Versione N° 1.0

LaurTec

ATmega 2560

Corso di programmazione per microcontrollori Atmel

Autore : *Marcello Pinna*

ID: UT0005-IT

Copyright © 2021 Mauro Laurenti

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.
Tutti i diritti di questa opera sono riservati. Ogni riproduzione ed ogni altra forma di diffusione al pubblico dell'opera, o parte di essa, senza un'autorizzazione scritta dell'autore, rappresenta una violazione della legge che tutela il diritto d'autore, in particolare non ne è consentito un utilizzo per trarne profitto.
La mancata osservanza della legge 22 Aprile del 1941 n. 633 è perseguibile con la reclusione o sanzione pecuniaria, come descritto al Titolo III, Capo III, Sezione II.
A norma dell'art. 70 è comunque consentito, per scopi di critica o discussione, il riassunto e la citazione, accompagnati dalla menzione del titolo dell'opera e dal nome dell'autore.

AVVERTENZE

I progetti presentati non hanno la marcatura CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.
L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.
Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Tutti i marchi citati in quest'opera sono dei rispettivi proprietari.

Indice

Capitolo XII....4
Il modulo TWI....4
Descrizione del modulo TWI....5
Condizioni di Start, Repeated Start e Stop....7
Trasferimento di un bit in trasmissione e ricezione....8
Bit di Acknowledge (ACK)....8
Formato del pacchetto dell'indirizzo....9
Formato del pacchetto dati....9
Trasmissione combinata dei pacchetti....10
Utilizzo generale del modulo TWI....11
Software per il controllo del modulo TWI Master – Slave....11
Modalità di trasmissione e ricezione....13
Registri....19
Esempi di funzioni generiche principali....22
History....34

Capitolo XII

Il modulo TWI

Il modulo TWI (*Two Wires Interface*) è un'interfaccia seriale, compatibile con il protocollo I2C sviluppato dalla Phillips (ora NXP), in grado di indirizzare fino a 128 dispositivi, ognuno identificato da un proprio indirizzo univoco. Il protocollo utilizza solo due fili, uno per la linea di *clock* (SCL) e l'altro per la linea dati (SDA). Queste linee sono di tipo *open-collector/open-drain* e di conseguenza devono essere mantenute a livello alto (Vcc) usando un resistore di pull-up per linea. L'interfaccia può essere impostata in modo *Standard* (Frequenza SCL ≤ 100 KHz) oppure *Fast* (Frequenza SCL ≤ 400 KHz). Le sue caratteristiche principali, sono:

- Supporto modalità Master e Slave.
- 7 bit di spazio per ogni indirizzo.
- Supporto per interconnessione multi-Master.

In Figura 1 è mostrato un tipico collegamento tra sistemi che supportano il protocollo I2C. In particolare è mostrato il dettaglio dei resistori di *pull-up* esterni.

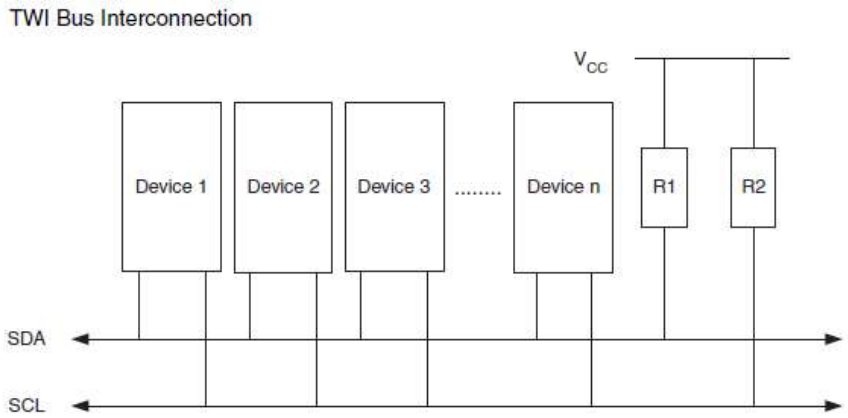

Figura 1: *Esempio di collegamento sul bus tra periferiche I2C.*

Una cosa che si deve tenere presente è che il numero dei dispositivi che possono essere interconnessi al bus varia in funzione del limite di capacità (400pF). In particolare la capacità del sistema viene a dipendere dal *layout* del PCB e dal numero di periferiche connesse sul bus stesso. Il bus TWI prevede l'utilizzo di più Master (multi-Master) che sono in grado di gestire entrambe le linee, mentre gli Slave possono gestire solo la linea dati.

Descrizione del modulo TWI

Per seguire la logica del *datasheet*, riporto le definizioni utilizzate:

TWI Terminology

Term	Description
Master	The device that initiates and terminates a transmission. The Master also generates the SCL clock
Slave	The device addressed by a Master
Transmitter	The device placing data on the bus
Receiver	The device reading data from the bus

In Figura 2 è mostrato il diagramma a blocchi relativo all'interfaccia TWI.

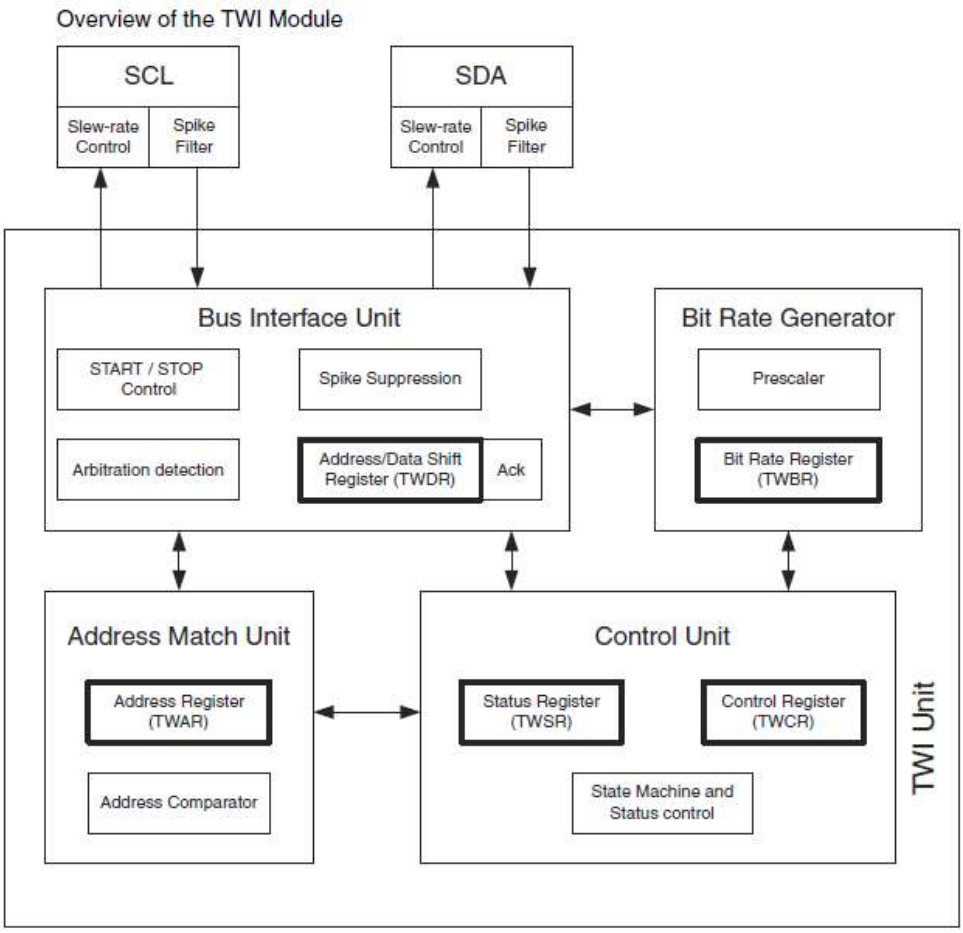

Figura 2: *Diagramma a blocchi del modulo TWI.*

In alto si trovano le due linee SCL e SDA, con il relativo hardware, che interfacciano il modulo con il resto del sistema. Entrambe le linee hanno in uscita un limitatore di velocità (*slew-rate*) al fine di rispettare le specifiche I2C, mentre in ingresso sono dotate di un soppressore di *spikes* che rimuove eventuali picchi di tensione inferiori a 50ns. Al di sotto delle linee si trova il *Bus Interface Unit*, l'unità di interfacciamento per il bus, che contiene il registro ad 8 bit TWDR (*Data and Address Shift Register*) dove vengono scritti gli indirizzi

e i dati da trasmettere o ricevere. Inoltre, questa unità contiene un altro registro nel quale viene scritto il bit ACK (o NACK) che deve essere inviato o ricevuto. In fase di ricezione questo bit può essere impostato o ripulito agendo sul registro TWDR. Sempre all'interno della stessa unità è presente il *controller Start-Stop*, responsabile della generazione e del rilevamento dei segnali di *Start*, *Repeated Start* e *Stop*. Sulla destra del diagramma si trova il *Bit Rate Generator*, che controlla il periodo della linea SCL in modalità Master. Questo periodo è impostato nel registro TWBR (*Bit Rate Register*) mentre il *prescaler* è impostato nel registro TWSR (*Status Register*). Le operazioni sui dispositivi Slave non dipendono da questi due parametri, tuttavia la frequenza di clock della CPU Slave deve essere almeno 16 volte più grande di quella della linea SCL. La frequenza di clock SCL è calcolata con la seguente formula:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

dove:

TWBR = valore di bit rate impostato nel registro TWBR
TWPS = valore di prescaler impostato nel registro TWPS

Generalmente, come assume anche il datasheet, i bit di prescaler TWPS1 e TWPS0 nel registro TWSR sono impostati sempre a 0 (prescaler = 1), con la conseguenza che $4^{TWPS} = 4^0 = 1$, numero ininfluente nel denominatore ai fini del calcolo finale. Infatti, per impostare una velocità di 100 KHz o 400 KHz, le velocità compatibili con il protocollo I2C, è sufficiente scrivere il giusto valore di bit rate nel registro TWBR. Per esempio, volendo calcolare il *bit rate* per ottenere una frequenza di clock uguale a 400KHz, basta applicare la seguente formula inversa:

TWBR = ((CPU *Clock frequency* / 400000) – 16) / 2

Con un quarzo, ovvero una CPU *Clock frequency*, uguale a 16000000 Hz si ha:

TWBR = ((16000000 / 400000) – 16) / 2

TWBR = (40-16) / 2

TWBR = 24 / 2 = 12

Tale valore corrispondente in binario a 0b00001100 oppure a 0x0C in esadecimale.

Il blocco *Address Match Unit* controlla che l'indirizzo ricevuto corrisponda a quello scritto nel registro TWAR (*Address Register*). L'abilitazione al controllo avviene tramite l'impostazione ad 1 del bit TWGCE (*General Call Recognition Enable*) nel registro TWAR. Una volta verificata la corrispondenza viene informata l'unità di controllo che effettua le azioni corrette. Da tenere presente che l'interfaccia TWI può servirsi o meno del bit di ACK in funzione dell'impostazione del registro TWCR. Il blocco *Control Unit* esegue continuamente il monitoraggio del bus TWI ed in base all'impostazione del registro TWCR genera le azioni richieste. Quando si verifica un evento rilevante sul bus TWI l'unità imposta ad 1 il flag di interrupt TWINT, mentre al successivo ciclo di clock aggiorna il registro TWSR con un codice di stato che identifica l'evento stesso. Questa identificazione può avvenire solo quando il flag TWINT è impostato ad 1, impostazione che allo stesso tempo fa in modo che la linea SCL sia tenuta a livello logico basso fino a quando lo stesso flag non viene azzerato, permettendo così al software di gestione di completare tutte le attività e di continuare la trasmissione. L'impostazione del flag avviene in queste situazioni:

- Dopo che il modulo TWI ha trasmesso una condizione di *Start* o *Repeated Start.*
- Dopo che il modulo TWI ha trasmesso uno SLA+RW (SLA *Slave Address*).
- Dopo che il modulo TWI ha trasmesso un byte di indirizzo.
- Dopo che il modulo ha trasmesso un byte di dati.
- Dopo che il modulo TWI ha perso l'arbitrato.
- Dopo che il modulo TWI è stato indirizzato dall'indirizzo del proprio Slave o da una chiamata generale.
- Dopo la ricezione di uno *Stop* o *Repeated Start* quando indirizzato come Slave (*Slave transmitter – Slave receiver*).
- Quando è avvenuto un errore generato da una condizione *Start* o *Stop* non consentita.

Condizioni di Start, Repeated Start e Stop

In Figura 3, sono riportati gli eventi di *Start*, *Stop* e *Repeated Start*, i quali sono associati al particolare livello delle linee SDA e SCL.

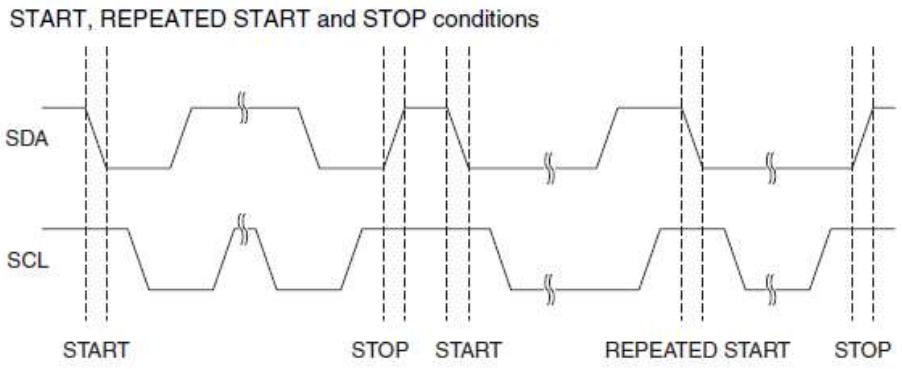

Figura 3: *Eventi di Start, Stop e Repeated Start.*

Innanzitutto si deve fare una premessa: il trasferimento dei dati è sempre attivato dal Master, inizia con una condizione di Start e termina con una condizione di Stop. In condizioni non operative le linee SDA e SCL sono sempre tenute a livello logico alto in quanto sono collegate a Vcc tramite le resistenze di *pull-up*. Durante la trasmissione dati, la linea SDA può cambiare il suo livello logico solo quando la linea SCL è a livello logico basso (GND). Se il segnale SDA cambia il suo livello logico mentre SCL è alto significa che il Master non sta trasmettendo o ricevendo un dato ma sta iniziando, ripetendo o terminando la comunicazione, cioè sta generando appunto le condizioni di *Start*, *Repeated Start* o di *Stop*. Lo *Start* ed il *Repeated Start* vengono generati dal Master portando la linea SDA a GND, mentre la linea SCL si trova a Vcc. Lo *Stop* viene generato portando la linea SDA nuovamente a Vcc mentre la linea SCL si trova a Vcc. Da tenere presente che lo *Start* ripetuto viene usato quando la condizione *Start* non è andata a buon fine, quando il modulo passa dalla modalità scrittura alla modalità lettura, oppure quando il Master desidera comunicare con un altro Slave. La sua condizione è identica a quella dello Start.

Trasferimento di un bit in trasmissione e ricezione

La trasmissione o la ricezione di un bit avviene sempre ad ogni impulso di clock e si deve sempre tenere presente che la linea SDA può cambiare il suo stato solo quando la linea SCL si trova a livello logico basso. Un esempio di trasmissione è riportato in Figura 4.

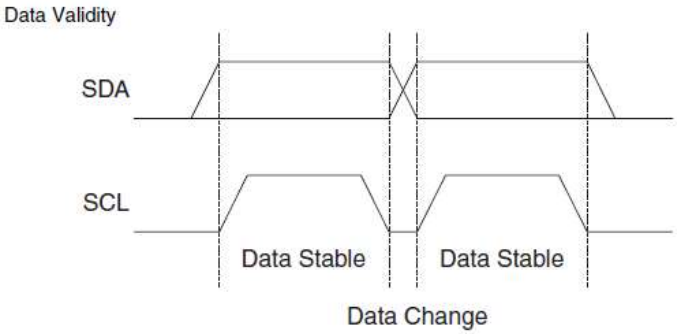

Figura 4: *Esempio di trasmissione bits.*

Subito dopo che il Master ha inviato la condizione Start, entrambe le linee sono a GND. In fase di trasmissione del dato, il Master carica il bit da trasmettere allo Slave sulla linea SDA. Poi porta a Vcc la linea SCL, trasmette il bit ed infine riporta quest'ultima a GND per caricare il bit successivo.

Bit di Acknowledge (ACK)

L'ACK è un bit di riconoscimento (*acknowledge*) che invia il dispositivo ricevente al dispositivo trasmittente per informarlo che la ricezione dei dati è avvenuta in modo corretto. Si supponga, per esempio, che il Master abbia inviato un byte ad uno Slave. Per

essere sicuro che il byte sia stato ricevuto correttamente, esso si metterà in attesa di ricevere e di leggere il bit di ACK portando la linea SDA a Vcc, mentre lo Slave trasmetterà l'ACK al nono ciclo di clock. Se la trasmissione è avvenuta correttamente lo Slave invierà l'ACK portando la linea SDA a livello logico basso, in caso contrario la manterrà a livello logico alto inviando così il bit di NACK (*Not ACK*). In caso di mancata ricezione del bit ACK il Master potrà inviare la condizione di *Stop* per terminare la comunicazione oppure inviare la condizione *Repeated Start* per iniziarne una nuova.

Formato del pacchetto dell'indirizzo

Ogni pacchetto di indirizzo trasmesso dal Master ad uno Slave è formato da 9 bit, i primi 7 bit compongono l'indirizzo univoco (formato a 7 bit) del dispositivo connesso, l'ottavo imposta l'operazione di lettura o scrittura (*Read-Write*) e l'ultimo è il bit di ACK (*acknowledge*). I bit più significativi dell'indirizzo (MSB) sono trasmessi per primi. Il bit *Read-Write* segue, e se è scritto a 0 viene eseguita un'operazione di scrittura, se scritto a 1 viene eseguita un'operazione di lettura. Quando uno Slave è libero e riconosce che è stato interpellato dal Master deve portare a GND la linea SDA al nono ciclo di clock (ACK). Viceversa, se risulta occupato e non può soddisfare la richiesta del Master, deve portare a Vcc la linea SDA al nono ciclo di clock ACK, permettendo così al Master di inviare il segnale di Stop e successivamente quello di Start ripetuto. In Figura 5 è riportato il relativo diagramma:

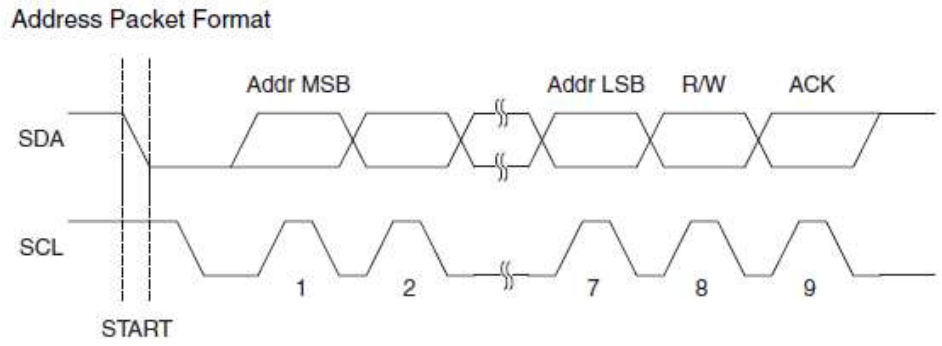

Figura 5: *Esempio di trasmissione dell'indirizzo.*

Formato del pacchetto dati

Ogni pacchetto dati trasmesso dal Master ad uno Slave è composto da 9 bit: i primi 8 bit (byte) compongono il dato mentre il nono è il bit di ACK. I bit più significativi del dato (MSB) vengono trasmessi per primi. Mentre il Master si occupa di generare il segnale di clock e le condizioni di Start e Stop, lo Slave si occupa esclusivamente della ricezione. Quando lo Slave riceve correttamente i dati segnala l'evento al Master trasmettendo un bit di ACK, ovvero portando a livello logico basso la linea SDA durante il nono ciclo di clock. Se la linea per qualche ragione resta a livello logico alto lo Slave ricevente segnala l'evento al Master trasmettendo un bit di NACK (*Not ACK*). Lo stesso segnale viene inviato quando

lo Slave ha ricevuto l'ultimo byte o per qualche ragione non può riceverne altri. In Figura 6 è riportato il relativo diagramma:

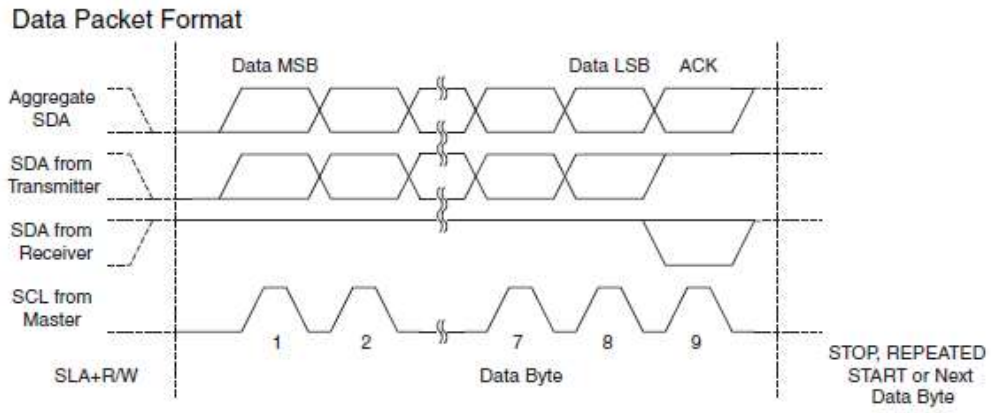

Figura 6: *Esempio di trasmissione dei dati.*

Trasmissione combinata dei pacchetti

Una trasmissione consiste essenzialmente nell'invio di una condizione di Start, di un indirizzo univoco dello Slave (SLA) comprensivo del bit *Read-Write* (SLA+RW), uno o più pacchetti dati ed una condizione di Stop. Come riportato nel datasheet, non è quindi possibile l'invio di un messaggio composto solo dai segnali di Start e Stop. In Figura 7 è riportato il relativo diagramma:

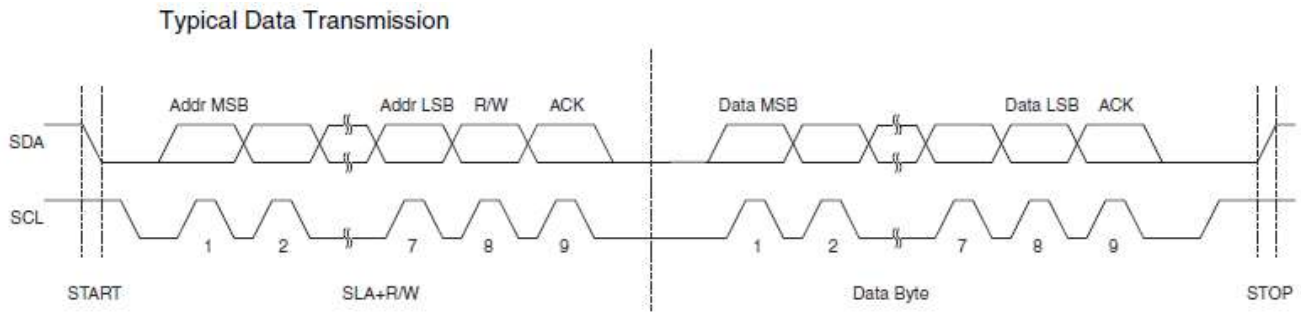

Figura 7: *Esempio di trasmissione I2C completa di indirizzo e dati.*

Una trasmissione tra Master e Slave prevede la seguente sequenza:

1. Invio della condizione Start da parte del Master.
2. Invio dell'indirizzo dello Slave (SLA) da parte del Master.
3. Invio del bit di scrittura (W) da parte del Master.
4. Invio del bit di ACK da parte dello Slave.
5. Ricezione del bit di ACK da parte del Master.
6. Invio del byte di dati da parte del Master.
7. Invio del bit di ACK da parte dello Slave.

8. Ricezione del bit di ACK da parte del Master.
9. Invio del bit di Stop da parte del Master.

I punti 5 e 6 devono essere ripetuti per scrivere o leggere più dati

Utilizzo generale del modulo TWI

Il modulo è orientato ai byte e sfrutta gli interrupt, che sono generati quando si verifica un evento sul bus, come la trasmissione di uno *Start* o la ricezione di un byte. L'interrupt è abilitato scrivendo ad 1 il bit TWIE (*Interrupt Enable*) nel registro TWCR e viene eseguito scrivendo ad 1 anche il bit I-bit (*Global Interrupt Enable*) nel registro SREG. Poiché il registro TWSR viene aggiornato con il codice di stato che identifica l'evento avvenuto, il software è in grado di decidere il comportamento del bus TWI, al successivo ciclo di clock, manipolando i registri TWCR e TWDR. Da tenere presente che se il bit TWIE viene resettato, il software deve continuamente verificare il suo stato per rilevare eventuali azioni avvenute sul bus.

Software per il controllo del modulo TWI Master – Slave

La Figura 8 descrive graficamente le istruzioni necessarie da scrivere nel programma per poter stabilire una semplice comunicazione tra un Master ed uno Slave.

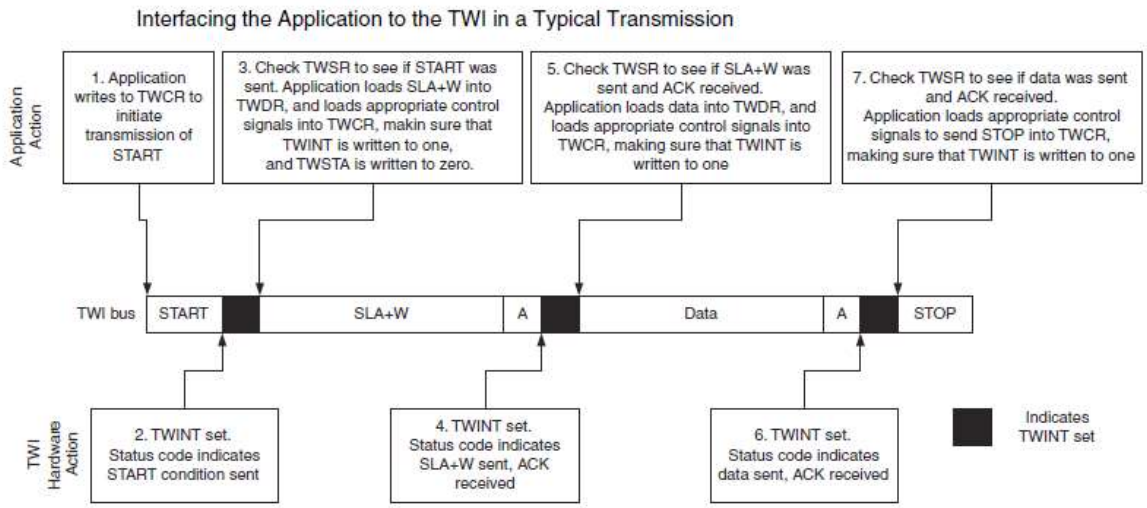

Figura 8: *Esempio di controllo del modulo.*

Queste istruzioni sono brevemente riassunte nella Tabella 1. In Tabella è riportata anche una traduzione in linguaggio C. Per ulteriori approfondimenti si rimanda alla lettura del *datasheet*.

	Istruzione	Esempio in C
1	Inviare la condizione di Start portando ad 1 i bit TWINT, TWSTA, TWEN nel registro TWCR. È importante che il flag TWINT sia resettato, in modo da far partire la trasmissione.	`TWCR=(1<<TWINT) | (1<<TWSTA) | (1<<TWEN);`
2	Attendere che il flag TWINT sia impostato, ovvero scritto a 0. Ciò indica che la condizione di Start è stata trasmessa	`while (!(TWCR & (1<<TWINT));`

3	Verificare il valore del registro di stato TWSR mascherando i bit di prescaler (*). Se il valore è diverso da Start eseguire la funzione ERROR.
	
	Caricare l'indirizzo dello Slave ed il bit di scrittura (SLA+W) nel registro TWDR. Ripulire il bit TWINT
		`if ((TWSR & 0xF8) != Start)`
		`ERROR();`
		
		`TWDR = SLA_W;`
		`TWCR = (1<<TWINT) | (1<<TWEN);`

4	Attendere che il flag TWINT sia impostato a 0. Ciò indica che SLA+W è stato trasmesso e che il bit ACK (NACK) è stato ricevuto.	`while (!(TWCR & (1<<TWINT));`

5	Verificare il valore del registro di stato TWSR mascherando i bit di prescaler (*). Se il valore è diverso da MT_SLA_ACK eseguire la funzione ERRORE.
	
	Caricare il pacchetto dati nel registro TWDR. Ripulire il bit TWINT per permettere la trasmissione del pacchetto
		`if ((TWSR & 0xF8)!=MT_SLA_ACK)`
		`ERROR();`
		
		`TWDR = DATA;`
		`TWCR = (1<<TWINT)|(1<<TWEN);`

6	Attendere che il flag TWINT sia impostato, ovvero scritto a 0. Ciò indica che il pacchetto dati è stato trasmesso	`while ( ! (TWCR & (1<<TWINT));`
7	Trasmettere la condizione di Stop	`TWCR=(1<<TWINT) | (1<<TWEN) | (1<<TWSTO);`

Tabella 1: *Istruzioni per il cotrollo del modulo TWI.*

Come si vedrà in seguito, il registro a 8 bit TWDR è formato da 5 bit (7-3) che contengono lo stato del registro, da un bit riservato (bit 2) e da due bit (bit 1-0) di *prescaler*. Per poter leggere lo stato del registro, ovvero il suo valore reale, è necessario mascherare i bit di prescaler utilizzando la *bitmask* 0xF8.

Modalità di trasmissione e ricezione

Il modulo TWI può operare in quattro modalità chiamate MT (*Master Transmitter*), MR (*Master Receiver*), ST (*Slave Transmitter*) e SR (*Slave Receiver*), dove:

- MT : un Master trasmette, controlla il clock e invia dati agli Slave.
- MR: un Master riceve, controlla il clock e riceve dati dallo Slave.
- ST: uno Slave trasmette, non controlla il clock e invia dati al Master.
- SR: uno Slave riceve, non controlla il clock e invia dati al Master.

Queste modalità possono essere arbitrariamente usate nella stessa applicazione, ad esempio un Master impostato inizialmente in MT può essere impostato successivamente in MR. Come già fatto presente, ogni modalità scelta porta comunque a generare degli eventi software e hardware, come quelli riportati in Tabella 1, che vengono segnalati con un codice di stato che viene scritto nel registro TWST.

Modalità MT (Master Transmitter)

L'impostazione del dispositivo come *Master Transmitter* viene effettuata inviando in sequenza una condizione *Start* ed il byte SLA+W, ovvero l'indirizzo univoco più l'ottavo bit impostato in scrittura. In Figura 9 è mostrato lo schema del sistema relativo alla modalità *Master Transmitter* mentre in Figura 10 è riportata la lista dei codici di stato generati.

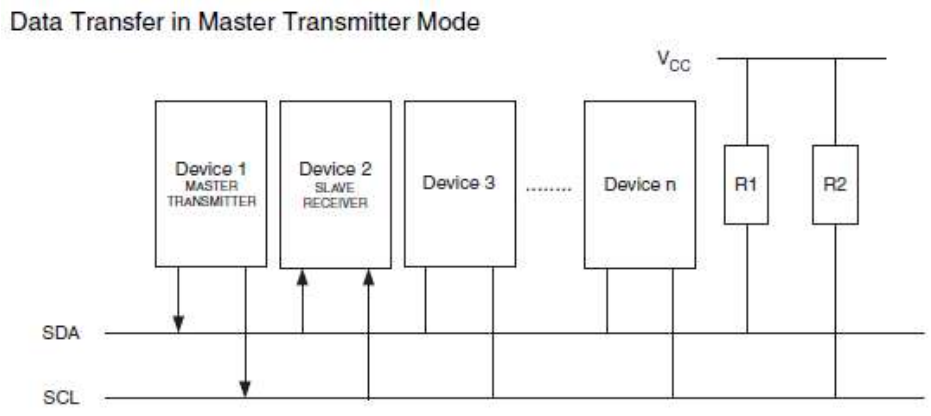

Figura 9: *Modalità Master Transmitter (MT).*

Status codes for Master Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+W	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+W or	0	0	1	X	SLA+W will be transmitted; ACK or NOT ACK will be received
		Load SLA+R	0	0	1	X	SLA+R will be transmitted; Logic will switch to Master Receiver mode
0x18	SLA+W has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x20	SLA+W has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x28	Data byte has been transmitted; ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x30	Data byte has been transmitted; NOT ACK has been received	Load data byte or	0	0	1	X	Data byte will be transmitted and ACK or NOT ACK will be received
		No TWDR action or	1	0	1	X	Repeated START will be transmitted
		No TWDR action or	0	1	1	X	STOP condition will be transmitted and TWSTO Flag will be reset
		No TWDR action	1	1	1	X	STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x38	Arbitration lost in SLA+W or data bytes	No TWDR action or	0	0	1	X	2-wire Serial Bus will be released and not addressed Slave mode entered
		No TWDR action	1	0	1	X	A START condition will be transmitted when the bus becomes free

Figura 10: *Codici di stato in modalità Master Transmitter (MT).*

Modalità MR (Master Receiver)

L'impostazione del dispositivo come *Master Receiver* viene effettuata inviando in sequenza una condizione *Start* ed il byte SLA+R, ovvero l'indirizzo univoco più l'ottavo bit impostato in lettura. In Figura 11 è mostrato lo schema del sistema relativo alla modalità *Master Receiver* mentre in Figura 12 è riportata la lista dei codici di stato generati.

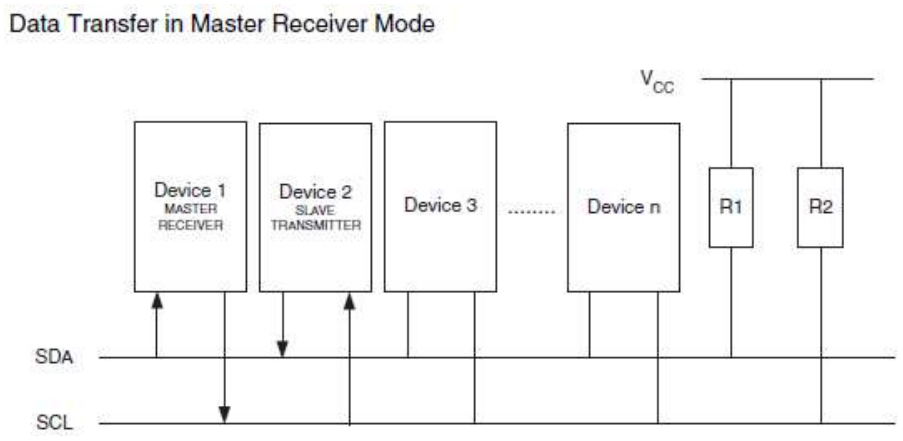

Figura 11: *Modalità Master Receiver (MR).*

Status codes for Master Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x08	A START condition has been transmitted	Load SLA+R	0	0	1	X	SLA+R will be transmitted ACK or NOT ACK will be received
0x10	A repeated START condition has been transmitted	Load SLA+R or Load SLA+W	0 0	0 0	1 1	X X	SLA+R will be transmitted ACK or NOT ACK will be received SLA+W will be transmitted Logic will switch to Master Transmitter mode
0x38	Arbitration lost in SLA+R or NOT ACK bit	No TWDR action or No TWDR action	0 1	0 0	1 1	X X	2-wire Serial Bus will be released and not addressed Slave mode will be entered A START condition will be transmitted when the bus becomes free
0x40	SLA+R has been transmitted; ACK has been received	No TWDR action or No TWDR action	0 0	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x48	SLA+R has been transmitted; NOT ACK has been received	No TWDR action or No TWDR action or No TWDR action	1 0 1	0 1 1	1 1 1	X X X	Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset
0x50	Data byte has been received; ACK has been returned	Read data byte or Read data byte	0 0	0 0	1 1	0 1	Data byte will be received and NOT ACK will be returned Data byte will be received and ACK will be returned
0x58	Data byte has been received; NOT ACK has been returned	Read data byte or Read data byte or Read data byte	1 0 1	0 1 1	1 1 1	X X X	Repeated START will be transmitted STOP condition will be transmitted and TWSTO Flag will be reset STOP condition followed by a START condition will be transmitted and TWSTO Flag will be reset

Figura 12: *Codici di stato in modalità Master Receiver (MR).*

Modalità ST (Slave Transmitter)

In questa modalità lo Slave trasmette byte dati al Master, il quale li deve avere precedentemente richiesti. In Figura 13 è mostrato lo schema del sistema relativo alla modalità *Slave Transmitter* mentre in Figura 14 è riportata la lista dei codici di stato generati.

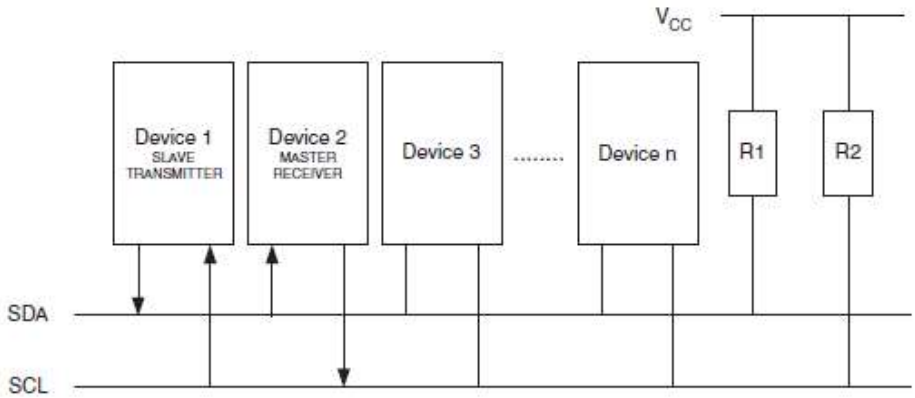

Figura 13: *Modalità Slave Trasmitter (ST).*

Status Codes for Slave Transmitter Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0xA8	Own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB0	Arbitration lost in SLA+R/W as Master; own SLA+R has been received; ACK has been returned	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xB8	Data byte in TWDR has been transmitted; ACK has been received	Load data byte or	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
0xC0	Data byte in TWDR has been transmitted; NOT ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
0xC8	Last data byte in TWDR has been transmitted (TWEA = "0"); ACK has been received	No TWDR action or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No TWDR action or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		No TWDR action or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No TWDR action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free

Figura 14: *Codici di stato in modalità Slave Transmitter (ST).*

Modalità SR (Slave Receiver)

In questa modalità lo Slave riceve un numero di byte dati dal Master. In Figura 15 è mostrato lo schema del sistema relativo alla modalità *Slave Receiver*, mentre in Figura 16 è riportata la lista dei codici di stato generati.

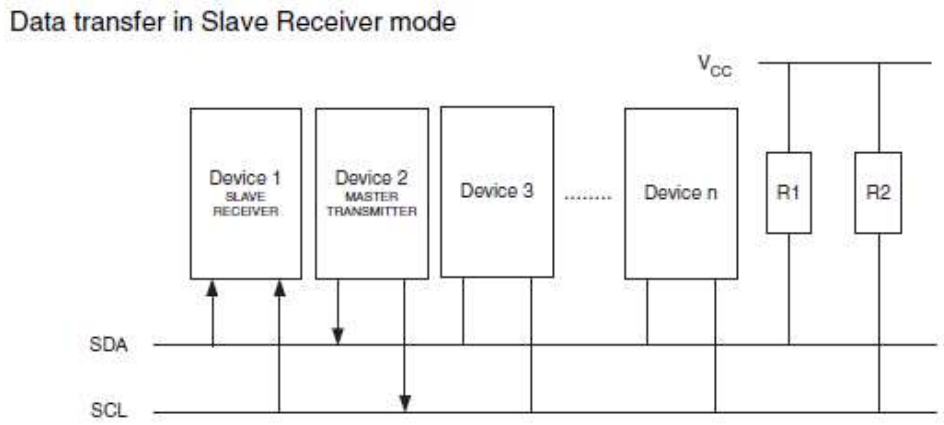

Figura 15: *Modalità Slave Receiver (SR).*

Status Codes for Slave Receiver Mode

Status Code (TWSR) Prescaler Bits are 0	Status of the 2-wire Serial Bus and 2-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
		To/from TWDR	To TWCR				
			STA	STO	TWINT	TWEA	
0x60	Own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x68	Arbitration lost in SLA+R/W as Master; own SLA+W has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x70	General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x78	Arbitration lost in SLA+R/W as Master; General call address has been received; ACK has been returned	No TWDR action or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		No TWDR action	X	0	1	1	Data byte will be received and ACK will be returned
0x80	Previously addressed with own SLA+W; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned
0x88	Previously addressed with own SLA+W; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
0x90	Previously addressed with general call; data has been received; ACK has been returned	Read data byte or	X	0	1	0	Data byte will be received and NOT ACK will be returned
		Read data byte	X	0	1	1	Data byte will be received and ACK will be returned

0x98	Previously addressed with general call; data has been received; NOT ACK has been returned	Read data byte or	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		Read data byte or	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
		Read data byte or	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		Read data byte	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
			0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"
			1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
			1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if TWGCE = "1"; a START condition will be transmitted when the bus becomes free

Figura 16: *Codici di stato in modalità Slave Receiver (SR).*

Registri

TWBR: TW Bit Rate Register

E' il primo registro ad 8 bit che viene descritto, in quanto la prima cosa logica che si deve fare per permettere un trasferimento dati è decidere la frequenza di trasmissione, ovvero la frequenza di clock SCL. In Figura 17 sono mostrati i bit contenuti nel registro.

TWBR – TWI Bit Rate Register

Bit	7	6	5	4	3	2	1	0	
(0xB8)	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 17: *Registro TWBR.*

I bit 7-0 **TWBR7 - TWBR0** (*TW Bit Rate*) selezionano il fattore di divisione per il generatore di bit rate, ovvero il divisore di frequenza che genera la frequenza di clock SCL quando il dispositivo è impostato come Master.

TWCR: TW Control Register

E' il registro usato per abilitare l'interfaccia TWI, per generare le condizioni di *Start* e *Stop*. In Figura 18 sono mostrati i bit contenuti nel registro.

TWCR – TWI Control Register

Bit	7	6	5	4	3	2	1	0	
(0xBC)	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	TWCR
Read/Write	R/W	R/W	R/W	R/W	R	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 18: *Registro TWCR.*

Il bit 7 **TWINT** (*TW Interrupt Flag*) è impostato a 1 dall'hardware quando il modulo ha finito il suo lavoro ed aspetta altre istruzioni dal software. Se questo bit viene scritto ad 1 insieme al bit TWIE (*TW Interrupt Enable*) ed al bit I-bit viene generato un interrupt. Quando questo bit è pari a 1, il periodo in cui la linea SCL sta a livello logico basso è allungato. Questo bit deve essere posto a 0 dal software prima di una condizione Start, Repeated Start o Stop.

Il bit 6 **TWEA** (*TW Enable Ackowledge*) se scritto ad 1, controlla la generazione dell'impulso di ACK, impulso che viene generato quando il dispositivo trova il suo proprio indirizzo, quando è stata ricevuta una chiamata generale (TWGCE = 1) o quando un byte

di dati è stato ricevuto in modalità MR o SR. Se il bit è scritto a 0 si ha la disconnessione temporanea dal bus seriale

Il bit 5 **TWSTA** (*TW Start Condition*), se scritto ad 1, imposta il dispositivo come Master, verifica che il bus sia libero e successivamente invia la condizione *Start*. Se il bus è occupato, il modulo TWI aspetta fino a quando non riceve la condizione di *Stop*, generando una nuova condizione *Start* per riprendere lo stato di Master.

Il bit 4 **TWSTO** (*TW Stop Condition*), se scritto ad 1, genera la condizione di *Stop*. Questo bit viene automaticamente resettato.

Il bit 2 **TWEN** (*TW Enable*), se scritto ad 1, abilita l'interfaccia TWI e prende il controllo sui pin connessi alle linee SDA e SCL, abilitando il limitatore della velocità di risposta ed il filtro anti-disturbo. Se questo bit viene posto a 0, il modulo TWI viene disattivato ed ogni operazione pendente viene terminata.

Il bit 0 **TWIE** (*TW Interrupt Enable*), se scritto ad 1, abilita la generazione di un interrupt.

TWSR: TW Status Register

Questo è il registro ad 8 bit del modulo TWI in cui vengono scritti i codici di stato ed il valore del prescaler. In Figura 19 sono mostrati i bit contenuti nel registro.

TWSR – TWI Status Register

Bit	7	6	5	4	3	2	1	0	
(0xB9)	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	TWSR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	1	1	1	1	1	0	0	0	

Figura 19: *Registro TWSR.*

I bit 7-3 **TWS7 - TWS3** sono le locazioni che compongono il vero registro di stato. Sarebbe come dire che il registro di stato è in realtà un registro a 5 bit. Per questo motivo la lettura dell'intero registro comporta necessariamente il mascheramento dei bit TWPS1 e TWPS0, che rappresentano il valore di prescaler. La *bitmask* utilizzata ha un valore uguale a 0b11111000, ovvero 0xF8

I bit 1-0 **TWPS1 - TWPS0**, come appena detto, rappresentano i valori di *prescaler*. La loro combinazione è riportata in Figura 20.

TWI Bit Rate Prescaler

TWPS1	TWPS0	Prescaler Value
0	0	1
0	1	4
1	0	16
1	1	64

Figura 20: *Valori del Prescaler.*

TWAR: TW Data Register

In modo trasmittente questo registro ad 8 bit contiene il byte successivo che deve essere trasmesso mentre in modo ricevente contiene l'ultimo byte ricevuto. E' scrivibile quando il modulo TWI non sta eseguendo un'operazione di *shifting* su un byte. In Figura 21 sono mostrati i bit contenuti nel registro.

TWDR – TWI Data Register

Bit	7	6	5	4	3	2	1	0	
(0xBB)	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0	TWDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	1	

Figura 21: *Registro TWDR.*

I bit 7-0 **TWD7 - TWD0** contengono, secondo la modalità in cui è impostata l'interfaccia, il byte successivo che deve essere trasmesso oppure l'ultimo byte che è stato ricevuto.

TWAR: TW Address Register

Questo registro ad 8 bit contiene l'indirizzo a 7 bit del dispositivo più il bit di chiamata generale. In Figura 22 sono mostrati i bit contenuti nel registro.

TWAR – TWI (Slave) Address Register

Bit	7	6	5	4	3	2	1	0	
(0xBA)	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	TWAR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	1	1	1	1	1	1	0	

Figura 22: *Registro TWAR.*

I bit **TWA6 - TWA0** sono le locazioni più significative in cui deve essere scritto l'indirizzo del dispositivo a cui il modulo risponderà quando è impostato come Slave Trasmittente (ST) o Slave Ricevente (SR). In modalità Master non è necessario il loro uso.

Il bit **TWGCE** (*TW General Call Enable*) è il bit meno significativo che, se scritto ad 1, abilita il riconoscimento di una chiamata generale inviata da un Master sul bus seriale.

TWAMR: TW Address Mask Register

E' un registro ad 8 bit in cui può essere caricato un valore che maschera l'indirizzo dello Slave. Ogni bit, se scritto ad 1, disabilita il bit corrispondente nel registro TWAR, bit che non verrà confrontato dal modulo *Address Match Unit* per verificarne la corrispondenza. In Figura 23 sono mostrati i bit contenuti nel registro.

TWAMR – TWI (Slave) Address Mask Register

Bit	7	6	5	4	3	2	1	0	
(0xBD)	TWAM[6:0]							–	TWAMR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	
Initial Value	0	0	0	0	0	0	0	0	

Figura 23: *Registro TWAMR.*

Esempi di funzioni generiche principali

Nei paragrafi seguenti vengono riportati alcuni esempi modificabili di funzioni generiche per il controllo del modulo TWI, impostato come Master Trasmettitore (MT).

La prima funzione che si deve scrivere nel firmware è quella di inizializzazione del modulo, ovvero quella che azzera il registro di stato TWSR e che definisce la frequenza di clock SCL. Si è visto in precedenza che per impostarla per una frequenza uguale a 400KHz, usando un quarzo da 16MHz, si deve scrivere il registro TWBR con il valore 12. Tuttavia, questo valore può essere calcolato automaticamente, definendo la frequenza della CPU e la frequenza di clock SCL

```
// Definisco la frequenza della CPU
#define F_CPU 16000000UL
```

```
// Definisco la frequenza di clock SCL in Hz
#define SCL_CLOCK_FREQUENCY 400000L
```

Funzione per l'inizializzazione del modulo TWI

```
void  TWI_Init (void){

 // Azzero il registro di stato
 TWSR = 0x00;

 // Imposto SCL clock frequency = 400 KHz
 TWBR = ((F_CPU / SCL_CLOCK_FREQUENCY) - 16) / 2;
}
```

Funzione per l'invio della condizione Start

```
void  TWI_Start (void){

 // Invio la condizione Start
 TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

 // Attendo che il flag di interrupt sia 1... Start trasmesso
 while ( ! (TWCR & (1<<TWINT));
}
```

Funzione per la verifica dello stato del registro TWSR

```
unsigned char  TWI_Get_Status (void){

 // variabile che contiene il valore del registro TWSR
 unsigned char status;

 // Maschero i bit di prescaler
 status = TWSR & 0xF8;

 // Ottengo lo stato del registro TWSR
 return status;
}
```

Funzione per la scrittura di un byte

```
void  TWI_Write_Byte (unsigned char data){

 // Carico il registro TWDR con il dato
 TWDR = data;

  // Pulisco il bit TWINT per avviare la trasmissione
  TWCR = (1<<TWINT) | (1<<TWEN);

 // Attendo che il flag di interrupt sia 1
 while (!(TWCR & (1<<TWINT)));
}
```

Funzione per l'invio della condizione Repeated Start

```
void TWI_Repeated_Start (void){

 // Invio la condizione Start
 TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);

 // Attendo che il flag di interrupt sia 1... Start trasmesso
 while ( ! (TWCR & (1<<TWINT));
}
```

Funzione per la lettura del byte con ACK

```
unsigned char TWI_Read_ACK (void){

 // Pulisco TWINT e TWEA
 TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);

 // Attendo che il flag di interrupt sia 1
 while ( ! (TWCR & (1<<TWINT));

 // Leggo TWDR
 return TWDR;
}
```

Funzione per la lettura del byte senza ACK (NACK)

```
unsigned char TWI_Read_NACK (void){

 // Pulisco TWINT
 TWCR = (1<<TWINT)|(1<<TWEN)

 // Attendo che il flag di interrupt sia 1
 while ( ! (TWCR & (1<<TWINT));

 // Leggo TWDR
 return TWDR;
}
```

Funzione per l'invio della condizione Stop

```
void TWI_Stop (void){

 // Invio la condizione Stop
 TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
}
```

Modi combinati per l'uso del modulo TWI

Ci sono casi in cui si devono usare diverse impostazioni del modulo per ottenere un certo risultato, tenendo sempre presente che i dati vengono sempre trasmessi dal Master allo Slave e viceversa. Prendendo spunto da un paragrafo del *datasheet* riferito alla lettura di una EEPROM, il Master deve informare quest'ultima sulla locazione di memoria in cui intende leggere. Ciò implica una trasmissione dell'informazione, ovvero una modalità *Master Transmitter* (MT). Successivamente, il Master deve leggere il dato memorizzato nella locazione e ciò implica una ricezione, ovvero una modalità *Master Receiver* (MR). Poiché ciò significa un cambio di direzione dei dati da scrittura a lettura, il Master deve tenere sotto controllo il bus seriale durante questi processi. Questo cambio di direzione si ottiene trasmettendo un *Repeated Start* tra la trasmissione dell'indirizzo e la ricezione dei dati. In Figura 24 sono mostrati i dettagli di una sequenza di accesso alla memoria EEPROM.

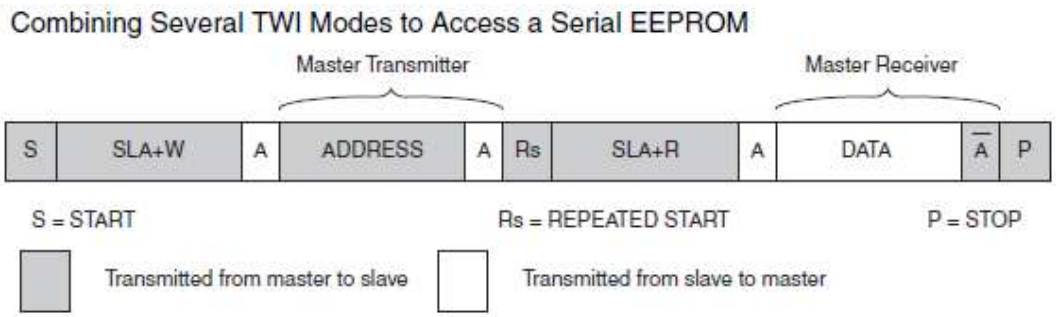

Figura 24: *Modi combinati.*

La EEPROM 24LC256

L'inserimento di questo paragrafo nel contesto di questo capitolo è da ricercarsi in due motivi: il primo perché le immagini allegate dovrebbero chiarire meglio il processo di trasferimento dati, dallo Start allo Stop, il secondo perché sarà lo Slave usato nell'esempio di programma che seguirà. In Figura 25 è mostrata la piedinatura della memoria 24xx256.

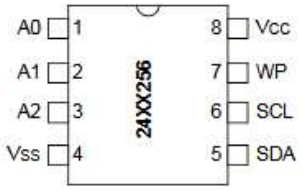

Figura 25: *EEPOM 24xx256.*

La EEPROM 24LC256 è una memoria esterna di tipo seriale-I2C prodotta dalla Microchip che ha 256 Kbit di memoria, corrispondenti a 256000 / 8 bit = 32000 byte, ovvero 32000 locazioni disponibili per la memorizzazione permanente dei dati. Con una tensione di 5V può lavorare con una frequenza operativa massima di 400KHz. Essa presenta il seguente pinout:

- Pin 1, 2, 3: A0, A1, A2: selezione dell'indirizzo del dispositivo sul bus I2C
- Pin 4: VSS massa (GND)
- Pin 5: Vcc (1,7 – 5,5V)
- Pin 6: WP protezione in scrittura. Se collegato a GND abilita la scrittura sulla EEPROM, se collegato a Vcc la disabilita
- Pin 6: SCL linea di clock (bus I2C)
- Pin 7 SDA linea dati (bus I2C)

Byte di controllo

La memoria EEPROM 24LC256 possiede un byte di controllo necessario per la trasmissione e ricezione dati da parte del Master. Il suo formato è riportato in Figura 26.

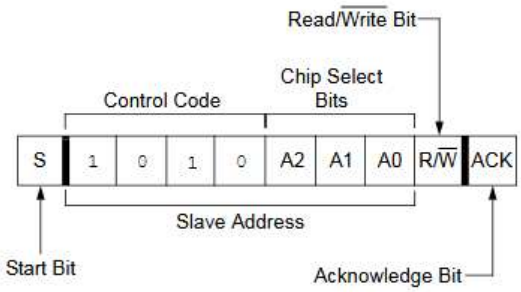

Figura 26: *Control byte (byte di controllo).*

Questo byte è costituito dall'indirizzo a 7 bit dello Slave più il bit lettura-scrittura R/W. L'indirizzo dello Slave è suddiviso a sua volta in due parti: una fissa, chiamata *Control Code*, rappresentata dai 4 bit più significativi 1010, e una impostabile, chiamata Chip Select Bit, rappresentata dai bit di selezione A2, A1 e A0. Ogni bit di selezione assume un valore 0 quando il suo pin corrispondente è collegato a massa ed un valore 1 quando è collegato a Vcc. Se, per esempio, si collegano tutti i pin di selezione a massa l'indirizzo Slave assume il valore 1010000, se si collegano a Vcc assume il valore 1010111. A questo indirizzo, come già detto, si deve aggiungere il valore impostato nel bit R/W (R =1 ; W = 0), in modo da completare il byte.

Per poter scrivere un dato nella EEPROM, oltre al byte di controllo impostato in scrittura, il Master deve trasmettere anche l'indirizzo della locazione di memoria scelta per questa operazione. Questo lo effettua inviando in sequenza i due byte alto (*Address High Byte*) e basso (*Address Low Byte*) che lo costituiscono, come mostrato in Figura 27.

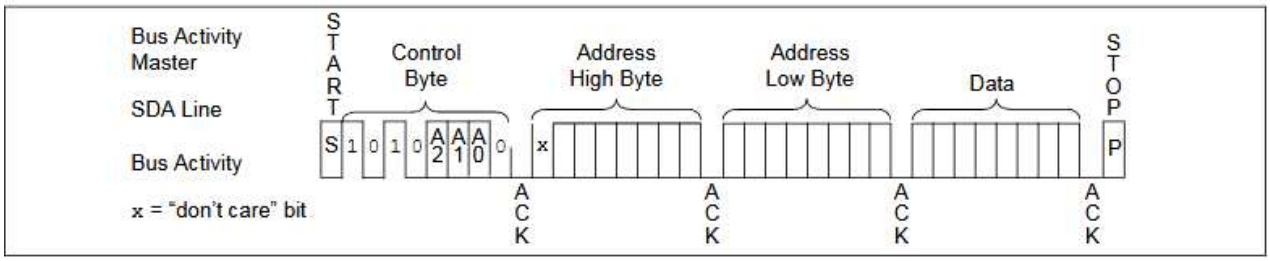

Figura 27: *Write byte.*

Scrittura di un byte (Byte Write)

Solo dopo aver selezionato l'indirizzo della periferica e la locazione di memoria in cui scrivere, il Master procede poi all'invio del dato, che viene temporaneamente memorizzato in un buffer chiamato *Page Write*. Il dato inviato richiede circa 5ms affinché possa essere scritto nella locazione di memoria selezionata. Questo modo di procedere, ovvero l'invio di un byte per volta, è chiamato *Byte Write*.

A titolo di esempio riassuntivo, si supponga che l'indirizzo della EEPROM sia 1010000 (pin A2, A1, A0 collegati a massa) e di voler scrivere il numero 100 (0b01100100) nella locazione di memoria EEPROM 1023. Il valore di High Byte è 0b00000011 mentre il Low Byte è 0b11111111. La sequenza delle operazioni da effettuare per la scrittura di un byte è la seguente:

1. Inviare la condizione Start.
2. Inviare il byte 10100000 (indirizzo della EEPROM + bit Write = 1010000 + 0).
3. Inviare il byte 00000011 (High Byte)
4. Inviare il byte 11111111 (Low Byte)

5. Inviare il byte 01100100 (numero 100)
6. Inviare la condizione Stop
7. Attendere 5 ms per la scrittura del byte

Lettura di un byte (Byte Write)

La procedura per la lettura di un byte è inizialmente identica a quella della scrittura fino all'invio dell'indirizzo della locazione di memoria. A questo punto, il Master invia consecutivamente la condizione *Repeated Start* e nuovamente il byte di controllo, questa volta impostato in lettura. Infine legge il dato ed invia il segnale di Stop. Riferendo il tutto all'esempio riportato sopra, la sequenza delle operazioni da effettuare per la lettura di un byte è la seguente:

1. Inviare la condizione Start.
2. Inviare il byte 10100000 (indirizzo della EEPROM + bit Write = 1010000 + 0).
3. Inviare 00000011 (High Byte).
4. Inviare 11111111 (Low Byte).
5. Inviare la condizione *Repeated Start*.
6. Inviare il byte 10100001 (indirizzo della EEPROM + bit Read = 1010000 + 1).
7. Leggere il valore memorizzato nella locazione 0x3FF (registro TWDR.)
8. Inviare la condizione Stop.

Scrittura di una Page Write

Nel paragrafo sulla scrittura di un byte si è accennato ad un buffer chiamato *Page Write*. Questo buffer ha una capienza di 64 byte e ciò permette di memorizzare temporaneamente altri 63 byte, prima di trasferirli sequenzialmente nelle rispettive locazioni. Poiché la EEPROM 24LC256 possiede 32000 byte di memoria si evince che si hanno a disposizione 32000 / 64 = 500 *Page Write* in cui scrivere. Tuttavia esistono alcune regole che non possono essere ignorate. Ogni pagina inizia e finisce con locazioni di memoria che stabiliscono dei confini ben precisi tra esse, ad esempio la *page write* 1 è definita dalle locazioni 0 - 63 (0x00-0x3F), la *page write* 2 è definita dalle locazioni 64 - 127 (0x40-0x3F) e così via. Ora si supponga di voler trasmettere in sequenza 65 byte. Sarebbe logico pensare che i primi 64 byte vengano scritti nelle locazioni 0x00-0x3F della pagina 1 e l'ultimo byte nella locazione iniziale 0x40 della pagina 2. Purtroppo non è così: se non si specifica esplicitamente che il 65.mo byte deve essere scritto nella locazione iniziale della *page write* 2, esso verrà scritto nella locazione 0x00 della *page write* 1, sovrascrivendo il dato precedentemente memorizzato. Questo ragionamento è valido anche per tutte le altre pagine e va sempre tenuto presente, specialmente quando si vuole memorizzare un considerevole quantitativo di dati. Infine, la scrittura - riscrittura - cancellazione di un singolo byte o di più byte contenuti in una pagina comporta la riscrittura (*refresh*) dell'intera pagina in cui essi sono contenuti. Per esempio, si supponga che la pagina 1 sia

vuota, ovvero con tutte le 64 locazioni scritte a 0xFF, e che si voglia scrivere il byte 0x64 nella locazione 0x00. La scrittura di questo byte comporta anche la riscrittura a 0xFF delle locazioni vuote, non del singolo dato, per cui bisogna fare attenzione a quanto dichiara il *datasheet* che la memoria può essere cancellata/scritta più di un milione di volte. Un semplice esempio di sequenza limitato alla scrittura di cinque byte, iniziando dalla locazione 0x00, è il seguente:

1) Inviare la condizione Start.
2) Inviare il byte 10100000 (indirizzo della EEPROM + bit Write = 1010000 + 0).
3) Inviare il byte 00000000 (High Byte).
4) Inviare il byte 00000000 (Low Byte).
5) Inviare il byte 0x01.
6) Inviare il byte 0x02.
7) Inviare il byte 0x03.
8) Inviare il byte 0x04.
9) Inviare il byte 0x05.
10) Inviare la condizione Stop.
11) Attendere 5 ms per ogni byte che viene scritto.

Lettura sequenziale dei byte

Anche in questo caso, la procedura per la lettura dei byte è inizialmente identica a quella della scrittura fino all'invio dell'indirizzo della locazione di memoria. A questo punto, il Master invia consecutivamente la condizione *Repeated Start* e nuovamente il byte di controllo impostato in lettura. Infine legge i dati dalla locazione iniziale a quella finale ed invia il segnale di Stop. Un semplice esempio limitato alla lettura di cinque byte, iniziando dalla locazione 0x00, è il seguente:

1) Inviare la condizione Start.
2) Inviare il byte 10100000 (indirizzo della EEPROM + bit Write = 1010000 + 0).
3) Inviare 00000000 (High Byte).
4) Inviare 00000000 (Low Byte).
5) Inviare la condizione *Repeated Start*.
6) Inviare il byte 10100001 (indirizzo della EEPROM + bit Read = 1010000 + 1).
7) Leggere il valore del byte memorizzato nella locazione 0x00 (registro TWDR).
8) Leggere il valore del byte memorizzato nella locazione 0x01 (registro TWDR).
9) Leggere il valore del byte memorizzato nella locazione 0x02 (registro TWDR).
10) Leggere il valore del byte memorizzato nella locazione 0x03 (registro TWDR).
11) Leggere il valore del byte memorizzato nella locazione 0x04 (registro TWDR).
12) Inviare la condizione Stop.

Programmi di esempio

Byte Write

Prima di tutto, vorrei suggerire di evitare le *breadboard* e di costruire una schedina, anche su millefori, che contenga a bordo la EEPROM, le resistenze di *pull-up*, il connettore per l'alimentazione e quello per le linee SCL e SDA, linee corrispondenti rispettivamente ai pin PD0 e PD1 dell'Atmega 2560. Come consigliato nelle *application notes* AVR311 e AVR315, il valore tipico per le resistenze di *pull-up* è 4,7 Kohm, che con una tensione di alimentazione pari a 5V garantisce una corrente di circa 1 mA (non è possibile utilizzare quelle interne della MCU in quanto esse hanno un valore ohmico troppo elevato).
Inoltre, per leggere i valori si userà l'USART, questo perché si dovrà scollegare fisicamente il display LCD (usato nei precedenti esempi) in quanto collegato ai sette pin PD0 - PD6 della PORTD. In Figura 28 riporto lo schema di collegamento.

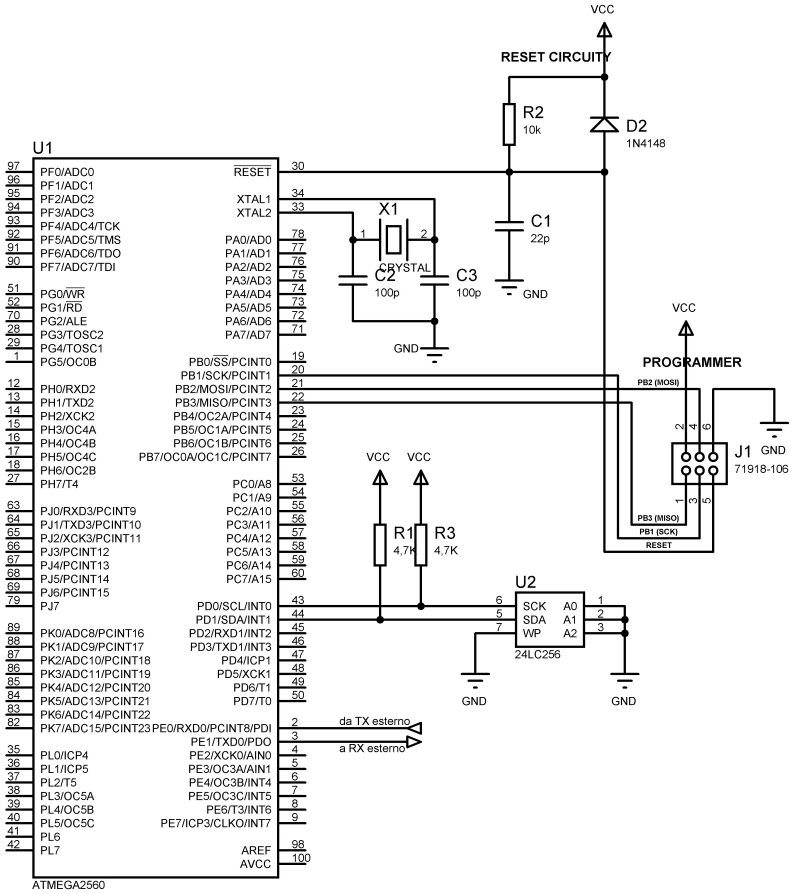

Figura 28: *Schema di collegamento.*

Essendo il programma molto lungo in quanto composto da sette files, in questo capitolo riporterò solo il listato del programma principale e descriverò brevemente le funzioni svolte dagli altri. La cartella dell'esempio "TWI_Byte_Write" contiene i seguenti sorgenti:

uart0.h
Il file contiene il define per il calcolo del prescaler ed i prototipi delle funzioni presenti nel progetto.

uart0.c
Il file contiene le funzioni per l'inizializzazione dell'usart 0, per la trasmissione di un dato, per la ricezione di un dato e per la trasmissione di una stringa.

twi.h
Il file contiene le definizioni di alcune macro estratte dal file "/util/twi.h" della libreria avr libC 2.0.0.0 ed i prototipi delle funzioni presenti nel file twi.c. La scelta di creare un file twi.h ridotto all'essenziale è stata dettata dalla necessità di poter confrontare i codici di stato generati con le rispettive macro. La sola inclusione del file "/util/twi.h" presuppone una conoscenza mnemonica di entrambi.

twi.c
Il file contiene le funzioni per l'inizializzazione dell'interfaccia TWI, per l'invio delle condizioni *Start-Repeated Start - Stop* e per la scrittura e lettura di un byte con e senza ACK – NACK.

24LC256.h
Il file contiene il define per il *control code* della EEPROM ed i prototipi delle funzioni presenti nel file 24LC256.c

24LC256.c
Il file contiene le due funzioni principali per la scrittura e per la lettura di un dato nella EEPROM. Entrambe richiamano le funzioni presenti nel file twi.c.

main.c
Il file contiene il programma principale e le variabili necessarie per la scrittura e lettura della EEPROM e l'inizializza l'USART ed il modulo TWI.

Riporto di seguito il listato:

```
#pragma config BODLEVEL = BODLEVEL_2V7
#pragma config OCDEN = CLEAR
#pragma config JTAGEN = SET
#pragma config SPIEN = SET
#pragma config WDTON = CLEAR
#pragma config EESAVE = CLEAR
#pragma config BOOTSZ = BOOTSZ_4096W_1F000
#pragma config BOOTRST = SET
#pragma config CKDIV8 = CLEAR
#pragma config CKOUT = CLEAR
```

```
#pragma config SUT_CKSEL = SUT_CKSEL_EXTXOSC_8MHZ_XX_16KCK_65MS

// Definisco la frequenza della CPU
#define F_CPU 16000000UL

// Definisco la prima locazione di memoria della pagina 1
#define Start   0x00

// Definisco la 16.ma locazione di memoria della pagina 1
#define END  0x0F

#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>

#include "24lc256.h"
#include "24lc256.c"
#include "twi.h"
#include "twi.c"
#include "uart0.h"
#include "uart0.c"

//*********************************************************************
//  PROGRAMMA PRINCIPALE
//*********************************************************************
int main (void){

 // Creo le variabili necessarie
 unsigned char select_bits = 0x00;      // pin A2,A!,A0 a massa
 unsigned char byte_to_send = 0x64; // Valore da scrivere nella locazione
                                    // (corrispondente al numero 100)
 unsigned char byte_to_read;       // Valore letto dalla locazione

 // Imposto la variabile per contenere il valore da convertire in stringa
 char buffer[5];

 // Inizializzo l'usart
 uart0_init ();
 _delay_ms(100);

 // Inizializzo il TWI
 TWI_Init ();

 // Invio l'indirizzo eeprom e scrivo il valore 0x64 nella locazione 0x3FF
 Eeprom_Write_Byte (select_bits, 0x3FF, byte_to_send);

 // Invio l'indirizzo eeprom e leggo il valore scritto nella locazione 0x3FF
 byte_to_read = Eeprom_Read_Byte (select_bits, 0x3FF);

 // Converto il valore letto in stringa
 itoa(byte_to_read, buffer, 10);
 // Trasmetto il valore al terminale
 uart0_transmit_String (buffer);

 // Invio ritorno carrello e linea successiva
 uart0_transmit_Char ('\r');
 uart0_transmit_Char ('\n');

 while (1);
}
```

Indice Alfabetico

2
24LC256....26
A
ACK....6, 8
acknowledge....8
Address High Byte....27
Address Low Byte....27
Address Match Unit....7, 22
Address Register....7
application notes AVR311....30
AVR311....30
AVR315....30
B
Bit Rate Generator....6
Bit Rate Register....6
bitmask....12, 20
breadboard....30
Bus Interface Unit....5
C
control code....31
Control Code....27
Control Unit....7
controller Start-Stop....6
D
Data and Address Shift Register....5
E
EEPROM....26
Enable Ackowledge....19
F
Fast....4
G
Global Interrupt Enable....11
I
I-bit....11
Interrupt Enable....11
L
layout del PCB....4
LCD....30
limite di capacità....4
M
Master....8
Master Receiver....13, 15, 25
Master Transmitter....13, 25
Master Trasmettitore....22
millefori....30
MR....13
MSB....9
MT....13
N
NACK....6, 9
Not ACK....9
NXP....4
O
open-collector....4
open-drain....4
P
Page Write....27
Phillips....4
PORTD....30
prescaler....6, 12
pull-up....4, 8, 30
R
Read-Write....9
Registri....19
Repeated Start....6, 8 e seg., 25, 28
S
SCL....4 e seg.
SDA....4 e seg.
shifting....21
SLA....7
SLA+RW....10
Slave....8
Slave Address....7
Slave Receiver....13, 17
Slave Transmitter....13, 16
SR....13
SREG....11
ST....13
Start....6
Status Register....6
Stop....6, 8
T
TW....19
TW Bit Rate....19
TW Bit Rate Register....19
TW Control Register....19
TW Enable....20
TW General Call Enable....21
TW Interrupt Enable....19 e seg.
TW Interrupt Flag....19
TW Start Condition....20
TW Stop Condition....20
TWA0....21
TWA6....21
TWAR....7, 22
TWBR....6, 19

TWBR0.....19
TWBR7.....19
TWCR.....7, 19
TWD0.....21
TWD7.....21
TWDR.....5, 29
TWEN.....20
TWGCE.....7, 21
TWIE.....11, 19 e seg.
TWINT.....7, 19
Two Wires Interface.....4
TWPS.....6
TWPS0.....20
TWPS1.....20
TWS3.....20
TWS7.....20
TWSR.....6
TWST.....13
TWSTA.....20
TWSTO.....20
U
USART.....30

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
14/12 /20	1.0	Marcello Pinna	Mauro Laurenti	Versione Originale.