN è stato studiato in modo da raggiungere un elevato grado di affidabilità, i.e. in modo da riconoscere e isolare in modo autonomo gli errori di trasmissione.

Il rilevamento di un errore da parte un nodo del bus comporta l'immediata trasmissione di un error frame.

Il messaggio "incriminato" viene ignorato da tutti i nodi e il trasmettitore provvede a inviarlo di nuovo, sempre competendo con eventuali altri messaggi.

Errori sul bus CAN

Ogni unità, o nodo, del bus rileva un errore quando viene violata una delle regole di trasmissione, sia a livello di bit che di frame.

La prima regola è detta bit stuffing.

Ogni trasmettitore non può trasmettere più di 5 bit dello stesso tipo. Dopo 5 bit uguali ne viene inserito sempre uno di polarità opposta.

Se un ricevitore trova una sequenza di più di 5 bit uguali è certo che si è verificato un errore.

Errori sul bus CAN

Ogni trasmettitore rileva sempre i bit che sta trasmettendo, in modo da poter riscontrare subito un errore, naturalmente solo se non è in corso la trasmissione dell'arbitration field o dell'acknowledgement.

In caso trovi un bit diverso da quello che intendeva trasmettere, il trasmettitore segnala una condizione di bit error.

Il checksum error viene generato da un ricevitore che, calcolando il CRC sui dati ricevuti, ottenga un valore diverso da quello trasmesso.

Errori sul bus CAN

Si genera invece un frame error quando le dimensioni dei campi di un frame vengono violate da un messaggio.

Infine, si genera un acknowledgement error quando nessun nodo segnala il recepimento del messaggio trasmesso.

Il tipico error frame è composto da 6 bit dello stesso segno, il che comporta la trasmissione di un error frame anche da parte di tutte le altre unità sul bus (bit stuffing).

Ogni unità sul bus conteggia gli errori rilevati, sia in trasmissione che in ricezione.

Errori sul bus CAN

In base al conteggio degli errori rilevati, una unità è in grado di stabilire autonomamente se sta funzionando correttamente (entrambi i conteggi sono minori di 127, e vengono ridotti di 1 ad ogni messaggio andato a buon fine) oppure se riscontra qualche problema (almeno uno dei contatori ha superato il valore 127).

Qualora uno dei due contatori superi addirittura il valore di 255, l'unità si distacca dal bus, permettendo alle altre di continuare a operare. Questo meccanismo consente al bus di non risentire del guasto di una unità.

Periferiche di trasmissione seriale

Molti mC e anche alcuni DSP incorporano tutto l'hardware necessario alla gestione di una trasmissione seriale, in modalità asincrona (UART) o, talvolta, sincrona (USRT, SPI).

Spesso sono disponibili periferiche per la gestione del bus I2C e, più recentemente, anche le periferiche per interfacciare il processore con un bus CAN, integrando così le funzioni di comunicazione e controllo di processo.

Periferiche di trasmissione seriale

La programmazione delle periferiche per la trasmissione seriale è di solito relativamente semplice, consistendo nella configurazione dei parametri della trasmissione (bit rate, bit di parità, lunghezza dei dati etc.).

La programmazione delle periferiche di controllo di un bus (I2C, CAN) è invece più complessa, richiedendo sia la definizione dei parametri della trasmissione che del protocollo di collegamento.

Esempio di uso di una UART

Con il mC LP2129, sviluppiamo il codice per la trasmissione seriale di una stringa di caratteri. Supponiamo di voler trasmettere a 9600 baud (@ pclk = 12 MHz) usando la periferica UART1 del mC.

Il primo problema consiste, come al solito, nel configurare correttamente la periferica. Dal manuale del mC, apprendiamo che questo richiede l'impostazione di almeno due registri ovvero: U1LCR per la configurazione del protocollo di trasmissione e U1DLL per la definizione del bit rate.

Esempio di uso di una UART

Impostando la seguente configurazione:

U1LCR = 0x83

si programma la trasmissione di 8 bit per volta, senza bit di parità e con un solo stop bit. Porre a 1 il bit 7 permette poi di accedere al registro di configurazione del baud rate. Scrivendo

U1DLL = 0x4E

si ottiene la generazione di un baud rate pari a 9615 bit/s. Il numero 0x4E rappresenta la metà del tempo allocato per la trasmissione di un bit, in termini di periodi di clock (pclk).

Esempio di uso di una UART

Altri registri di interesse sono U1THR che è il buffer di trasmissione e U1LSR che è il registro di stato della trasmissione.

Il primo, che può essere scritto solo quando il bit 7 del registro U1LCR è pari a 0, contiene il byte da trasmettere. In realtà, la UART gestisce una struttura dati di tipo FIFO di cui U1THR è il top byte. Questo permette di riversare sulla UART più di un byte per volta e impostare la trasmissione di tutto l'insieme dei byte. Nel nostro caso, però, l'uso della FIFO non è necessario.

Esempio di uso di una UART

Possiamo infatti gestire la trasmissione in "polling", usando il registro di stato U1LSR per vedere quando il buffer di trasmissione è libero (perché il byte precedente è stato spedito) e quindi un nuovo byte può essere scritto.

In questo modo evitiamo di gestire la trasmissione con le interruzioni, semplificando il codice.

Non avendo altri compiti da svolgere, ciò non costituisce per noi una penalizzazione. In generale, l'uso delle interruzioni è preferibile.

Esempio di uso di una UART

```
            LDR   r0,=stringa     /* r0 punta primo byte della stringa          */
            LDR   r1,=eos         /* r1 punta ultimo byte della stringa         */
Trasmetti:  LDR   r2,[r0]         /* r2 contiene 4 byte della stringa da spedire */
            AND   r2,r2,#0xFF     /* seleziono il byte corrente                 */
            MOV   r14,PC          /* preparo rientro da subroutine              */
            B     Putchar         /* salto alla subroutine di scrittura         */
            ADD   r0,r0,#0x01     /* incremento puntatore carattere             */
            CMP   r0,r1           /* verifico se eos                            */
            BLE   Trasmetti       /* se no continuo                             */
Fine:       B     Fine            /* stringa trasmessa: mi fermo                */
Putchar:    LDR   r3,=U1THR       /* indirizzo buffer                           */
            LDR   r4,=U1LSR       /* indirizzo stauts register                  */
Attesa:     LDR   r5,[r4]         /* leggo status register                      */
            MOV   r6,#0x20        /* maschera per occupato (bit 5 alto)         */
            CMP   r5,r6           /* verifico se occupato                       */
            BEQ   Attesa          /* se sì aspetto                              */
            STR   r2,[r3]         /* scrivo byte da spedire                     */
            MOV   PC, r14         /* ritorno da subroutine                      */
```