Lo standard I^2C

I^2C (*Inter Integrated Circuit)* è uno standard per lo scambio dati con una interconnessione seriale sincrona, studiato da **Philips** (attualmente **NXP**) che ne detiene i diritti. Analogo a I^2C è SMBus (System Management Bus) di Intel, con struttura simile, ma con qualche differenza sulle soglie dei livelli logici.

Come dice il nome, si tratta di una connessione tra circuiti integrati, prevista per collegare vari dispositivi presenti su circuito stampato o all'interno di una apparecchiatura. Lo standard prevede l'esistenza di più dispositivi, sia Master (**multi-Master**) che Slave, collegati tra di loro utilizzando solamente due conduttori, uno per i dati (**SDA**,**S**erial **DA**ta) ed uno per il clock (**SCL**,**S**erial **CL**ock); ovviamente anche la massa deve essere in comune.

Siccome sia Master che Slave possono trasmettere e ricevere dati, la **linea dati è bi-direzionale**. E lo è **anche il clock**, e sarà il Master attivo a comandarlo. Secondo la definizione originale del protocollo sono identificati le seguenti voci:

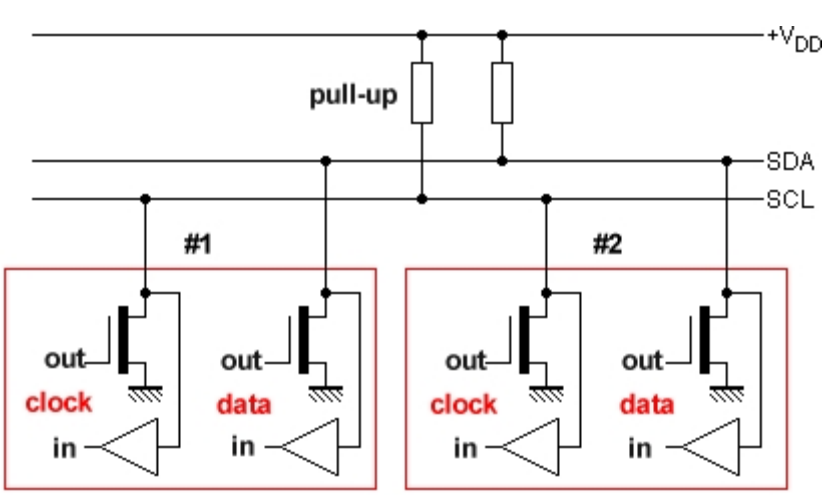
Transmitter	il dispositivo che sta inviando dati sul bus
Receiver	il dispositivo che riceve dati dal bus
Master	il dispositivo che inizia e termina un trasferimento dati.Prendein carico il bus in un dato momento e genera il clock della trasmissione
Slave	il dispositivo indirizzato dal Master; uno Slave non può comunicare direttamente con un altro Slave, ma solo con il Master attivo
Multi Master	l'architettura che consente la presenza di più di un Master nel bus; in un determinato istante solo uno è attivo
Arbitraggio	la procedura per assicurarela attivazione di un solo Master alla volta
Sincronizzazione	la tecnica per scambiare dati tra due o più dispositivi utilizzando un clock

Siccome I^2C è multi-Master, un dispositivo che opera come Master potrà operare anche come Slave di un altro Master. I Master sono solitamente dei microcontrollori e questo significa che più microcontrollori possono essere collegati al bus e comunicare sia tra di loro che con eventuali Slave.

Per riassumere, **I^2C è un bus seriale multi-Mastersincrono con due linee bi-direzionali**:

- la linea dati, chiamata **SDA** (*Serial DAta*)
- la linea del clock, chiamata **SCL** (*Serial CLock*).

La configurazione tipica è costituita da periferiche con interfaccia **open drain**, o open collector, "appese" a linee dotate di resistenze pull-up (***wired-AND***).



E' una configurazione single-ended con il riferimento alla tensione di alimentazione. I vari dispositivi sono aggiunti al bus semplicemente collegandoli ai due conduttori del bus.

I resistori di **pull-up**, dunque, **non sono opzionali**, ma sono indispensabili, in quanto le linee, se non mandate a livello basso alla chiusura degli open drain, sarebbero ad un livello logico indeterminato e quindi inaccettabile dagli ingressi di ricezione.

Questa soluzione circuitale consente di aggiungere (entro i limiti del collegamento fisico) periferiche al bus senza modifiche, consentendo una **scalabilità** verso un numero maggiore di dispositivi[1].

La soluzione open collector (o open drain) fa sì che, quando uno dei dispositivi manda in conduzione la propria uscita, essa forza a livello basso la linea (livello logico zero); nel momento in cui l'uscita si disabilita, la linea viene riportata a livello alto (livello logico uno) dal pull-up.
La configurazione presenta il vantaggio di **non avere conflitti hardware** poiché **nessun dispositivo può forzare il livello logico alto**: nel caso indesiderato che due dispositivi vadano in conduzione nello stesso istante, non si verifica alcuna sovracorrente poiché entrambi non sono altro che interruttori in parallelo, e la corrente viene limitata dal pull-up[2].

In sintesi, i conduttori del bus possono assumere solo due stati:

- ***float high*** quando nessun dispositivo è in conduzione; la tensione positiva è assicurata dai pull-up
- ***driven low*** quando un open drain (o open collector) viene chiuso, mandando la linea a livello basso

[1]Sono stati prodotti anche line extender che consentono di superare le brevi distanze costituite da cablaggi all'interno di una apparecchiatura o anche buffer in grado di portare i segnali fuori di essa. Per contro, la bi-direzionalità delle linee rende più complessa la bufferizzazione e l'isolamento galvanico rispetto a bus con linee mono direzionali.

[2]Ovviamente il pilotaggio del bus non può essere effettuato da GPIO generici che tipicamente hanno uno stadio di uscita push-pull, come la gran parte dei pin dei microcontrollori (come Arduino): se un dispositivo push-pull sul bus porta la sua uscita a livello alto mentre unaltro lo forza a livello basso, **si verifica un corto circuito** tra la tensione di alimentazione e la massa attraverso le uscite.Dovendo utilizzare questo genere di I/O, il livello alto richiederà il passaggio del pin **da uscita a ingresso** (ad alta impedenza), lasciando al solo pull-up il compito di attribuire alla linea il giusto livello alto. Tuttavia esistono microcontrollori con I/O open drain o programmabili come tali. Mancando questi sarà opportuno ricorrere a componenti esterni.

Ovviamente occorre che solamente una unità sia in trasmissione in un dato istante; diversamente, se più dispositivi agiscono contemporaneamente portando a livello basso la linea il contenuto dell'informazione diventa ambiguo. Questo richiede che **sia verificato lo stato di linea libera prima iniziare la trasmissione**.[3]

L'alimentazione è la stessa dei circuiti integrati e va tipicamente dai classici 5V al minimo supportato dai dispositivi collegati, ad esempio 3V, anche se non ci sono limiti teorici al valore della Vdd.[4]

Su un bus di questo genere non è consigliabile inserire altro che dispositivi I^2C, dato che il cambio di stato delle linee ha funzione di segnalazione; l'eventuale inserimento di altri dispositivi richiede la massima cautela.

I^2C prevede il supporto per una ampia gamma di frequenze di clock:

modo		data rate
standard-mode	**Sm**	100 kbps
fast-mode	**Fm**	400 kbps
fast-mode plus	**Fm+**	1 Mbps
high speed mode	**Hs**	3.4 Mbps

Trattandosi di una comunicazione sincrona con il clock separato dai dati, la precisione dell'oscillatore è relativa e, al limite, il clock può variare durante la trasmissione senza perdita di informazione[5].

Per quanto detto, sul bus possono comunicare solamente due dispositivi alla volta, in **modalità half-duplex**; l'accesso di più periferiche in trasmissione è controllato e impedito dai sistemi di handshake, mentre tutti i dispositivi collegati al bus si trovano sempre e comunque in stato di ricezione in attesa di

---

[3]Per questa ragione i dispositivi non invertono la loro direzione input/output a seconda che siano in trasmissione o ricezione, ma dispongono su ognuna delle linee di uno switch per la trasmissione (transistor bipolare open collector o MOSFET open drain) e, **contemporaneamente, un gate di ricezione**, tipicamente Schmitt trigger per minimizzare il rumore.Questa architettura è indispensabile non solo per la verifica dello stato *idle* della linea, ma per l'ulteriore ragione che i livelli logici sulle due linee non costituiscono solo la trama della trasmissione, ma, con particolari configurazioni, forniscono degli stati che il sistema identifica come handshake, detti "**condizioni**", che saranno descritti più avanti.

[4]Dato che, però, i dispositivi non solo trasmetto in linea, ma anche ricevono nello stesso tempo, se sullo stesso bus sono collegati dispositivi alimentati a tensioni diverse occorrerà introdurre traslatori di livello o altri artifici hardware per adattare i livelli logici, anche se dispositivi recenti hanno ingressi che supportano un'ampia gamma di tensioni.

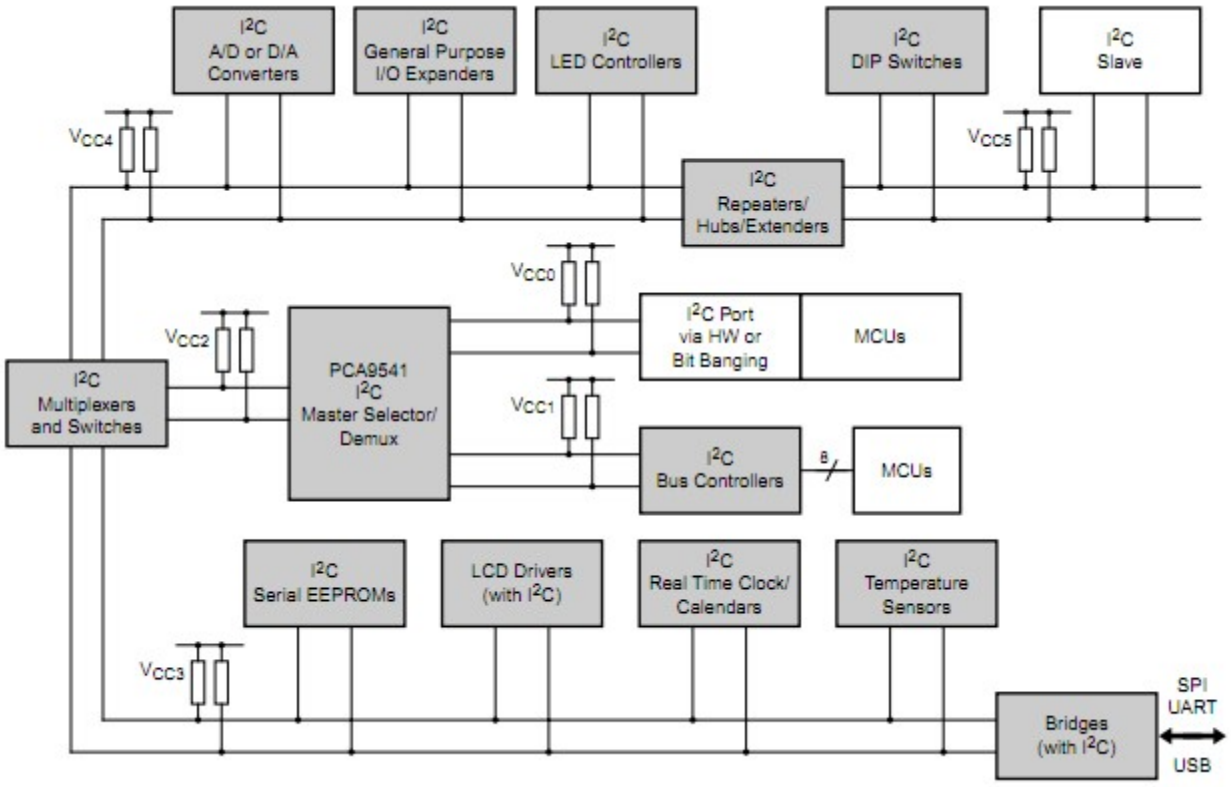
[5]Addirittura il protocollo prevede una azione, detta *clock stretching*, in cui l'unità in trasmissione rallenta il clock.Questo consente di mescolare su uno stesso bus unità differenti come prestazione e non far dipendere la larghezza di banda solamente da quella più lenta.

essere selezionati con il proprio indirizzo oppure di entrare in trasmissione come Master nel momento in cui le linee siano libere[6].

[6]Lo standard stabilisce anchesistemi di *general call* per inviare particolari messaggi a tutti i dispositivi sul bus contemporaneamente

Il bus I^2C

Philips-NXP ha ideato il bus in modo da collegare ogni genere di periferica in modo seriale ad uno o più microcontrollori che governano un sistema o una apparecchiatura. Le applicazioni spaziano dagli apparecchi TV ai personal computer, dagli strumenti di misura ai sistemi di supervisione di processo. Di seguito una figura che presenta un sistema complesso che interconnette dispositivi I^2C



La pubblicazione **UM1024** esemplifica un bus complesso dove sono presenti più microcontrollori e una suddivisione del bus stesso in sezioni attraverso multiplexer e repeater. Philips ha infatti realizzato una gamma molto ampia di funzioni per I^2C , non solo periferiche come I/O expander e EEPROM, ma anche elementi di supporto a bus complessi, come multiplexer, repeater, buffer, level translator, ecc.

Il protocollo I^2C

Il protocollo ha le seguenti caratteristiche:

- il trasferimento dati viene iniziato da un Master, con la linea di clock come sincronismo. Come per SPI, si tratta di un sistema a shift register ed essendo la linea clock separata da quella dati la sua frequenza può cambiare durante la trasmissione senza compromettere il funzionamento
- è necessario che ogni dispositivo che trasmette sul bus utilizzi un proprio clock. Data la natura sincrona della trasmissione, non occorre una elevata precisione negli oscillatori, ma le periferiche saranno più complesse che non semplici shift register

- non essendoci linee di selezione delle singole periferiche (i cosiddetti “enable” o “chip select”), per la comunicazione vengono utilizzati indirizzi a 7 o 10 bit; questo richiede una maggiore complessità delle periferiche
- sullo stesso bus possono essere presenti più Master;questo implica una procedura di arbitraggio, ottenuta manipolando le linee di clock e dati
- lo standard prevede un sistema di acknowledgement che consente ad ogni dispositivo Master di iniziare e chiudere la trasmissione e agli Slave di comunicare il successo o meno del trasferimento dei dati

Gli elementi fondamenti della trasmissione sono questi:

- la condizione di **START** (**S**)
- la condizione di **STOP** (**P**)
- la condizione di **REPEATED START** o **RESTART** (**R**)
- il pacchetto dati o indirizzo
- la condizione di **ACKNOWLEDGE** (**A**)

Le "condizioni" sono segnalazioni operate da un dispositivo per comunicare lo stato corrente del bus e della trasmissione in corso. Nel diagramma che segue è riportata una tipica sequenza di trasmissione:

La sequenza prevede:

- inizio trasmissione con la condizione di **Start**
- trasmissione dell’**indirizzo**
- condizione di **Acknowledge**
- trasmissione del **dato**
- condizione di **Acknowedge**
- trasmissione di un altro **dato**
- chiusura trasmissione con **Not-Acknowledge** e **Stop**

Osserviamo che, per un trasferimento di **8 bit**, sono utilizzati **9 impulsi di clock**: l’impulso aggiuntivo serve per segnalare la condizione Acknowlege / Not-Acknowledge.

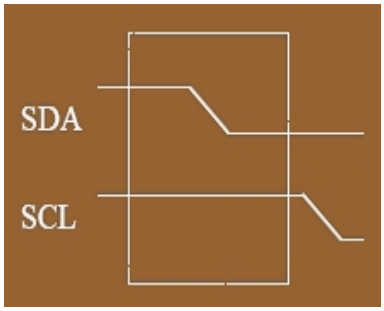
Le condizioni del bus I2C

Sono chiamate "**condizioni**" delle particolari configurazioni delle linee SDA e SCL che hanno lo scopo di segnalare ai dispositivi collegati al bus lo stato della comunicazione in corso. Eccole in dettaglio.

START

La prima condizione imposta è lo **START**, indicata con la lettera **S**: questa segnala ai dispositivi sul bus che uno di essi intende iniziare una trasmissione.

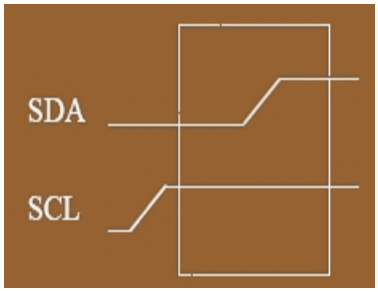
La condizione è realizzata mandando a **livello basso la linea SDA** mentre la linea **SCL è a livello alto**, poi abbassando anche la linea SCL.



Le temporizzazioni sono rilevabili dalle specifiche dei dispositivi. I microcontrollori hanno moduli di trasmissione che garantiscono il rispetto delle corrette temporizzazioni suddette.

STOP

La condizione di **Stop**, indicata con la lettera **P**, segnala che il dispositivo rilascia (abbandona) il controllo del bus.



Lo Stop è generato così: **SCL viene rilasciata** e **successivamente viene rilasciata anche SDA**. Alla fine della condizione di Stop, entrambe le linee sono a livello alto - situazione di *bus idle* - grazie alle resistenze di pull-up.

Una volta rilasciato il bus, esso può essere acquisito da un altro dispositivo. I moduli di comunicazione sincrona dei microcontrollori si fanno carico delle corrette temporizzazioni.

Start e **Stop sono condizioni generate dal Master**. Il bus è considerato occupato dopo lo Start e libero un certo tempo dopo lo Stop. Le temporizzazioni sono specificate dallo standard.

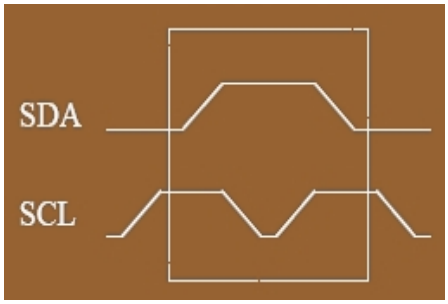
L'individuazione delle condizioni Start e Stop da parte dei dispositivi connessi al bus è facile se essi incorporano l'hardware necessario. I microcontrollori hanno moduli specifici per lo scopo[7].

[7]Microcontrollori senza tale interfaccia devono **testare lo stato della linea SDA almeno due volte più rapidamente della frequenza del clock** della trasmissione per rilevare correttamente la variazione di stato della linea.

RESTART

La condizione di **Restart** (*Repeated Start*), indicata con **R**, segnala che il dispositivo che sta controllando il bus intende inviare altri dati senza rilasciare il bus stesso.

Questo è utile quando si sta comunicando con un dispositivo che richiede più byte e non è conveniente interrompere la comunicazione tra un byte e il successivo; insomma, bisogna evitare che un altro dispositivo prenda il controllo del bus.



Sostanzialmente si tratta di una condizione di Stop seguita immediatamente da una di Start (vedi diagramma). Una condizione di Stop si ha quando SDA va alto mentre SCL è alto. Una condizione di Start si ha è quando SDA è mandato basso mentre SCL è alto.

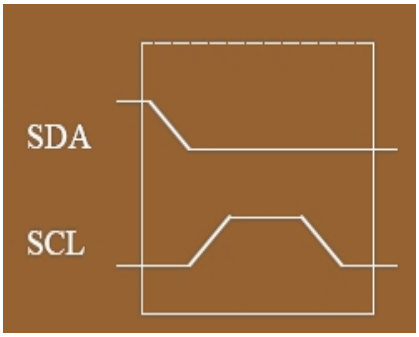
Quando utilizzare un Restart invece di uno Start? Durante un trasferimento c'è spesso la necessità di inviare un comando e poi leggere un dato di risposta dalla periferica. Questo richiede che il bus non sia preso nel frattempo da un altro Master, interrompendo l'operazione. Normalmente, dopo avere inviato il byte dell'indirizzo (indirizzo e lettura/scrittura bit) il Master può inviare un numero qualsiasi di byte, seguito da una condizione di Stop. Però, se è necessario riallineare lo Slave prima di inviare altri dati, invece di generare la condizione di Stop è consentito inviare il Restart, seguito da un indirizzo. Questo meccanismo consente un numero qualsiasi di Restart per consentire operazioni di lettura/scrittura combinate, verso uno o più dispositivi, senza rilasciare mai il bus, con la garanzia che la successione di operazioni non venga interrotta.

Un caso d'uso tipico è quello di accesso alle EEPROM. Occorre per prima cosa selezionare il dispositivo inviando il suo indirizzo, seguito dal comando di accesso; si inserirà poi un Restart per effettuare la lettura all'indirizzo desiderato della EEPROM senza che un altro Master prenda possesso del bus.

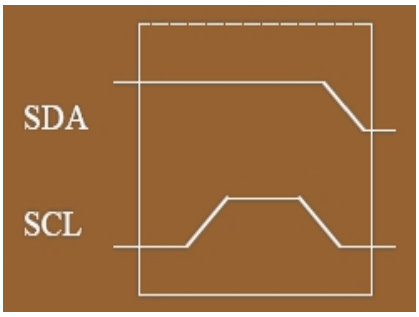
ACK e NACK

Un dispositivo può inviare la condizione **ACK**(**NACK**) come riscontro di un corretto(errato) trasferimento portando bassa (alta) la linea SDA durante il nono impulso del clock. **Il protocollo I^2C richiede che ogni byte trasmesso sia confermato con un ACK o un NACK.** 8 impulsi del clock sono necessari per i bit del byte trasmesso; l'impulso successivo (il nono) è utilizzato per generare ACK/NACK e viene emesso dal Receiver.

Un **ACK** si ottiene mandando basso SDA.



ACK è la conferma della corretta ricezione del dato; se riceve un ACK il master verifica implicitamente che la periferica chiamata è connessa e attiva.



Lo stato di **NACK**, il riscontro negativo, è segnalato dal livello alto di SDA sul nono impulso di clock. Siccome il bus in stato di idle è a livello alto, a causa dei pull-up, NACK è una risposta "passiva"; cioè se la periferica non risponde con ACK, non fa nulla ma di fatto risponde con un NACK; NACK significa che: o la trasmissione non è andata a buon fine o la periferica non è attiva.

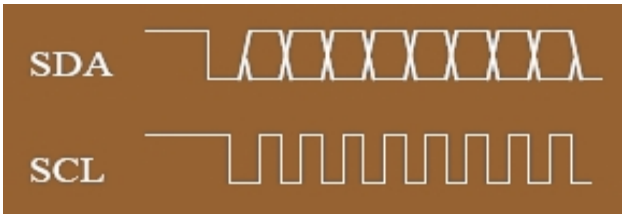
Per verificare la presenza di una periferica, sul microcontrollore che governa il bus occorrerà implementare un sistema per escludere o segnalare lo stato delle periferiche che rispondono con un NACK. La condizione NACK sarà presente quando:

- **nessun dispositivo di ricezione è presente** all'indirizzo chiamato
- **il ricevitore è impossibilitato a ricevere o trasmettere**
- durante il trasferimento di dati **il ricevitore ha ottenuto un dato o un comando che non può comprendere/gestire**
- **il ricevitore non può accettare ulteriori dati**
- **un Master in ricezione segnala allo Slave che non intende ricevere ulteriori dati**

Al ricevimento della condizione **NACK**, il Master può generare la condizione **Stop** per porre fine alla comunicazione, rilasciando il bus oppure un **Restart** per riavviare un nuovo tentativo di trasferimento dei dati, senza perdere il controllo del bus.

I DATI in I^{2}C

I dati trasmessi sono a 8 bit, dimensione dello shift register interno ai dispositivi I^{2}C. Se il dato ha dimensioni maggiori, occorrerà un adeguato numero di trasferimenti a 8 bit.



Per ogni bit immesso sulla linea SDA viene generato un impulso di clock sulla linea SCL. I dati sono validi con SCL a livello alto, precisamente al momento del **fronte di salita** del clock. Quando SCL non è a livello alto, i dati possono cambiare. Byte di dati sono usati per trasferire tutti i tipi di informazioni: quando si comunica con un altro dispositivo I^{2}C, gli 8 bit di dato possono essere: un codice di controllo, un indirizzo, dati grezzi. Occorrerà verificare i manuali dei vari dispositivi per individuare le specifiche necessità; periferiche differenti potranno usare codici uguali con scopi differenti.

I^{2}C trasmette sempre per primo il bit più significativo (MSb).

Gli INDIRIZZI sul bus I^{2}C

Il primo byte che viene immesso sul bus dopo uno Start è quello di indirizzo per lo Slave con cui comunicare. I^{2}C consente indirizzi a 7 bit o a 10 bit. Nel primo caso, i bit dell'indirizzo occuperanno i 7 bit più significativi, mentre il bit 0 (LSb) conterrà l'indicazione dell'operazione di lettura o scrittura.

Al momento in cui viene inviato un indirizzo, ogni dispositivo sul bus compara questo byte con il proprio indirizzo e, se la comparazione ha successo, risponde con un ACK.

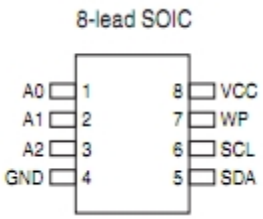
È evidente che sullo stesso bus non devono coesistere due dispositivi con lo stesso indirizzo.

Il bit 0 contiene l'informazione R/W che vale:

- **1** per indicare la richiesta di una **lettura di dati** da parte del Master
- **0** per indicare che la **scrittura di dati** sempre da parte del Master

Vari dispositivi presenti sul mercato hanno **un indirizzo fissato dal costruttore** oppure limitato a un numero di possibilità molto ridotto, ad esempio 2, impostato con dei jumper; in altri casi il costruttore del dispositivo rende disponibili alcuni pin per selezionare una parte dell'indirizzo.

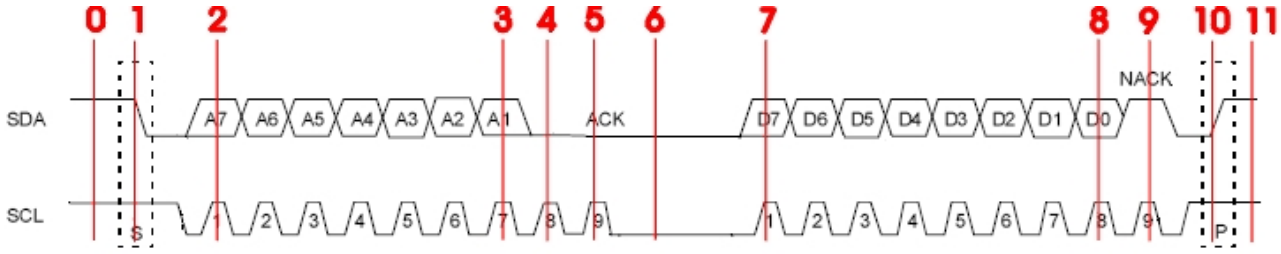
Ad esempio, nelle memorie **EEPROM** "seriali", 3 pin del piccolo package a 8 pin sono destinati alla selezione dell'indirizzo. A lato la piedinatura del chip **AT24CxxA** di **Atmel**: i pin A2:0 vanno impostati collegandoli a Vdd o Vss per formare la parte bassa parte dell'indirizzo. Nel caso di una **AT24C01** o **02** (1K o 2K) tutti e tre i pin sono attivi e quindi si possono avere sino a 8 memorie di questo tipo, opportunamente configurate con indirizzi differenti[8].



Va osservato che l'**assegnazione degli indirizzi ai dispositivi è coordinata da uno specifico comitato** (maggiori informazioni a www.nxp.com/i2c). Esiste anche un documento Philips del 1997 *i2c-bus Allocation Table - General* che riporta le assegnazioni di indirizzo di vari dispositivi.

Rivediamo il protocollo

Alla luce di quanto finora detto, possiamo tracciare con maggior precisione l'evoluzione di una sessione di comunicazione; i numeri distinguono le fasi successive:



0. Il dispositivo che intende trasmettere, e che diventerà quindi il Master, verifica che la linea sia *idle*, ovvero **SDA e SCL siano entrambi a livello alto**.
1. il Master acquisisce il controllo del bus generando la **condizione S-Start**. Gli altri dispositivi recepiscono questa condizione e si pongono in attesa dei dati successivi
2. viene inviato un indirizzo (nell' esempio 7 bit). Per ogni bit, appena diventato stabile, **SCL** viene portato a livello alto attraverso il pull-up. I dati sono validi con SCL = 1
3. Al settimo colpo di clock i bit di indirizzo sono esauriti
4. Viene generato **un ulteriore impulso di clock** per l'ottavo bit, che contiene l'informazione R/W. Nell' esempio R/W=0. Tutte le periferiche comparano l' indirizzo ricevuto con il proprio e quella che lo rileva coincidente al proprio si predispone per la risposta

[8]Se si vogliono collegare più periferiche uguali ma i jumper/pin sui dispositivi non sono sufficienti per differenziare le periferiche (per esempio: si voglionoinserire nel bus più di 8 memorie EEPROM AT24CxxA), occorrerà **spezzare il bus in segmenti**, ad esempio con multiplexer (**Philips PC9544** o simili), gestiti separatamente. Per quanto I^2C sia strutturalmente flessibile, è chiaro che si tratta di un appesantimento.

5. Un ulteriore **impulso sul clock**, il nono, serve alla periferica chiamata per inviare la condizione di **ACK**, mantenendo SDA a livello basso. Il Master rileva la condizione
6. Il Master mantiene il controllo del bus
7. ed inizia la trasmissione del dato successivo, che si svolge come il precedente
8. Tutti gli 8 bit del byte dello shift register sono "dato", per cui occorrono 8 impulsi di clock
9. Al nono impulso di clock, la periferica risponde con il riscontro (in questo esempio con un **NACK**, non agendo sulla linea SDA che resta a livello alto)
10. il Master chiude la trasmissione generando la condizione **P-Stop**. I dispositivi sul bus rilevano tale condizione e sanno che il Master ha deciso di liberare il bus
11. Il Master rilascia il bus, le cui linee si riportano a livello alto, grazie ai pull-up. Ora un altro dispositivo potrà assumere il controllo delle linee

Il protocollo non è particolarmente complesso: è ragionevole e logico nello sfruttare al massimo le due linee di trasmissione. Ma non è neppure banale, in quanto è necessario che i dispositivi sul bus identifichino le condizioni, generino i segnali con le giuste temporizzazioni, ecc. Il tutto con la massima frequenza possibile del clock[9][10][11].

---

[9] Se poi il dispositivo deve supportare il clock stretching, occorrerà anche che il Master verifichi, al rilascio della linea SCL, il suo effettivo stato a livello alto (e, per lo Slave, sia implementata questa funzione).

[10] Una emulazione software del protocollo è fattibile utilizzando le istruzioni di un microcontrollore privo di modulo specifico per la comunicazione sincrona, e con I/O altrettanto generici, ma è ragionevole solamente nell'ambito di un bit-banging.
Implementare il protocollo per ogni aspetto della comunicazione e, principalmente per una gestione non in polling, richiede risorse e un lavoro ben più costosi che non la scelta di un microcontrollore che disponga del modulo adeguato. Modulo che, peraltro, è presente nella gran parte dei prodotti più recenti, che, per inciso, avranno costi minori di quelli più datati.

[11] Inoltre è comprensibile che sia necessario, nell'analisi dello stato delle linee in attesa delle varie condizioni, una frequenza di lavoro molto maggiore di quella del clock, sia che si tratti di una soluzione hardware sia, anzi decisamente di più, nel caso si tratti di una soluzione software. Questa necessità di campionamenti è la causa per cui, anche nei microcontrollori dotati di modulo di comunicazione sincrona, il clock massimo consentito è solitamente molto minore di quello primario: ad esempio, nei PIC18F, con Fclock= 40 MHz, il clock sul bus I2C dichiarato è di 400 kHz, 100 volte inferiore.Ne deriva che emulazioni software, comunque sempre possibili, avranno anche una prestazione inferiore a quella ottenibile con il modulo adeguato in cui, ad esempio, l'identificazione delle condizioni non è effettuata da istruzioni, ma dalla logica hardware del modulo stesso.Quindi è comprensibile come trasmissioni High Speed a 3.4 MHz richiedano dispositivi particolari, oltre ad una struttura fisica del bus adeguata a questa frequenza.Sarà invece più semplice operare con clock a frequenza minore, dove il 400 kHz rappresenta un valore limite per molte periferiche.