La connessione I^2C

I^2C (*Inter Integrated Connection)* 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.

Esiste poi un **CBUS** che però presenta varie differenze anche hardware. Troviamo anche costruttori che realizzano dispositivi denominati genericamente *2-Wire Serial Interface*, con una compatibilità I^2C.
Comunicazioni del genere I^2C sono presenti in molte applicazioni, ad esempio nell' interfaccia tra schede video e monitor (VGA, HDMI) o per l' accesso ai *serial presence detect* (SPD) delle DIMM e altri componenti per sistemi computerizzati, ecc.

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.
Il sistema include la possibilità di avere più dispositivi sia Master (**multi-Master**) che Slave, collegati tra di loro utilizzando solamente due conduttori, uno per i dati ed uno per il clock.

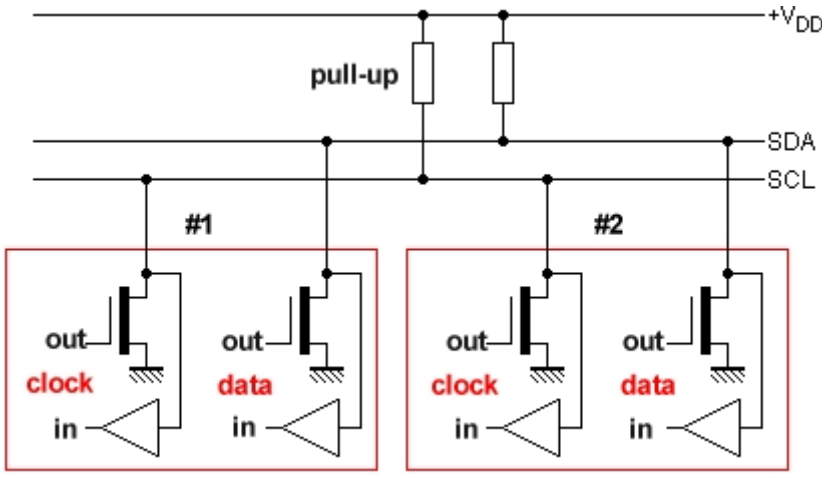
Siccome sia Master che Slave possono trasmettere e ricevere dati, la **linea dati è bi-direzionale**.
E lo è **anche il clock**, in quanto sarà il dispositivo in linea in quel momento a comandarlo.
In effetti, secondo la definizione originale del protocollo, sono identificati i seguenti elementi:

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. Il Master è il dispositivo che può prendere in carico il bus in un dato momento e che sta generando il clock della trasmissione.
Slave	il dispositivo indirizzato dal Master. Sono dispositivi Slave quelli che non possono comunicare tra di loro, ma solo con un Master.
Multi Master	più di un Master che possono cercare di avere il controllo del bus senza che siano corrotti i messaggi in linea
Arbitraggio	l' operazione che assicura ad un solo Master alla volta l' accesso al bus
Sincronizzazione	procedura per sincronizzare due o più dispositivi.

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 microcontroller e questo significa che più microcontroller possono essere collegati al bus e comunicare sia tra di loro che con eventuali Slave.

Per riassumere, **I^2C è un sistema seriale sincrono con due linee bi-direzionali, multi-Master.**

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



Dovendo consentire accessi bi-direzionali alle linee fisiche di trasporto del clock e dei dati, ne risulta che la configurazione tipica è costituita da periferiche con interfaccia **open drain**, o open collector, "appese" a linee dotate di pull-up (***wired-AND***).

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

I 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. 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-dierzionalità delle linee rende più complessa la bufferizzazione e l' isolamento galvanico rispetto a bus con linee mono direzionali.

La soluzione open collector (o open drain) fa si 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 ha il vantaggio di **non avere in nessun caso conflitti hardware**, dato che **nessun dispositivo può forzare il livello logico alto**: al massimo, se due dispositivi vanno in conduzione nello stesso istante, non si verifica alcuna sovracorrente, dato che entrambi non sono altro

che interruttori in parallelo e la corrente viene limitata dal pull-up.
Ovviamente il pilotaggio del bus non può essere effettuato da GPIO generici che tipicamente hanno una uscita puh-pull, come la gran parte dei pin dei microcontroller: se un dispositivo push-pull sul bus porta la sua uscita a livello alto mentre un' altro lo forza a livello basso, **si verifica un corto circuito** tra la tensione di alimentazione e la massa attraverso i semiconduttori di pilotaggio delle uscite.
Dovendo utilizzare questo genere 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 microcontroller con I/O open drain o programmabili come tali.
Mancando questi sarà opportuno ricorrere a componenti esterni.

In sostanza, 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.

Ovviamente occorre che solamente una unità sia in trasmissione in un dato istante, dato che se più unità agiscono contemporaneamente portando a livello basso la linea, non si verificano sovracorrenti, ma il contenuto dell' informazione va perso. Questo richiede che **sia verificato lo stato di linea libera prima iniziare la trasmissione**.
Per questa ragione i dispositivi non invertono la loro direzione 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 decritti più avanti.

La comunicazione si svolge al livello dell' alimentazione dei circuiti integrati, che va tipicamente dai classici 5V al minimo supportato dai dispositivi collegati, ad esempio 3V, anche se non ci sono teoricamente limiti al valore della Vdd.
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 una ampia gamma di tensioni.

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:

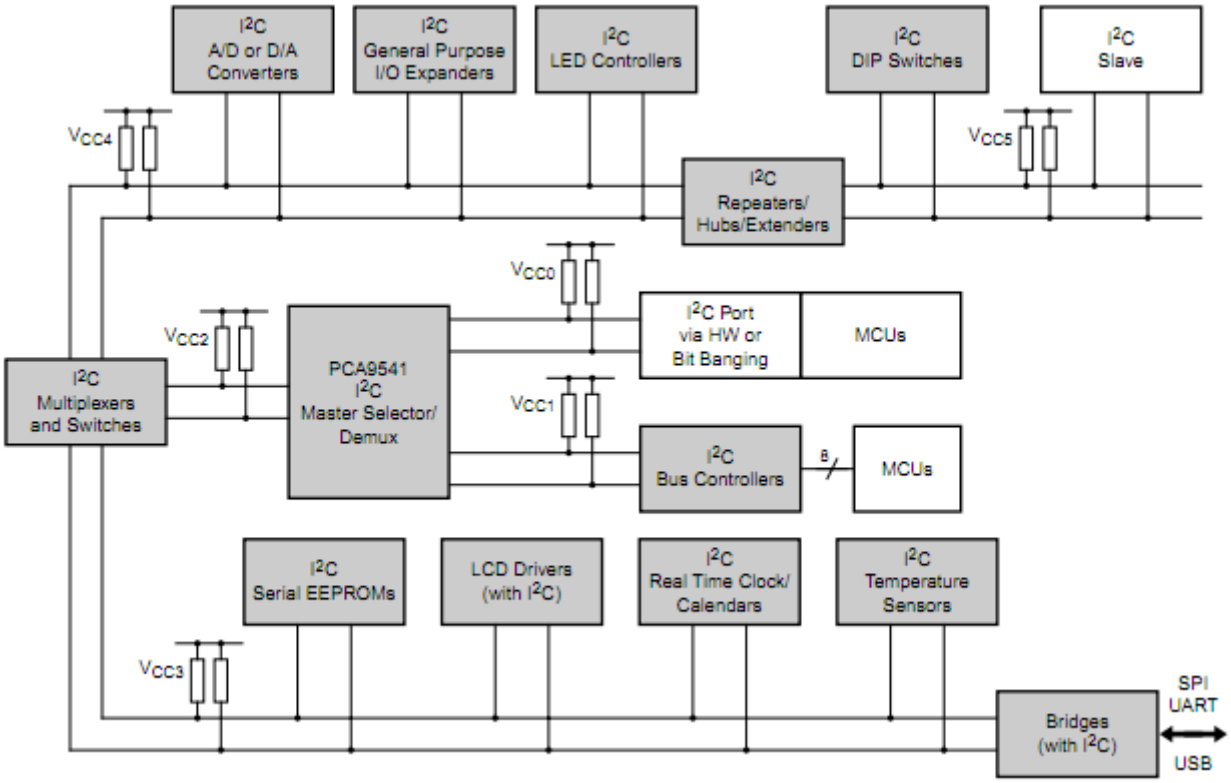
modo		data rate
standard-mode	**Sm**	100 kbps
fast-mode	**Fm**	400 kbps
fast-mode plus	**Fm+**	1 Mbps
hig speed mode	**Hs**	3.4 Mbps

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 del dato, tanto che 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.

Come è evidente, sul bus potranno comunicare solamente due dispositivi alla volta in una **modalità half-duplex**; l' accesso di più periferiche in trasmissione è controllato ed impedito dai sistemi di handshake, mentre tutti i dispositivi collegati al bus si trovano sempre e comunque in stato di ricezione in attesa di essere selezionati con il proprio indirizzo oppure di entrare in trasmissione come Master nel momento in cui le linee siano libere; lo standard stabilisce sistemi di *general call* per inviare particolari messaggi a tutti i dispositivi sul bus contemporaneamente.

Il bus I2C

Philips-NXP ha previsto il bus per collegare ogni genere di periferica in modo seriale ad uno o più microcontroller 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.



La pubblicazione **UM1024** esemplifica un bus complesso dove sono presenti più microcontroller e una suddivisione del bus stesso in sezioni attraverso multiplexer e repeater.

Philips ha infatti realizzato una gamma molto ampia di funzioni per I²C , 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 I²C

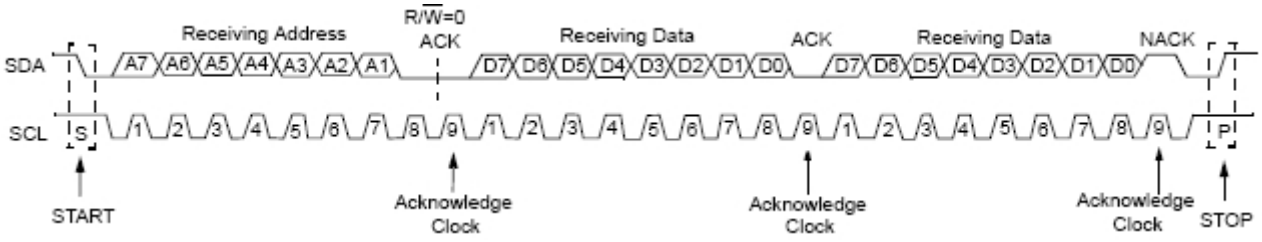
Il protocollo ha le seguenti caratteristiche:

- 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 danneggiarla.
- Inoltre 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 periferiche come in SPI, per la comunicazione vengono utilizzati degli indirizzi a 7 o 10 bit. Anche questo richiede una maggiore complessità delle periferiche.
- Contrariamente a SPI, sullo stesso bus possono essere presenti più Master. Questo vuol dire che è necessaria una procedura di arbitraggio, che è ottenuta manipolando le linee di clock e dati.
- La complessità è necessaria anche per supportare un sistema di acnowledgement che consente ad ogni dispositivo Master di iniziare e chiudere la trasmissione ed agli Slave di comunicare il successo o meno del trasferimento dei dati. Questo è ottenuto manipolando le due linee di collegamento.

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 **ACNOWLEDGE** (**A**)

Le "condizioni" sono i segnali che un dispositivo applica al bus per comunicare lo stato corrente del bus e della trasmissione in corso. Osserviamo in un diagramma una sequenza tipica di trasmissione:



La sequenza comprende:

- inizio trasmissione con la condizione di **Start**

- trasmissione **indirizzo**
- condizione di **Acknowledge**
- trasmissione **dato**
- condizione di **Acknowedge**
- trasmissione **dato**
- chiusura trasmissione con **Not-Acknowledge** e **Stop**

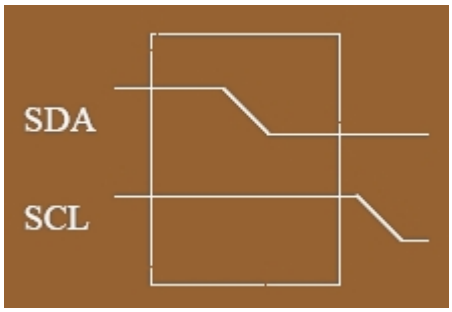
Osserviamo che, per un trasferimento di **8 bit**, sono utilizzati **9 impulsi di clock**: questo impulso in più serve al protocollo per determinare la condizione Acknowlege - Not Acknowledge.

Le condizioni del bus I2C

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

START

La prima condizione imposta è lo **START**: questa indica ai dispositivi sul bus che uno di essi intende iniziare una trasmissione.



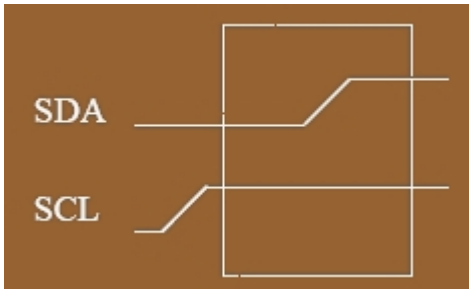
La condizione è realizzata mandando a **livello basso la linea SDA** mentre la linea **SCl è a livello alto**, poi abbassando anche la linea SCL.

Le temporizzazioni sono rilevabili dalle specifiche dei dispositivi. Nel caso di impiego di microcontroller con moduli di trasmissione sincrona, il problema di rispettare temporizzazioni corrette è superato in quanto il modulo stesso provvede in tal senso, a seconda del clock.

La lettera **S** indica questa condizione.

STOP

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



Anche Stop è generato manipolando lo stato delle due linee: **SCL viene rilasciata** e **successivamente viene rilasciata anche SDA**. Alla fine della condizione di Stop, entrambe le linee sono a livello alto grazie alle resistenze di pull-up (*bus idle*).

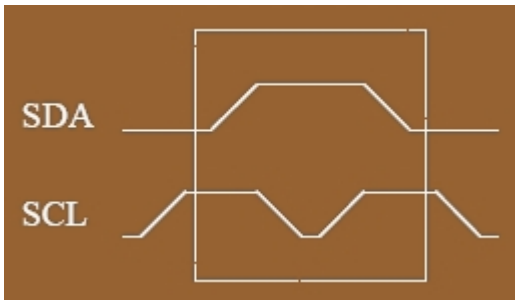
Una volta rilasciato il bus, esso può essere acquisito da un altro dispositivo.
Anche in questo caso i moduli di comunicazione sincrona si fanno carico delle corrette temporizzazioni.

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 nella descrizione dello standard.

L' individuazione delle condizioni Start e Stop da parte dei dispositivi connessi al bus è facile se essi incorporano l'hardware necessario. Nei microcontroller si tratta di un modulo specifico per la gestione delle comunicazioni sincrone. Microcontroller 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 un colloquio costituito da più bytes e che non è conveniente sospendere per evitare che un altro dispositivo possa prendere il controllo del bus.

Sostanzialmente si tratta di una condizione di Stop seguita immediatamente da una di Start, come possiamo vedere nel diagramma.

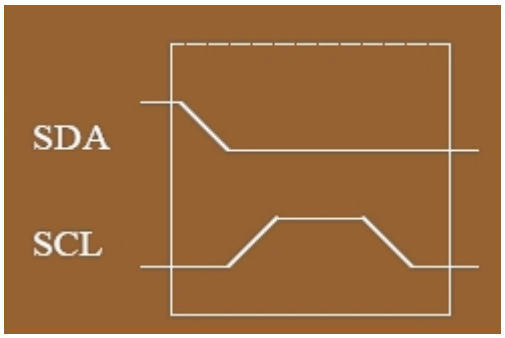
Una condizione di Stop si ha quando SDA va alto mentre SCL è alto. Una condizione di Start si ha è quando SDA è mandato basso mentre SCL è alto.
I moduli di gestione delle comunicazioni sincrone generare automaticamente anche questa condizione con le corrette temporizzazioni.

Quando si utilizza una condizione di Restart invece di una condizione di 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 da un altro Master e l' operazione sia interrotta. Il protocollo definisce quindi la condizione di Restart.
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.
Questa struttura ricorsiva consente un numero qualsiasi di condizioni di restart per consentire operazioni di lettura/scrittura combinate su a uno o più dispositivi senza rilasciare il bus e, quindi, con la garanzia che l'operazione 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 ora un Restart per fare in modo che sia effettuata la lettura all' indirizzo interno alla EEPROM voluto senza che un altro Master prenda possesso del bus.

ACK e NACK

Un dispositivo può inviare una condizione **ACK** o riconosce un trasferimento di un byte correttamente recepito portando basso la linea SDA durante il nono impulso del clock.
Il protocollo I^2C richiede che ogni byte trasmesso deve essere concluso con un NACK o ACK.

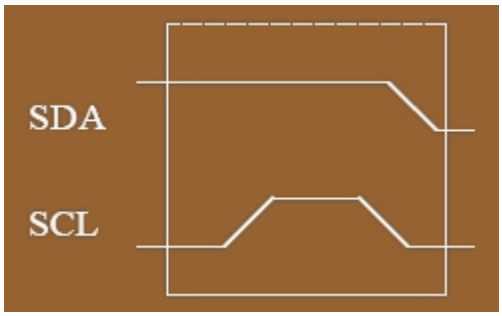


8 impulsi del clock sono utilizzati per il sincronismo con i dati trasmessi dal dispositivo master, mentre il nono serve a sincronizzare la condizione di Acknowledge dell' unità ricevente.

Questo si ottiene mandando basso la linea dati. In caso contrario, SDA sarà mantenuta a livello alto dal pull-up.

Quindi, per inviare un ACK occorre che la periferica agisca sullo SDA.

Siccome ACK è la conferma della corretta ricezione del dato, questa condizione consente al master di verificare il fatto che la periferica chiamata è connessa e attiva.



Infatti lo stato di **NACK**, ovvero la negazione della corretta ricezione del dato è verificata dal livello alto di SDA sul nono impulso di clock.

Siccome il bus non controllato (idle) è a livello alto per i pull-up, NACK è una risposta "passiva": se la periferica non risponde ACK al byte trasmesso, questo indica che o la trasmissione non è andata a buon fine o la periferica non è attiva.

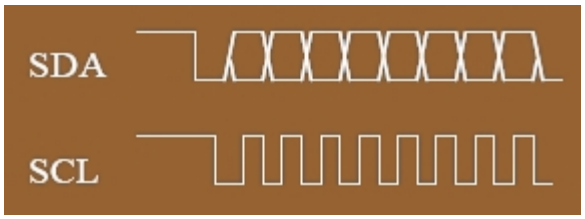
Se occorre verificare la presenza di una periferica, sul microcontroller che governa il bus occorrerà implementare un sistema di per escludere o segnalare lo stato delle periferiche che rispondono con un NACK.
La condizione NACK sarà presente quando:

- **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**
- **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 trasferimento dati, senza perdere il controllo del bus.

I DATI in I^{2}C

I dati trasmessi sono a 8 bit, dato che lo shift register ha questa dimensione. 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.

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 dati possono essere un codice di controllo, un indirizzo o dati grezzi. Occorrerà verificare i manuali dei vari dispositivi per individuare le specifiche necessità: diverse periferiche potranno usare codici uguali con fini differenti.

I dati devono essere validi al momento del **fronte di salita** del clock.

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 la periferica da attivare. 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, se lettura o scrittura.

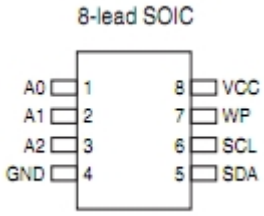
Quando viene inviato, ogni dispositivo sul bus compara questo byte con il proprio indirizzo e, se la comparazione ha successo, risponde con un ACK.

E' evidente che sullo stesso bus non possono coesistere due dispositivi con lo stesso indirizzo.

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

- **1** per indicare che è richiesta una **lettura di dati**
- **0** per indicare che sarà effettuata una **scrittura di dati**

Dato che il fattore costo è sempre sensibile, è comune che vari dispositivi abbiano **un indirizzo fissato dal costruttore** oppure limitato ad un numero di possibilità molto ridotto, ad esempio 2; in altri casi il costruttore del dispositivo rende disponibili alcuni pin per selezionare una parte dell' indirizzo.



Ad esempio, nelle memorie **EEPROM**, 3 pin del piccolo package a 8 pin vengono dedicati alla selezione dell' indirizzo. A lato la piedinatura di **AT24CxxA** di **Atmel**: i pin A2:0 vanno polarizzati collegandoli a Vdd o Vss per formare parte dell' indirizzo.
Nel caso di una **AT24C01** o **02** (1K o 2K) tutti e tre i pin sono attivi.
Però una **AT24C16** (16k) non li utilizza, impiegando i bit forniti dal Master nell' indirizzo per selezionare le pagine interne.

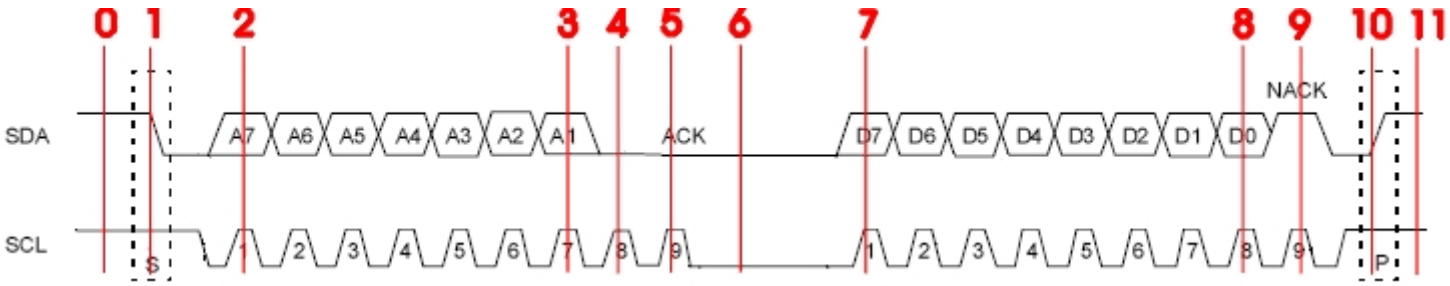
La situazione non è comune, ma è possibile che si vogliano collegare più periferiche uguali: ne risulta che, ove sia necessario avere sullo stesso bus un numero di dispositivi superiore alla alla varietà degli indirizzi, occorrerà **spezzare il bus in segmenti**, ad esempio con multiplexer (**Philips PC9544** o simili), gestiti separatamente. Per quanto I^{2}C sia strutturalmente flessibile, è chiaro che si tratta di un appesantimento.

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 vedere, quindi, con maggior consapevolezza lo svilupparsi di una sessione di comunicazione.



0. Il dispositivo che intende trasmettere, e che diventerà quindi il Master, verifica che la linea sia *idle*, ovvero **SDA e SCL sia rilasciati a livello alto**.
1. Quindi 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.
   SDA e SCL sono costretti a livello basso.
2. Per primo viene inviato un indirizzo. Nell' esempio, 7 bit. Per ogni bit **viene rilasciato SCL**, che va 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 uguale si predispone per la risposta.
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 e 8 i bit del byte dello shift register sono "dato", per cui occorrono 8 impulsi di clock
9. Al nono impulso di clock, la periferica risponde, in questo esempio, con un **NACK**, non agendo sulla linea SDA, che resta a livello alto.
10. Quindi il Master chiude la trasmissione generando uno **Stop-S**. I dispositivi sul bus rilevano la condizione che avvisa dell' intenzione del Master di liberare il bus.
11. Il Master rilascia il bus, le cui linee si riportano a livello alto, grazie ai pull-up. Ora un qualsiasi dispositivo potrà assumere il controllo delle linee.

Il protocollo, dunque, non è particolarmente complesso: visto alla luce di quanto finora detto è ragionevole e logico nello sfruttare al massimo le due linee di trasmissione.

Ma non è neppure semplice, in quanto è necessario che i dispositivi sul bus identifichino le condizioni, generino i segnali con le giuste temporizzazioni, ecc. Il tutto con il massimo clock possibile.
Se poi il dispositivo deve supportare ilo 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).
Da quanto sopra si deduce che una emulazione software del protocollo è fattibile utilizzando le istruzioni di un generico microcontroller, 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 microcontroller che disponga del modulo adeguato. Modulo che, peraltro, è presente nella gran parte dei prodotti più recenti, che, tra l' altro, avranno costi minori di quelli più datati.

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, e decisamente di più, nel caso si tratti di una soluzione software.

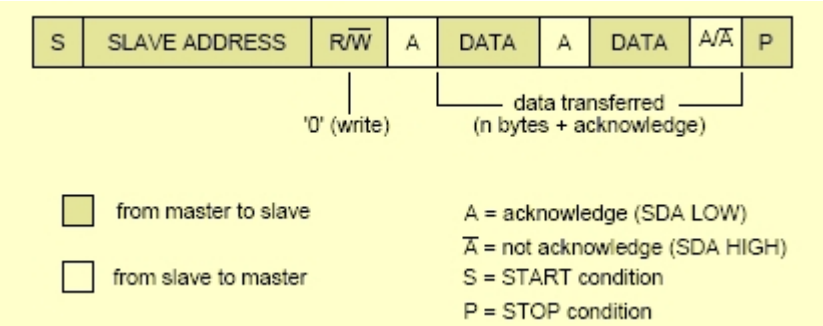
Questa necessità di campionamenti è la causa per cui, anche nei microcontroller dotati di modulo di comunicazione sincrona, il clock massimo consentito è solitamente molto minore di quello primario: ad esempio, nei PIC18F, con Fosc di 40 MHz, il massimo baud rate ottenibile è di 400 kHz, ovvero 100 volte inferiore.
Ne deriva che emulazioni software, comunque sempre possibili, avranno anche una prestazione inferiore a quella ottenibile con il modulo adeguato, dove, 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.

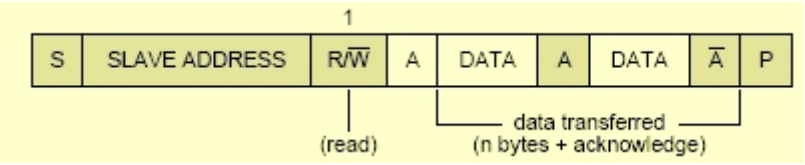
Il trasferimento dati

I formati di trasferimento dati possibili sono:

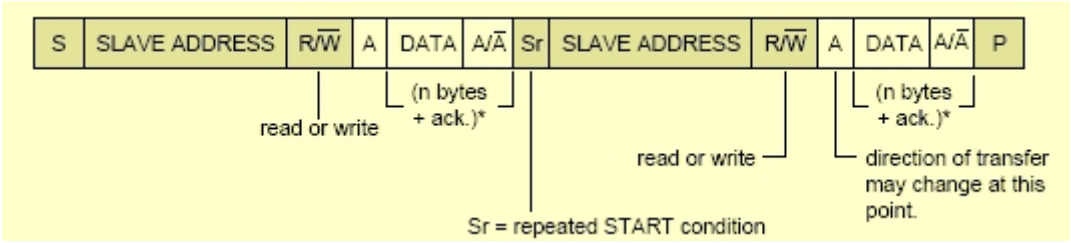
- Il Master trasmette al ricevitore Slave. La direzione di trasferimento è diretta dal Master allo Slave.



- Il Master legge dallo Slave dopo avergli trasmesso il primo byte di indirizzo. Al momento dell' ACK, il Master-trasmettitore diventa un Master-ricevitore e il ricevitore Slave diventa un trasmettitore slave. Il primo acknowledge, dopo l' indirizzo, è ancora generato dallo Slave; i successivi vengono generati dal Master. La condizione STOP viene generata dal master, che manda un NACK e poi uno Stop quando intende chiudere la comunicazione.



- Formato combinato. Durante un cambio di direzione all'interno di un trasferimento, la condizione di Start e l'indirizzo dello Slave sono ripetute, ma con il bit R/W invertito. Se un Master-ricevitore invia una condizione di Restart, invia prima un NACK. Questo genere di formato può essere utilizzato, ad esempio, per controllare una memoria seriale. Con il primo byte di dati viene scritta una locazione interna alla memoria; dopol Restart e l' invio di un indirizzo, viene effettuato il trasferimento dati.



Condizioni illegali

L' invio di uno Start seguito immediatamente da uno Stop è un formato illegale, anche se molti dispositivi lo supportano. Questa successione non è da confondere con quella di Restart, dato che Restart è valido se prima è stato inviato uno Start.

Indirizzi speciali

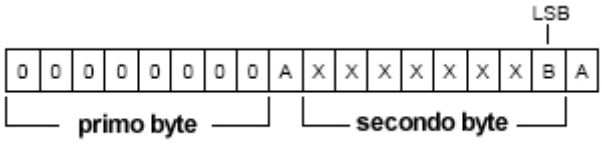
Per quanto riguarda **gli indirizzi**, lo standard ne prevede alcuni collegati a funzioni speciali:

Indirizzo	R/W	Note
0000 000	0	indirizzo della ***General Call***
0000 000	1	Start bit. Nessun dispositivo risponde a questo indirizzo con un ACK
0000 001	x	Indirizzo riservato per mescolare dispositivi CBUS e I^2C sullo steso bus. I dispositivi I^2C non rispondono a questo indirizzo.
0000 010	x	Riservato per la compatibilità di altri dispositivi con il bus I^2C.
0000 011	x	Riservato per future applicazioni
0000 1xx	x	Riservato al master in modo high speed
1111 1xx	x	Riservato per future applicazioni (identificazione ID)
1111 0xx	x	indirizzo a 10 bit

Vediamo qualche dettaglio.

General Call

L'indirizzo di General Call (chiamata generale) viene utilizzato per trasmettere a tutte le periferiche dell'intero bus, indipendentemente dalla loro indirizzo. Va tenuto conto, però, che non tutti i dispositivi supportano questa funzione e ignoreranno la chiamata.
Se il dispositivo non supporta la ***General Call*** o non richiede dati dopo questa, non risponde con un acknowledgement. Se, invece, lo Slave richiede dati dopo la chiamata, genera un ACK.



Il significato della ***General Call*** è precisato dal secondo byte che segue l' indirizzo di chiamata; in particolare il bit 0 ha una funzione di definizione particolare.
Se il bit 0 del byte dati è uguale a 0, il byte dati ha le seguenti possibilità:

- ***00000110 (06h)*** - reset software e scrittura della parte programmabile dell' indirizzo dello Slave. Il dispositivo che accetta questo byte resetta il suo hardware interno; inoltre tutti i dispositivi che definiscono la parte programmabile del loro indirizzo in hardware ricaricano questa parte programmabile. Il dispositivo però non resetta.
- ***00000100 (04h)*** - scrittura della parte programmabile dell' indirizzo dello Slave. Tutti i dispositivi che definiscono la parte programmabile del loro indirizzo in hardware ricaricano questa parte programmabile. Il dispositivo però non resetta.
- ***00000000 (00h)*** - Codice riservato

Se il bit 0 è uguale a 1, significa che questa chiamata è effettuata da un dispositivo Master hardware che non dispone di alcuna informazione preliminare degli indirizzi slave collegati. In questo caso il master fa la chiamata con il proprio indirizzo in modo che gli slave possono identificare la fonte del messaggio: i sette bit più significativi del secondo byte contengono l' indirizzo di questo Master. Questo permette ad un dispositivo intelligente, come un microcontroller, di rilevare la presenza del master hardware; se questo può operare anche come Slave, il suo indirizzo Slave sarà quello trasmesso.
La gestione della General Call ha anche lo scopo di realizzare sistemi auto configuranti, implementando sul Master configuratore gli opportuni algoritmi.

START bit

Un microcontroller può essere collegato al bus I2C in due modi:

- se dispone di una un'interfaccia on-chip hardware per la gestione del bus I2C può essere programmato per essere interrotto dalle condizioni di Start sul bus.
- se il dispositivo non dispone di un'interfaccia, deve monitorare costantemente il bus via software.

Ovviamente, più volte il microcontroller effettua sondaggi sul bus (polling), meno tempo ha a disposizione per eseguire altre funzioni. C'è quindi una differenza di velocità tra dispositivi con un supporto hardware e quelli senza che si basano su polling software. In questo caso, il trasferimento dei dati può essere preceduto da una procedura di Start che è molto più lunga del normale. La procedura di partenza è costituito da:

- un invio della condizione di Start
- l' invio del byte di start (0000 0001)
- un acknowledge del clock (ACK)
- una condizione di Restart (Sr).

Dopo la condizione di Start che è stata trasmessa da un master per accedere al bus, viene trasmesso il byte riservato START (0000 0001). Un altro microcontroller può quindi testare la linea SDA con una frequenza di campionamento relativamente bassa fino a quando uno degli zeri nel byte START viene rilevato. Dopo il riscontro di questo livello basso sulla linea SDA, il microcontroller può aumentare la frequenza di campionamento per identificare la condizione di Restart che viene poi utilizzata per la sincronizzazione. Un ricevitore hardware verrà reimpostato al ricevimento della condizione di Restart e quindi ignorerà il byte di inizio. Un acknowledge viene generato dopo il byte di inizio solo per conformarsi con formato utilizzato sul bus, dato che a nessun dispositivo è consentito di rispondere con un ACK al byte di Start.
Attraverso il byte di Start è possibile implementare anche sistemi di auto rilevamento degli elementi sul bus e auto configurazione del sistema.

CBUS

Ricevitori **CBUS** possono essere collegati al bus I^2C . Tuttavia, CBUS utilizza un terzo conduttore, detto ***DLEN***, che assume funzione di acknowledge e sostituisce la condizione ACK del bus I^2C. Anche il formato dei dati può essere differente: normalmente, in I^2C le trasmissioni sono sequenze di byte di 8 bit, mentre i dispositivi compatibili CBUS hanno diversi formati.
Ne risulta che, per poter realizzare una struttura mista in cui sul bus I^2C si trovano anche dispositivi CBUS, si utilizza uno speciale indirizzo (0000001X) a cui essi rispondono.
Dopo la trasmissione dell'indirizzo CBUS, la linea DLEN può essere attivata e inviata una trasmissione in formato CBUS. La trasmissione è chiusa da una condizione STOP, riconosciuta da tutti i dispositivi.

Compatibilità con altri dispositivi

I^2C prevede la possibilità di inserire sullo stesso bus dispositivi che usino lo stesso hardware, ma differente protocollo. All' indirizzo 0000010X risponderanno i dispositivi che appartengono a questa categoria e i dispositivi I^2C che rispondono ad entrambi i protocolli.

High speed

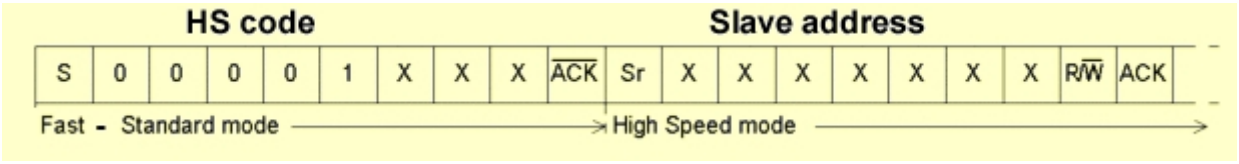
I2C prevede una estensione con clock fino a 3.4MHZ, chiamata high speed (Hs).
I dispositivi ad alta velocità sono compatibili verso il basso, permettendo per sistemi bus misti.

I segnali in modo HS sono identificati come SDAH e SCLH e sono applicate varie modifiche allo standard finora descritto.
Per abbreviare il tempo di salita del segnale a 3.4 MHz, i Master dispositivi hanno una combinazione di un open-drain pull-down e un generatore di corrente pull-up su SCL, abilitato durante il modo Hs e solo per un Master. I Master Hs generano un segnale di clock seriale con rapporto tra il periodo basso e quello alto di 1: 2.
Le periferiche possono avere un ponte incorporato per separare il dispositivo a bassa velocità dal bus durante il trasferimento ad alta velocità: lo scopo principale di tale ponte è di ridurre il carico capacitivo sul bus ed evitare possibili conflitti causati da dispositivi a bassa velocità.

Lo schema di indirizzamento per i trasferimenti ad alta velocità si differenzia dalla normale procedura di indirizzamento. Una trasmissione ad alta velocità si avvia in modalità full o standard, cioè al massimo 400 kbit.

- Dopo la condizione di Start viene trasmesso un byte '00001XXX', seguito da un not-acknowledge obbligatorio. I tre bit meno significativi vengono utilizzati per identificare diversi Master sullo stesso bus, ognuno dotato del suo identificatore univoco. Durante la trasmissione del codice si può avere una situazione di arbitraggio, così che solo un Master acquisti il controllo del bus. L' arbitraggio, però, non avviene in Hs.
  Con gli ultimi 3 bit dell' indirizzo speciale possono far parte del bus ad alta velocità fino a 8 Master, anche se il codice '00001000' dovrebbe essere riservato per scopi diagnostici e test.
- Dopo la fase di acknowledge, il trasferimento ad alta velocità inizia con una condizione di avvio ripetute, seguita dall'indirizzo Slave e da i dati successivi, come in modalità Fast o Standard, ma con un clock che può arrivare a 3.4 MHz.
- Il generatore di corrente, attivato dopo la trasmissione del codice iniziale, viene disattivato dopo ogni Restart o ACK o NACK per dare modo allo Slave di operare il clock stretching
- La modalità ad alta velocità rimane attiva fino a quando viene trasmessa una condizione di Stop, sulla quale i dispositivi collegati ad alta velocità tornano a rallentare la trasmissione come modalità Fast o Standard

Nell'immagine qui sotto viene esemplificato l'inizio di un trasferimento ad alta velocità.



Bus con questa velocità di clock richiedono dispositivi particolari e una struttura hardware molto curata e non sono comuni.

Maggiori informazioni su questa modalità sono descritti nella documentazione linkata alla fine di questo tutorial.

Indirizzo a 10 bit

Il sistema degli indirizzi si rende necessario proprio per la natura del bus; abbiamo visto che lo shift register tipico di I^2C ha una capacità di 8 bit e questo limita la dimensione di un indirizzo a soli 7 bit.
E l' invio di un pacchetto di 7 bit rende possibile l' accesso a 128 dispositivi. Si tratta comunque di quantità teorica, dato che nella realtà esistono due limiti: l' estensione fisica e le caratteristiche elettriche del bus e, come abbiamo visto, il fatto che vari indirizzi sono utilizzati per funzioni speciali.

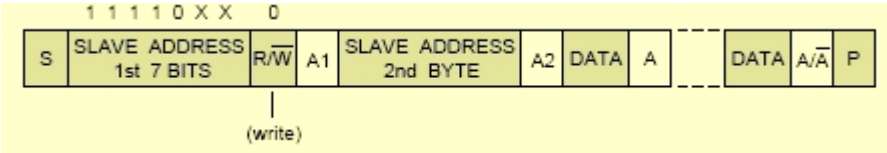
Lo standard, però, ha previsto la possibilità di estendere il numero delle periferiche indirizzabili a 1024 utilizzando indirizzi a 10 bit. E' evidente che, disponendo solo di pacchetti a 8 bit, occorrano due emissioni successive per completare l' indirizzo a 10 bit.
Per distinguere questa sequenza di due byte di indirizzo da una qualsiasi altra trasmissione di dati, il primo byte di indirizzo dovrà essere 1111 0xx.
I due bit meno significativi costituiscono i due bit più alti dell' indirizzo a 10 bit; il successivo byte completerà questo indirizzo. L' ottavo bit del primo byte è sempre riservato all' indicazione R/W.

Dispositivi con indirizzi 7 e 10 bit possono essere collegati allo stesso bus.
Ne consegue che non ci potranno essere dispositivi con indirizzo a 7 bit nell' area 1111000-1111011 (78h-7Bh).

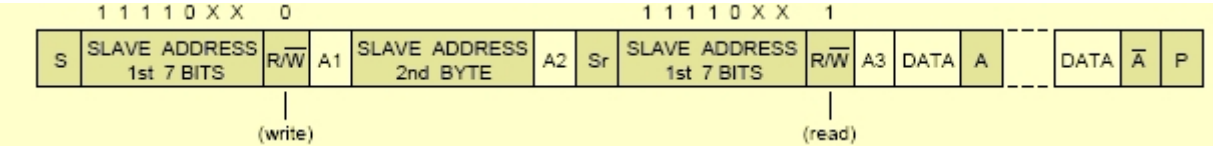
Attualmente, l' indirizzamento a 10-bit non viene ampiamente utilizzato, ma, in generale, i moduli di comunicazione sincrona dei microcontroller prevedono anche questa opzione.

Tutte le combinazioni di formati di lettura/scrittura sono possibili si con indirizzamento a 7 che a 10 bit. Vediamone due:

- Il Master-trasmettitore trasmette al ricevitore Slave con un indirizzo a 10 bit. La direzione di trasferimento è unica. Quando un indirizzo 10 bit segue una condizione di Start, ogni periferica confronta i primi sette bit del primo byte dell'indirizzo (1111 0XX) con il proprio indirizzo e verifica se l'ottavo bit (R/W) è 0. È possibile che più di un dispositivo trovi questa coincidenza e generi un acknowledge (A1). Ora i dispositivi confronteranno gli otto bit del secondo byte dell'indirizzo con i propri indirizzi, ma solo uno potrà trovare una corrispondenza e generare il secondo acknowledge (A2). Lo Slave corrispondente rimarrà in comunicazione con il Master fino a che riceve una condizione di STOP (P) o una condizione RESTART (Sr), seguita da un indirizzo diverso.



- Il Master-ricevitore legge dati da uno Slave-trasmettitore con un indirizzo a 10 bit. La direzione di trasferimento è cambiata dopo il secondo bit R/W. La procedura è la stessa di quella descritta prima. Dopo il Restart (Sr), una periferica corrispondente all' indirizzo si ricorda che esso è stato inviato prima, quindi controlla se i primi sette bit del primo byte dell' indirizzo seguito a Sr sono gli stessi, come se si trattasse di un indirizzo dopo la condizione di Start (S) e verifica se l'ottavo bit (R/W) è 1. Se esiste una corrispondenza, lo Slave genera riconoscere un ACK (A3). Il trasmettitore Slave rimane in connessione fino a quando riceve una condizione di STOP (P) o fino a quando non riceve un altro Restart (Sr) seguito da un indirizzo diverso. Dopo il Restart, tutti gli altri dispositivi confrontano i primi sette bit del primo byte dell'indirizzo (1111 0XX) con i propri indirizzi e testano l'ottavo bit (R/W). Tuttavia, nessuno di loro entrerà in linea perché R/W = 1 (per periferiche a 10-bit) o per l'indirizzo 0XX 1111 (per le periferiche di 7 bit) non corrisponde.



Il fatto che più periferiche mandino a livello basso la line SDA al momento della generazione del primo ACK (A1) non crea alcun problema al bus: ricordiamo che si tratta di un open collector-open drain e quindi qualunque dispositivo può mandare a livello basso la linea. Con la linea a livello basso sul nono colpo di clock, il Master rileverà un ACK e procederà con l' invio del secondo elemento dell' indirizzo, alla cui verifica risponderà questa volta una sola periferica (in quanto è obbligo che non esistano sul bus due elementi con lo stesso indirizzo completo).

In un bus misto di periferiche a 7 e 10 bit, quelle a 7 bit non entreranno mai in comunicazione quando un Master invia un indirizzo a 10 bit dato che il primo byte non corrisponde ad alcuno dei loro possibili indirizzi. Per contro, nessuna periferica a 10 bit potrà rispondere ad una chiamata con indirizzo a 7 bit, dato che per attivarsi deve ricevere il primo byte di indirizzo speciale.

Uno dispositivo con indirizzo a 10 bit, però, se abilitato, reagirà a una chiamata generale nello stesso modo dei dispositivi a 7 bit, in quanto per attivare questa funzione bastano i sette bit dell' indirizzo di General Call. Master hardware sono in grado di trasmettere il loro indirizzo a 10 bit dopo la General Call, con l' invio di due bytes, dove il primo byte di dati contiene gli otto bit meno significativi dell'indirizzo.

ID

I dispositivi I^2C contengono opzionalmente 3 bytes (24 bit) di identificazione del componente:

- 12 bit costituiscono l' identificatore del costruttore
- 9 bit definiscono l' identificatore del componente
- 3 bit definiscono la revisione assegnata dal costruttore

Con l' indirizzo riservato 11111000 è possibile accedere alla lettura di questo identificatore.
La funzione è descritta nella documentazione del bus di NXP.

L' identificatore del costruttore viene assegnato dal comitato di gestione del bus I^2C.

Funzioni particolari

La manipolazione delle linee **SDA** e **SCL** consente di introdurre alcune condizioni speciali che aumentano la flessibilità e l' efficienza del bus. Sono:

- **clock stretching**
- **clock synchronization**
- **arbitraggio**

CLOCK STRETCHING

Una delle caratteristiche più significative del protocollo è il clock stretching (letteralmente "allungamento del clock"). Questa azione ha lo scopo di dare ai dispositivi "lenti" abbastanza tempo per rispondere.

Se un dispositivo è lento, è possibile che occorra un certo tempo prima di aver elaborato il dato ricevuto; in questo caso la **periferica Slave** è autorizzata a portare a **livello basso la linea SCL** e a mantenerla a questo livello fino a che è necessario, indicando che non è ancora pronta per elaborare i altri dati.
Da parte sua **il Master**, rilasciando la linea del clock, **dovrà verificare** che essa ritorni effettivamente a livello alto per via dei pull-up; se questo non succede è indice che lo Slave sta chiedendo più tempo. Ne deriva che il **Master arresta le operazioni** fino a che SCL non viene rilasciata (questo è possibile perchè, ricordiamo, le linee del bus sono open drain - open collector e vanno a livello alto attraverso i pull-up).

Anche se il Master può tenere la linea SCL a livello basso come desidera, il termine "clock stretching" è normalmente utilizzato solo quando l' azione è eseguita da uno Slave.
Il clock stretching è l' unico momento in cui uno Slave può manipolare la linea SCL.
Se su un bus nessun dispositivo può richiedere il clock stretching, il Master non ha necessità di controllare questa funzione.

A livello di bit, in teoria qualsiasi impulso di clock può essere allungato, come nel caso in cui il clock del Master sia troppo rapido per lo Slave. A livello di byte, un dispositivo può essere in grado di ricevere un dato, ma richiedere tempo per elaborarlo e quindi operare un clock stretching dopo l' ACK con lo scopo di arrestare la trasmissione del Master per il tempo necessario. In modo High Speed il clock stretching è ammesso solo a livello di byte.

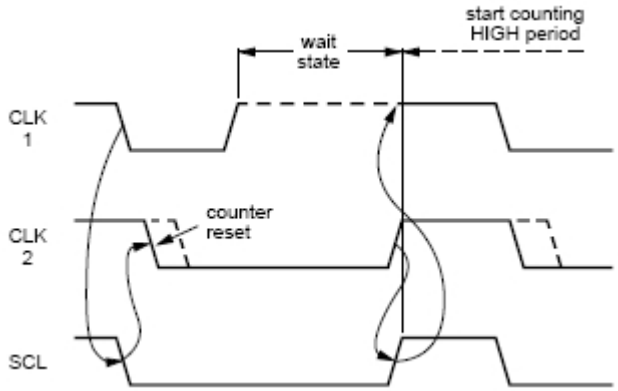
Alcuni dispositivi non hanno la possibilità di supportare il clock stretching e potranno essere indicati come "2-wires interface" e non come I²C puri. Da notare che, per garantire un transfer rate minimo, SMBus pone dei limiti alla lunghezza del clock e i dispositivi non possono bloccare il bus oltre un certo tempo.

Clock synchronization

Due Master possono iniziare a trasmettere contemporaneamente sul bus inattivo. E' evidente che questa circostanza è inaccettabile, in quanto la trasmissione dei dati sarebbe corrotta dall' accesso contemporaneo di due unità che cercano di inviare bit differenti. Occorre, quindi, un sistema di arbitraggio per decidere quale unità prenderà il controllo del bus e completerà la sua trasmissione.
Questa operazione è svolta dal clock synchronization e dall'arbitraggio.
Ovviamente se sul bus è presente un solo Master, queste funzioni non sono necessarie.

La sincronizzazione del clock si basa sulla linea SCL che è un wired-AND di open collector-open drain.
Tutti i Master generano il proprio clock e i dati sono validi durante il periodo di livello alto del clock.
E' possibile che i clock di diversi Master siano differenti ed è necessario per l' arbitraggio, che procede bit per bit, che i clock siano uguali.

Il diagramma rende graficamente la situazione.



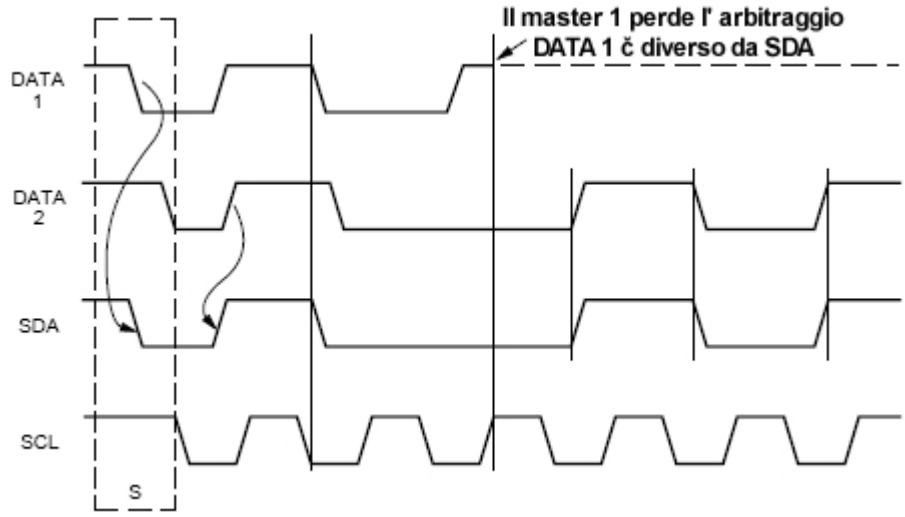
La transizione da livello alto a livello basso della linea SCL causerà da parte dei Master un conteggio basato sul proprio periodo di clock. Se il Master rilascia SCL, ma la linea non ritorna a livello alto, questo significa che un altro Master la sta impegnando. Non si può cambiare lo stato della linea di SCL se un altro clock la sta tenendo a livello basso; quindi SCL resterà a livello basso per il periodo del clock più lento, mentre il Master con il periodo più breve porterà il suo switch della linea SCL ad alta impedenza e aspetterà in wait state.

Quando tutti i master interessati hanno esaurito il loro conteggio del periodo a livello basso, la linea di clock sarà rilasciata e andrà a livello alto. Non ci sarà quindi alcuna differenza tra i clock dei Master e lo stato della linea SCL; tutti i Master inizieranno a contare i loro periodi di livello alto nello stesso momento. Il primo Master che completa il suo periodo, costringerà nuovamente la linea SCL basso. In questo modo viene generato un clock SCL sincronizzato nel periodo a livello basso, determinato dal Master con il più lungo periodo di livello basso e il periodo a livello alto determinato da quello con il più breve periodo a livello alto.

Arbitraggio

L' arbitraggio, come la sincronizzazione del clock, si riferisce ad una porzione del protocollo richiesto solo se verrà utilizzato più di un Master nel sistema. Gli **Slave non sono coinvolti** in queste procedure.

Un Master può iniziare un trasferimento solo se il bus è libero, ovvero entrambe le linee a livello alto, sostenute dai pull-up. Il Master che rileva il bus libero genererà una condizione di Start entro il tempo minimo di attesa stabilito dalle specifiche (tHD; STA); però è possibile che due Master cerchino di accedere al bus nello stesso momento: l'arbitraggio è quindi necessario per determinare quale Master prenderà il controllo. L' arbitraggio procede bit per bit: durante ogni bit, mentre SCL è a livello alto, ogni Master controlla il livello di SDA per vedere se corrisponde a quello che ha inviato. Questo processo potrebbe richiedere un certo numero di bit: i Master sono effettivamente in grado di completare un'intera transazione senza errore, fintanto che il dato trasmesso è identico, ma se il Master 1 tenta di inviare un livello alto mentre l' altro invia un livello basso, sa che ha perso l'arbitrato e disconnette il driver di uscita SDA, mentre il Master 2 va a completare la transazione. Nessuna informazione viene persa durante il processo di arbitraggio.



Un Master che perde l'arbitraggio può generare impulsi di clock fino alla fine del byte in cui perde l'arbitraggio e dovrà riavviare la transazione quando il bus sarà libero.
Se un Master incorpora anche una funzione di Slave e perde l' arbitraggio durante la fase di indirizzamento, è possibile che il Master vincente stia cercando di indirizzarlo; il Master perdente deve quindi passare immediatamente alla sua modalità di Slave

Siccome il controllo del bus è deciso esclusivamente sull'indirizzo e i dati inviati dai Master concorrenti, il protocollo non prevede alcun Master a livello superiore, né alcun ordine di priorità sul bus.

Di conseguenza si potrebbero verificare condizioni non definite nelle seguenti situazioni:

- Un Master invia una condizione di Restart mentre un'altro sta inviando bit di dati.
- Un Master invia una condizione di STOP e l' altro sta inviando dati.
- Un Master invia una condizione di Restart e l' altro invia uno Stop

Auto incremento

Dispositivi I2C dotati di un'area dati di dimensioni consistenti, come RAM o EEPROM o Flash, implementano un automatismo per cui il dispositivo mantiene un puntatore interno che viene incrementato automaticamente su dati da leggere o scrivere; questa funzione che può essere impostato manualmente oppure interamente automatica. Questo è utile per acedere ad una succesione di dati in modo ordinato ed efficiente.
L' incremento automatico non è parte dello standard I2C, ma è una caratteristica comune in molti dispositivi I2C.
Una tipica sessione di comunicazione è questa:

- Il Master invia una condizione di Start
- Segue un byte dell'indirizzo dello Slave con il bit di lettura/scrittura in modo da impostare il il puntatore dell'auto incremento
- Se la funzione è supportata, il Master invia il dato di auto incremento
- Il Master invia una condizione di Restart per leggere dati a partire da posizione iniziale dell' area puntata
- Il Master invia un byte dell'indirizzo a cui accedere
- e legge/scrive un numero arbitrario di byte dal dispositivo: la caratteristica auto incremento fa si che il puntatore interno, ad ogni accesso, sia modificato, permettendo di accedere a byte consecutivi senza ulteriori invii di indirizzo
- Terminato il trasferimento, il Master rilascia il bus con una condizione di Stop

Ovviamente sarà necessario consultare i fogli dati dei singoli dispositivi per verificare la possibilità di auto incremento e i le relative operazioni necessarie.

Rilevamento degli elementi sul bus e auto configurazione

Utilizzando, ad esempio, il byte di Start è possibile implementare sistemi di identificazione delle periferiche sul bus e modalità di auto configurazione del sistema molto complessi. La sequenza tipica è

- < Indirizzo primario > + W
- <byte di Start 00000001>
- < Indirizzo secondario >
- < DeviceType> < Sensor String > < Indirizzo Sensore >

Sono classificabili i DeviceType, che contiene la descrizione della classe di appartenenza del dispositivo (ad esempio Sensori, battery manager, dispositivi di comunicazione, GPS, terminatori, ecc) e i SensorString, che contengono le specifiche del sensore (temperatura, pressione, velocità, umidità, ecc) in modo da permettere al un Master di configurazione di operare una configurazione dinamica del bus

L' hardware del bus

Va osservato che **la Vdd del bus non è specificata come valore**, ma solo attraverso i parametri

- livello alto $\mathbf{V_{IH}}$ **= 0.7 Vdd** e
- livello baso $\mathbf{V_{IL}}$ **= 0.3 Vdd**,

per cui, se normalmente il bus utilizza la Vdd di alimentazione dei componenti digitali (da 2.5 a 5 V), nulla vieta che essa sia maggiore, ad esempio 15 V.
Il limite della tensione Vdd dipenderà dalle caratteristiche dei dispositivi e dalla corrente che possono commutare. La specifica indica i limiti 3 mA

per il modo standard e 20 mA per il modo Fast, ma molti dispositivi possono trattare anche correnti maggiori.
La corrente sarà limitata dal valore dei pull-up:

$$R_{p\ (min)} = [Vdd - V_{OL(max)}] / I_{OL}$$

La necessità dei pull-up rende pure necessario adeguarne il valore alla capacità della linea, abbassandone il valore con l' aumentare della frequenza di comunicazione. I valori tipici vanno da 2 k per clock a 400 kHz, fino a 10 k per clock inferiori a 100 kHz. **Microchip**, nel documento ***Getting Started:I2C Master Mode*** consiglia:

<100 kHz	**100 kHz**	**400 kHz**
4k7	2k2	1k

Va notato che, con il ridursi del valore dei pull-up, aumenta la corrente quando i dispositivi comunicano, il che rende il bus I^2C non ideale per applicazioni a basso consumo.

In ogni caso, la documentazione di NXP fornisce indicazioni per il calcolo del valore dei pull-up.
Considerando che il livello alto è definito come $V_{IH} = 0.7\ Vdd$ e il livello baso come $V_{IL} = 0.3\ Vdd$, occorre conoscere il valore della capacità C_b della connessione per determinare il valore della costante RC di capacità e resistenza del pull-up R_p.
Il valore dei pull-up sarà dato da:

$$R_p\ (max) = t_r / 0.8473\ C_b$$

dove t_r è il tempo minimo di salita della commutazione, dipendente dal clock e rilevabile dalle tabelle fornite dal costruttore. Come ordine di grandezza, ecco dei valori approssimati in dipendenza dalla frequenza del clock:

in funzione della capacità

Cp [pF]	**Rp** [kohm]		
	standard	**fast**	**fast plus**
100	12	4	1.5
200	6	2	0.5
300	4	1	0.5
400	3	1	0.5

in funzione della tensione

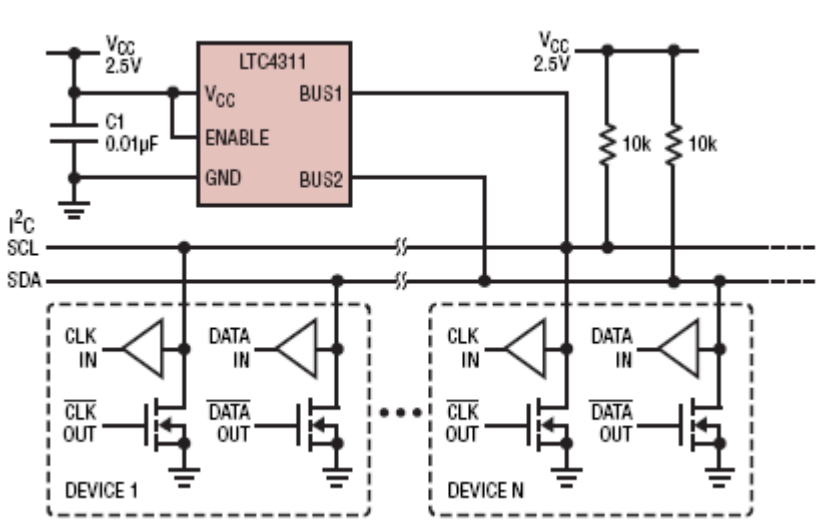
Vdd [V]	**Rp** [kohm]	
	standard e **fast**	**fast plus**
2.5	0.5	0.2
5	1.5	0.3
10	2.7	0.4
15	3.5	0.5

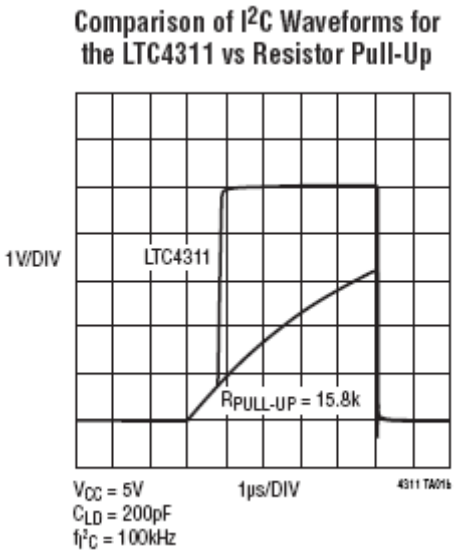
I valori sopra riportati sono puramente indicativi, tratti dai diagrammi forniti nella documentazione di NXP.

Se la frequenza e/o la capacità di linea aumentano, il bus richiederà dei pull-up attivi, ovvero dei generatori di corrente, in modo da avere una compensazione del carico capacitivo e quindi una buona qualità dei fronti di commutazione.
Ci sono vari metodi base per implementare questo:

- uno specchio di corrente (current mirror)
- una resistenza ed un diodo di calmping a partire da una tensione più elevata
- una generatore di corrente costante a FET o un diodo a corrente costante bipolare
- un integrato specifico, ad esempio LTC4311 di Linear Technology

Qui sotto, come esempio, l' applicazione di LT4311, presa dal foglio dati del componente, e, a lato, la forma d' onda della commutazione sulla linea in cui si nota chiaramente l' effetto positivo del generatore di corrente rispetto al solo pull-up.



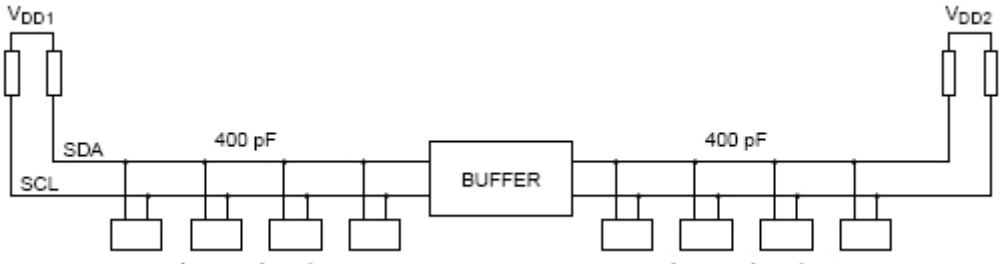


Per quanto riguarda i collegamenti del bus, occorre essenzialmente che le capacità parassite sia contenute entro un valore tale da non creare problemi ai fronti di commutazione. Anche se il nome dello standard fa riferimento a connessioni tra circuiti integrati, quindi all' interno dello stesso circuito stampato, sono stati realizzati buffer per trasportare la connessione I2C a molti metri di distanza, come il repeater P82B96 che consentirebbe di arrivare a 1000 m, ma con la riduzione del clock ad una trentina di kHz. Tipicamente il limite di capacità della connessione è 400 pF, il che consente di superare vari metri utilizzando cavi che non abbiano capacità superiore a 40-50 pF/m (Belden, cavi STP, ecc). Però, come è evidente, la frequenza massima a cui la trasmissione è possibile sarà in funzione non solo delle caratteristiche dei dispositivi collegati, ma anche di quelle del collegamento e,
come per ogni altro caso di bus remotati, la trasmissione è possibile con sicurezza in relazione alla frequenza di trasmissione, alle caratteristiche dei cavi, al rumore indotto dall' ambiente e, dato che si tratta di una connessione sigle-ended, ai ground bounces.. Ne deriva che, pur essendo possibile remotare elementi di un bus I2C anche a distanze considerevoli, sarà più sensato dare la preferenza ad altri tipi di interfaccia previsti proprio per questa situazione (CAN, RS-422/485, Ethernet, ecc).
All' atto pratico, al di fuori del circuito stampato, è consigliabile usare:

- flat cables fino a qualche decina di centimetri
- cavi piatti twistati (dati+gnd e clock+gnd) per distanze dell' ordine del metro
- cavi schermati e twistati per lunghezze maggiori

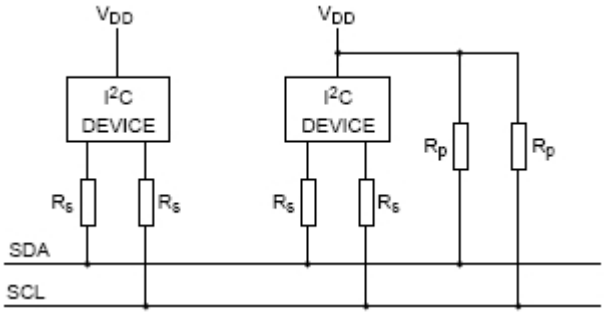
mantenendo le alimentazioni separate e ben disaccoppiate ad ogni periferica. Peraltro lo standard non specifica particolari cavi o connettori, per cui ogni costruttore è ragionevolmente libero nella scelta dei componenti da utilizzare. La stessa documentazione di Philips, però, consiglia di usare connessioni a 4 fili intrecciati; i conduttori dei segnali (clock e dati) non vanno intrecciati tra di loro per limitare la mutua induzione, o crosstalk, ma preferenzialmente sono associati ad un cavo di massa (gnd) con lo scopo di mantenere costante l' impedenza e di minimizzare il crosstalk.

Nel caso in cui le capacità parassite siano superiori al limite di 400 pF, si potrà suddividere il bus in sezioni attraverso un buffer o dei multiplexer.



Come nel caso esemplificato (o in quello più complesso presentato all' inizio del tutorial). Tra l' altro, una simile configurazione permette l' impiego di due Vdd differenti.
Sempre nel caso di Vdd differenti, sono disponibili traslatori di livello per adeguare le periferiche sullo steso bus.

NXP ha realizzato numerosi dispositivi per il supporto hardware al bus e sono virtualmente coperte tutte le possibili applicazioni. Le application notes di NXP ***AN255, I2C / SMBus Repeaters, Hubs and Expanders*** e ***AN262, PCA954x Family of I2C / SMBus Multiplexers and Switches*** illustrano queste possibilità.



Una ulteriore nota è utile dare riguardo alla possibilità di proteggere i dispositivi sul bus da possibili sovratensioni, inserendo in serie delle resistenze.

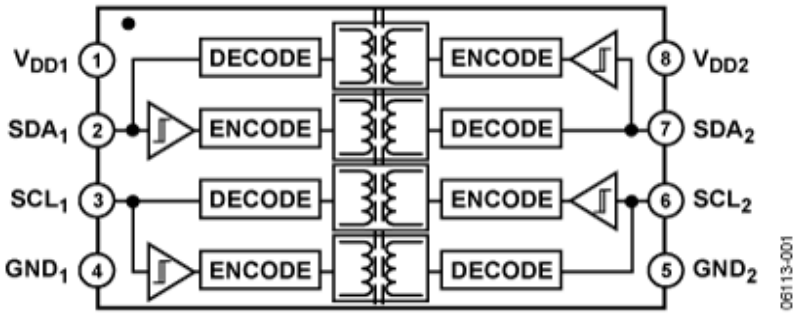
L' aggiunta delle resistenze Rs richiede un ricalcolo dei pull-up.

La documentazione di NXP fornisce informazioni anche relativamente a questo aspetto.

Come ordine di grandezza per le resistenze serie:

Rp	**Rs** (ohm)	
	Vdd = 2.5V	**Vdd = 5 V**
10 k	500	1k5
4 k	200	600
2 k	100	300
1 k	75	100

Esistono comunque dispositivi che integrano già sistemi di protezione contro sovratensioni.



In applicazioni critiche è indispensabile aggiungere un sistema di isolamento completo tra unità a microcontroller e periferiche remote: questo evita i potenzialmente dannosi loop di terra, i rischi per l' operatore e i disturbi raccolti dalle linee.

Un esempio è rappresentato da un componente specifico di **Analog Devices**.

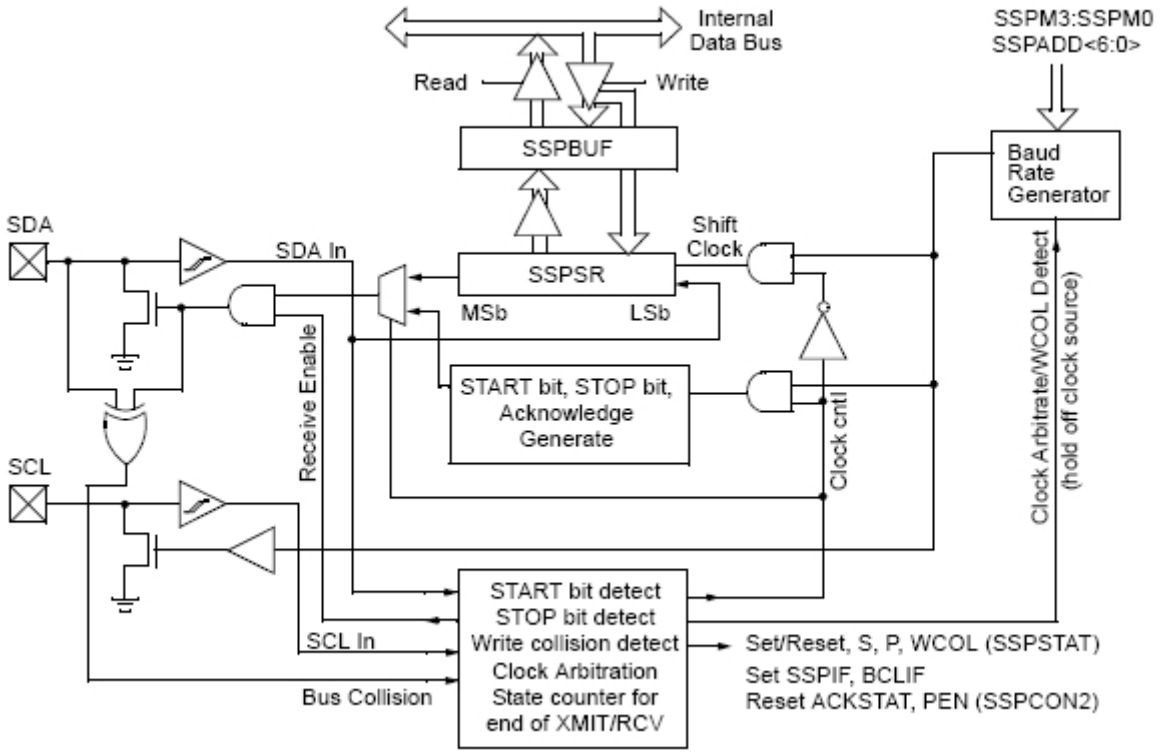
Si tratta di isolatore galvanico ADUM1250, hot swappable a due canali (due bus I2C oppure PBUS o SMBUS). Può operare fino a 1 MHz e con un isolamento di 2.5 kV.
Altri produttori di dispositivi analoghi sono **Silicon Labs** o **Linear technology**.
Un efficace isolamento può essere effettuato anche con opto isolatori.

I moduli di comunicazione sincrona

Un drive per shift register può essere scritto con semplicità per qualsiasi microcontroller, ma clock elevati e trasmissioni complesse come quella di I^2C costituiscono situazioni gravose da emulare in software.
In particolare risulta molto impegnativo realizzare gestioni che non siano in polling e del genere bit-banging.

Per evitare questo carico all' utente, nei principali microcontroller viene integrata una periferica specifica per la comunicazione sincrona, dotata di meccanismi di buffer e di flag per la gestione sia in polling che in interrupt. In genere questi moduli sono in grado di gestire diversi formati, ad esempio SPI, I^2C e SMBus.

Nel modulo di gestione seriale sincrona tipico di un microcontroller, per i PIC chiamato **MSSP** o **SSP** (*Master Synchronous Serial Peripheral* o *Synchronous Serial Peripheral*), sono presenti, oltre agli shift register (**SSPSR** nell' immagine sotto), anche dei buffer (**SSPBUF** nell' immagine) per conservare il dato ricevuto.



Possiamo notare la complessità che non si limita ad un solo registro di spostamento, ma ha come elemento essenziale una complessa logica per il riconoscimento delle condizioni e per l' arbitraggio del clock e le collisioni sul bus e la generazione delle corrette temporizzazioni dei segnali.
Osserviamo anche che, pur avendo gli I/O dei PIC una struttura push-pull (totem pole), i pin configurati per connettersi al bus I^2C sono programmati come open drain, con associato il buffer di ingresso a trigger di Schmitt.
I moduli integrati nei microcontroller ricalcano questa struttura, ma possono essere strutturati in modo differente, per cui occorrerà, prima di utilizzarli, documentarsi sui fogli informativi.

Va considerato che, se il rilevamento delle condizioni di Start e Stop, ACK e NACK sono eseguite hardware dai moduli di gestione delle comunicazioni sincrone, la gestione delle operazioni di trasmissione, delle conseguenze dell' arbitraggio, della gestione multi-master, di multiplexer o, comunque, delle periferiche, è **demandata al software che controlla il dispositivo**.

Quando si comunica con un altro dispositivo I^2C, gli 8 bit di dati possono essere un codice di controllo, un indirizzo o dati grezzi. Occorrerà verificare i manuali dei vari dispositivi per individuare le specifiche necessità: diverse periferiche potranno usare codici uguali con fini differenti e per ognuna si dovrà scrivere la struttura di controllo e comunicazione adeguata.

In questo senso, I2C può presentare un non indifferente livello di complicazione, ma, per contro, può dare origine a sistemi complessi dotati di prestazioni elevate.

Architetture basate su I^2C

Da quanto detto finora, si evidenzia che I^2C ha potenzialità notevoli per la creazione di sistemi articolati, con gestioni basate su messaggi, con possibilità di auto configurazione, hot swap, remote management, shutdown, restart e power control per la gestione intelligente di piattaforme hardware negli ambiti più svariati. Ad esempio:

- **Intelligent Platform Management Interface** , **IPMI** di Intel
- **Advanced Telecom Computing Architecture**, **ATCA**
- **Power Management Bus**, **PMBUS**
- **Access.Bus**

PRO I^2C

- Richiede solo due pin e due linee fisiche
- La selezione dei dispositivi non richiede chip select, ma indirizzi inviati sul bus
- Protocollo con handshake (condizioni) per controllare il flusso dei dati, lo stato delle periferiche e gli errori di trasmissione
- Struttura multi-Master
- Facile aggiunta di dispositivi, semplicemente "appesi" al bus
- Clock sincrono ai dati e quindi che non richiede elevata precisione
- Ampia disponibilità di accessori come buffer, multiplexer, remoters che consento di elaborare bus con topologie e funzioni molto complesse

CONTRO I^2C

- Trasmissione in half-duplex, appesantita dalla necessità di trasmettere indirizzi e dalle condizioni
- Trasferimento limitato ad 8 bit
- Bus con consumo anche sensibile, causato dai pull-up
- Velocità di trasmissione non elevata (400 kHz tipico)
- I dispositivi sono piuttosto complessi per supportare il protocollo
- I segnali sono bidirezionali: buffering e separazione galvanica sono meno semplici che nel caso di linee mono direzionali

Per un confronto con SPI:

	SPI	**I^2C**	**Note**
Topografia	master + slave	multi master + slave	I^2C è un protocollo multi-Master nativo
Conduttori	3-4 (+n)	2	I^2C richiede solo due conduttori, qualsiasi sia il numero delle periferiche
Trasmissione	full duplex	half duplex	SPI usa linee separate per trasmissione e ricezione e consente clock di frequenza maggiore
Velocità di trasmissione	alta	medio-bassa	
Periferiche utili	limitate	molte	In I^2C la quantità di Slave collegabili dipende dalla disponibilità di indirizzi
Selezione delle periferiche	chip select	indirizzi	
Protocollo	semplice	complesso	I^2C utilizza pacchetti con priorità, sistemi di arbitraggio del bus, indirizzi di selezione delle periferiche, ecc.
Emulabile software	facilmente	non semplice	SPI non ha un protocollo vero e proprio e si presta bene all' emulazione softwarei anche su microcontroller privi di moduli dedicati alla comunicazione sincrona

Gestione del clock	solo dal Master	da ogni Master	La linea di clock di I^2C è bidirezionale
Buffering delle linee	semplice	complesso	SPI ha linee mono direzionali. I^2C ha linee bi direzionali
Isolamento galvanico			

Come abbiamo visto, un tipico **modulo di comunicazione seriale sincrona** integrato in un microcontroller non è una opzione trascurabile, ma un elemento di grande utilità nella gestione delle comunicazioni sincrone; seppure esse siano emulabili da software, il modulo dedicato consente operazioni più semplici, gestioni sia in polling che in interrupt (indispensabili queste per applicazioni ad alta velocità o con più task attive) oltre alla possibilità di modificare con semplici comandi il modo di funzionamento e ottenere risultati che sarebbero assai difficili da ottenere da una emulazione.

Questa utilità è più evidente per I^2C, che presenta maggiori problematiche a causa del protocollo più complesso, ma anche in SPI offre soluzioni che una emulazione sarebbero quanto mai complicate, come il full duplex.

Si deve considerare però, che un modulo può gestire un solo modo di comunicazione per volta, dato che, in generale, fa capo agli stessi pin di I/O e che i due bus non sono compatibili tra di loro. Quindi, dovendo disporre di una comunicazione I^2C e SPI contemporaneamente, occorrerà scegliere un microcontroller con due o più moduli integrati. Anche se potrà essere fattibile a volte utilizzare il modulo integrato per I^2C ed emulare SPI via software.

In tutti i casi, la scelta di uno o dell' altro dipenderà in modo significativo dalle periferiche che si devono collegare, anche se attualmente il mercato offre una scelta molto ampia per cui si trovano sia in SPI che in I^2C molte funzioni analoghe.
Inoltre sarà scelto in tutti i casi in cui si voglia implementare un bus con prestazioni quali auto configurazione, hot swap, remote management, shutdown, restart, power control e, in generale, funzioni che il semplice protocollo di SPI non è in grado di supportare.