UNIVERSITA' DEGLI STUDI DI TRIESTE
FACOLTA' DI INGEGNERIA

Dipartimento di Elettronica, Elettrotecnica ed Informatica

TESI DI LAUREA TRIENNALE
in
ELETTRONICA

Progetto di un circuito a microcontrollore per la gestione del ricetrasmettitore impiegato nel satellite Atmocube

Laureando:
Stefano Punis

Relatore:
Chiar.mo Prof. Mario Fragiacomo

ANNO ACCADEMICO 2004-2005

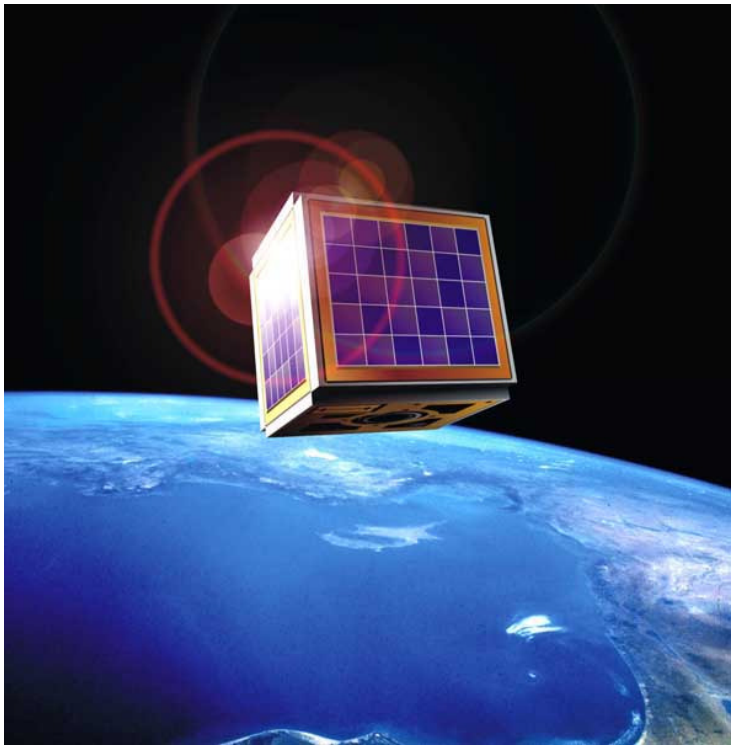
…lascia che le cose vadano come devono andare;
almeno ogni tanto…

1 – Introduzione

In questo capitolo viene fornita un'esposizione generale sul progetto realizzato. Vengono esposte inoltre le funzionalità del circuito accompagnate da una breve descrizione dei componenti impiegati.

1.1 – Descrizione generale ed obiettivi

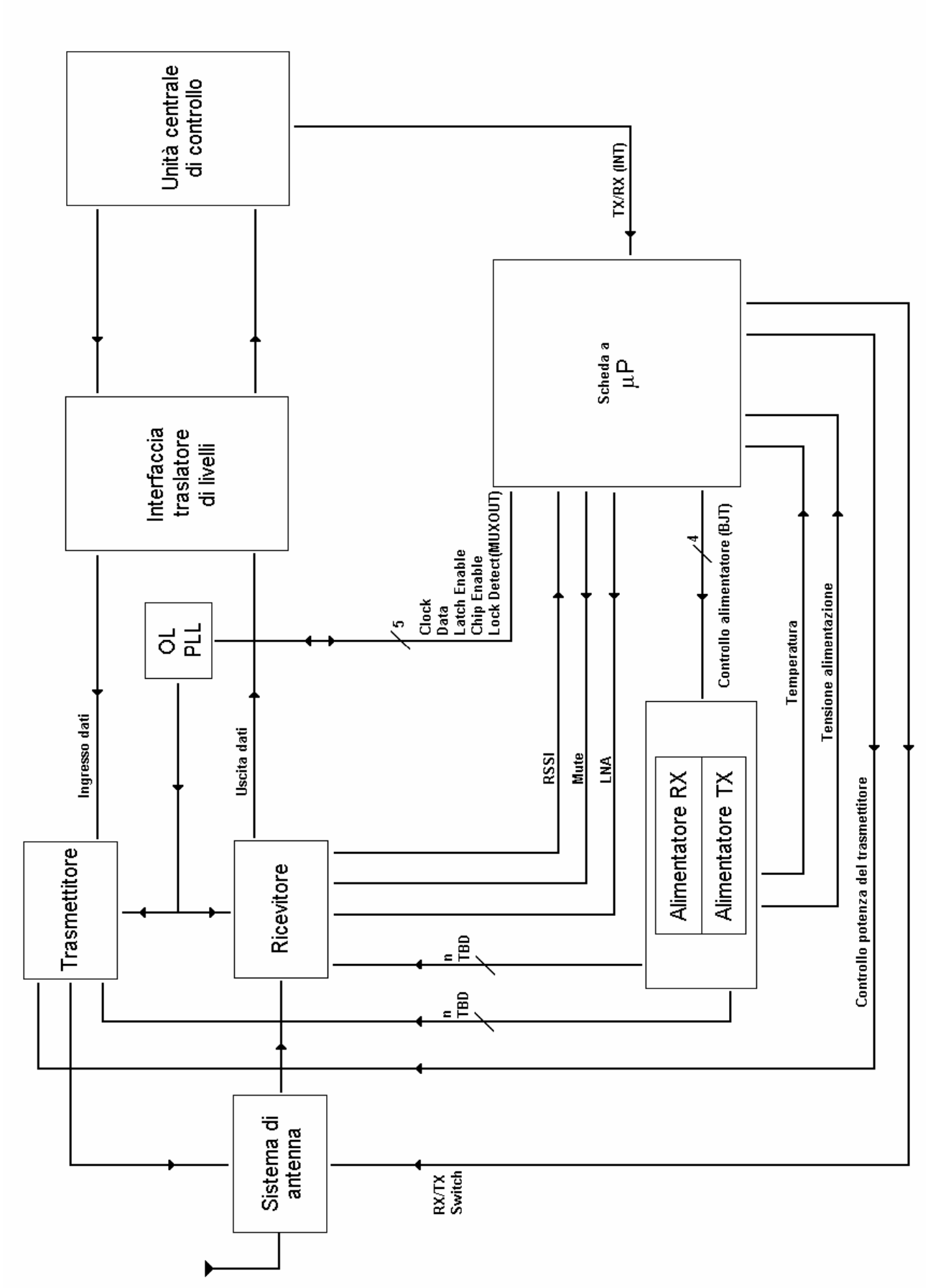
Il presente elaborato tratta la progettazione di un circuito a microcontrollore per la gestione e il controllo di un ricetrasmettitore a sua volta sviluppato per la comunicazione via radio tra una stazione terrestre e un satellite. Questa scheda fa parte dei vari circuiti elettronici già implementati o in fase di implementazione per il progetto Atmocube, di cui l'Università di Trieste ne è fautrice.
Il termine Atmocube deriva dal fatto che questo progetto ha come finalità la realizzazione di un mini satellite completo dalla forma cubica di lato 13 cm. Tale satellite lanciato una volta nello spazio invierà informazioni riguardo il campo magnetico terrestre, l'atmosfera e le radiazioni, grazie alla presenza di un magnetometro e un dosimetro.
Essendo il progetto Atmocube tutt'ora in fase di elaborazione ed essendoci in futuro altri studenti che prenderanno parte a tale progetto, tale elaborato, oltre ad essere una descrizione del lavoro svolto risulta essere allo stesso tempo una guida per coloro che continueranno a lavorare a questa specifica parte del satellite.
Ecco perché alcune parti, in particolare i protocolli di controllo, vengono descritti nel loro funzionamento anche se tali informazioni sono ricavabili da altri testi.
La scheda sviluppata, nonostante sia stata realizzata su circuito stampato, è utilizzabile solo in laboratorio per le varie prove e non è una scheda definitiva per essere montata sul satellite.
Per quanto riguarda il programma che governa il microcontrollore, anch'esso è possibile oggetto di future modifiche o comunque aggiornamenti per adattarlo al meglio ai futuri sviluppi.
In base alle specifiche iniziali infatti non è stato possibile scrivere un programma definitivo in particolare per quanto riguarda il "main" o programma principale. Quello che è stato fatto in modo sicuramente quasi definitivo sono state le varie subroutine di gestione dei singoli blocchi.
Sarà compito di coloro che svilupperanno a fondo tale ricetrasmettitore, scrivere un programma "main", il quale richiami adeguatamente le singole subroutine di gestione di tutti i blocchi.



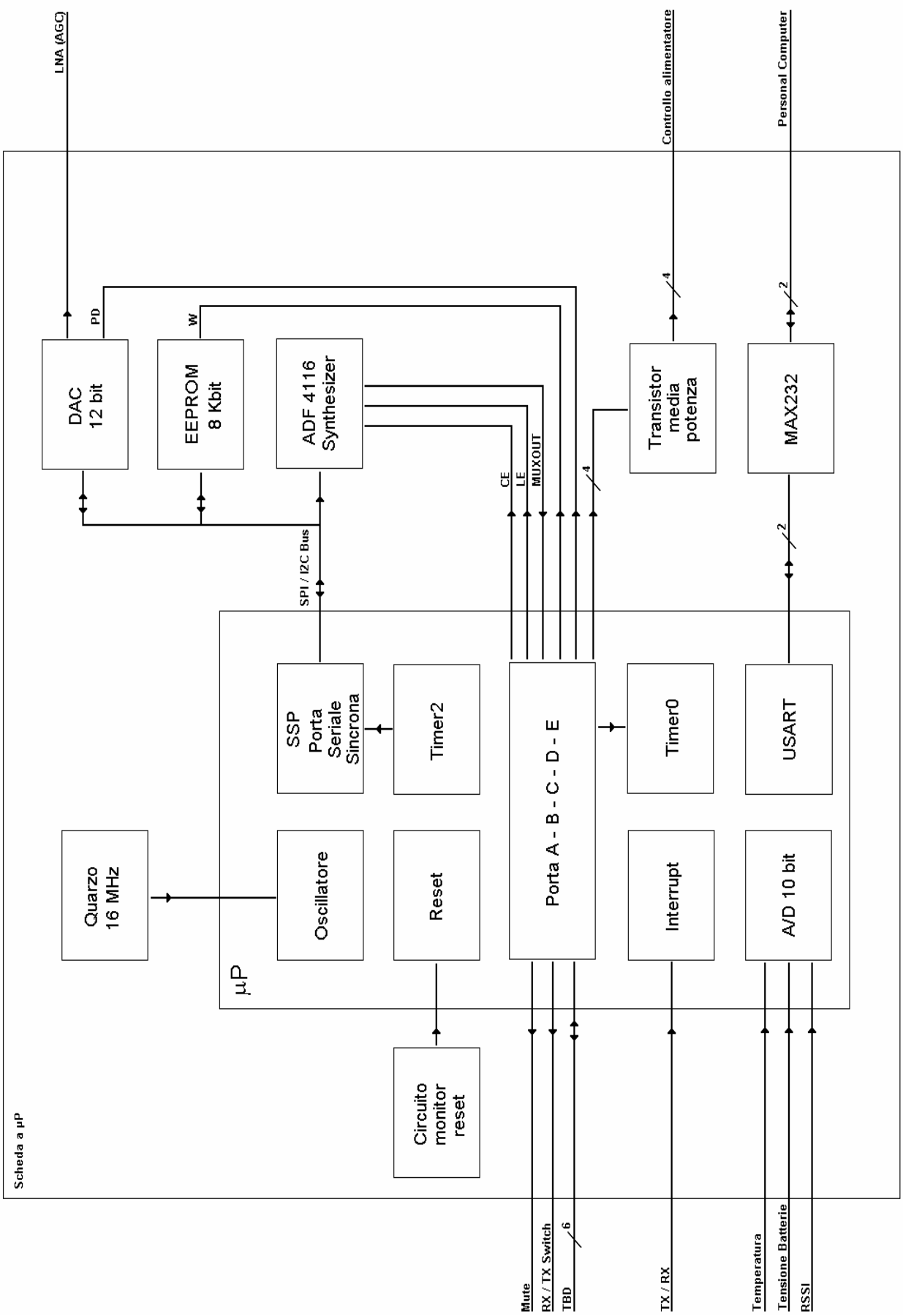
- Un disegno a computer di un mini satellite nello spazio -

1.2 – Funzioni del circuito di controllo

Uno dei compiti del circuito a microcontrollore è quello di controllare l'oscillatore locale utilizzato per generare le opportune frequenze del ricetrasmettitore. L'oscillatore locale è stato già realizzato in un altro progetto mediante un circuito a PLL che utilizza un circuito integrato sintetizzatore di frequenza. Quest'ultimo per essere configurato opportunamente e generare determinate frequenze, deve essere programmato a sua volta con dei dati inviati tramite protocollo seriale SPI.
Le linee che collegano il circuito PLL al microcontrollore sono in tutto cinque: una di dati e una di clock necessarie per il protocollo di comunicazione SPI, due linee di abilitazione LE (Latch Enable) e CE (Chip Enable) e una linea di controllo dell'avvenuto aggancio da parte del circuito PLL sulla frequenza desiderata LOCK DETECT (MUXOUT).
Un'altra funzione della scheda a microcontrollore prevede il monitoraggio di tre parametri analogici: il primo è l'intensità del segnale radio ricevuto RSSI, proveniente appunto dal blocco del ricevitore, il secondo è la temperatura raggiunta dai blocchi di alimentazione del ricetrasmettitore e infine il terzo è il valore della tensione delle batterie che alimentano il satellite.
Come detto questi tre parametri sono segnali analogici e pertanto fanno capo agli ingressi del convertitore analogico – digitale presente nel microcontrollore.
Si provvede inoltre a gestire tramite il segnale denominato LNA, il circuito del controllo automatico del guadagno del ricevitore. Dato che il microcontrollore deve controllare un segnale analogico e non dispone di un convertitore digitale – analogico, si è usato uno esterno.
Una qualsiasi linea del circuito viene utilizzata per gestire il comando MUTE del ricevitore, nel caso si desiderasse non voler ricevere il rumore presente in assenza di segnale utile.
I circuiti di alimentazione del ricetrasmettitore sono controllati anche dal microcontrollore, il quale provvede ad accendere e spegnere in maniera sequenziale i blocchi funzionali che fanno parte della catena di trasmissione. L'alimentazione in sequenza è necessaria per evitare che uno stadio venga pilotato prima che la sua porta di uscita sia chiusa sulla sua impedenza di carico. Questo evita disadattamenti di impedenza e quindi riflessioni indesiderate, durante la fase di transizione da acceso a spento e viceversa. In pratica il microcontrollore trasforma il comando di abilitazione alla trasmissione, ricevuto dall'unità centrale di controllo, in una sequenza di comandi. Il controllo delle linee di alimentazione dei vari stadi è ottenuto tramite alcuni transistori di media - bassa potenza.
Il segnale di avvio di trasmissione da parte dell'unità centrale viene gestito dal microcontrollore tramite un ingresso di interrupt.
Per commutare il sistema d'antenna sul blocco di trasmissione o sul blocco di ricezione, è possibile implementare un ulteriore comando non presente attualmente, controllato a sua volta da una qualsiasi linea del circuito. Sono presenti infatti sette linee libere utili per eventuali espansioni e modifiche future.
Risulta non essere presente inoltre anche un circuito di controllo della potenza del trasmettitore non essendone ancora stato definito uno in dettaglio.
La scheda a microcontrollore oltre a controllare l'hardware descritto precedentemente, dispone di una memoria EEPROM, un'interfaccia RS232 e un circuito di controllo del reset.
La memoria EEPROM consente, qualora risultasse necessario, la memorizzazione di vari parametri in modo permanente anche se successivamente viene tolta l'alimentazione, come ad esempio i valori dei segnali analogici e i registri di configurazione del circuito PLL.
Il circuito RS232 invece, è stato implementato per permettere la comunicazione con un qualsiasi Personal Computer dotato di porta seriale. Tramite questo circuito si può controllare l'intero microcontrollore e conseguentemente agire sulle varie periferiche.
Infine per il corretto funzionamento della scheda nei primi istanti dopo aver fornito l'alimentazione, un apposito circuito integrato gestisce il reset del microcontrollore. Fino a quando la tensione di alimentazione non risulta stabilizzarsi sopra una determinata soglia, tale circuito provvede a mantenere uno stato di reset.



- Schema a blocchi generale del sistema di comunicazione radio del satellite -



- Schema a blocchi della scheda di controllo a microcontrollore -

1.3 – Scelta dei componenti impiegati

1.3.1 – Microcontrollore

Il microcontrollore usato è un PIC 16F877 a 8 bit della Microchip, già introdotto in alcune lezioni del corso di Elettronica III per quanto riguarda la sua struttura interna. Questo tipo di microcontrollore presenta la caratteristica di disporre di molte periferiche interne utili al progetto della scheda. Infatti sono disponibili una porta seriale sincrona MSSP utilizzabile per generare segnali di tipo SPI e anche di tipo IIC, una porta seriale asincrona USART per consentire una comunicazione del tipo RS232, un convertitore analogico – digitale a 10 bit multiplexabile fino ad un massimo di 8 canali, tre timer indipendenti utili sia come contatori che come temporizzatori ed infine 5 porte esterne per un totale di 33 linee interfacciabili.
Questo microcontrollore si basa su tecnologia RISC e per la stesura del programma sono sufficienti solamente 35 diverse istruzioni in linguaggio assembler.
Il codice eseguito dal microcontrollore risiede in una memoria a tecnologia flash programmabile un numero indefinito di volte. La scrittura in modalità standard della memoria flash con il codice dell'utente avviene per mezzo di un opportuno programmatore esterno che si aggancia con il microcontrollore tramite cinque piedini, tra i quali tre di alimentazione (Vcc, Vpp, GND) e due di dati (Data, Clock).
Ogni istruzione per essere eseguita impiega un ciclo di clock, fatta eccezione per alcune istruzioni di salto incondizionato o di test che ne impiegano due.
Per la velocità di clock si è scelto di usare un quarzo da 16 MHz, in modo da mantenersi in via precauzionale un po' sotto la velocità massima di funzionamento. Quindi dato che la frequenza di lavoro effettiva risulta essere divisa per quattro, ogni istruzione viene eseguita in 250 ns.

1.3.2 – Convertitore digitale - analogico

Come precedentemente detto, il microcontrollore non dispone di un convertitore DAC e pertanto è stato necessario dotare la scheda di uno esterno.
Per non dover impiegare un numero notevole di linee, si è pensato di impiegarne uno gestibile con lo standard IIC, che impiega principalmente solo due segnali per la sua comunicazione.
Il DAC utilizzato è un AD5321 della Analog Device che permette una buona conversione grazie alla risoluzione di 12 bit.

1.3.3 – Memoria EEPROM

Per il mantenimento dei vari dati si è pensato di utilizzare una memoria eeprom esterna seriale del tipo 24Cxx. Anche questo è un componente gestibile con lo standard IIC.
La capacità di tale memoria è di 8 Kbit e consente quindi la memorizzazione di 1024 registri da 8 bit.
In ogni caso se in futuro risultasse necessario, è possibile sostituire tale componente con uno compatibile a capacità più elevata, senza compromettere il funzionamento dei protocolli o comunque della scheda.

1.3.4 – Adattatore di livelli RS232

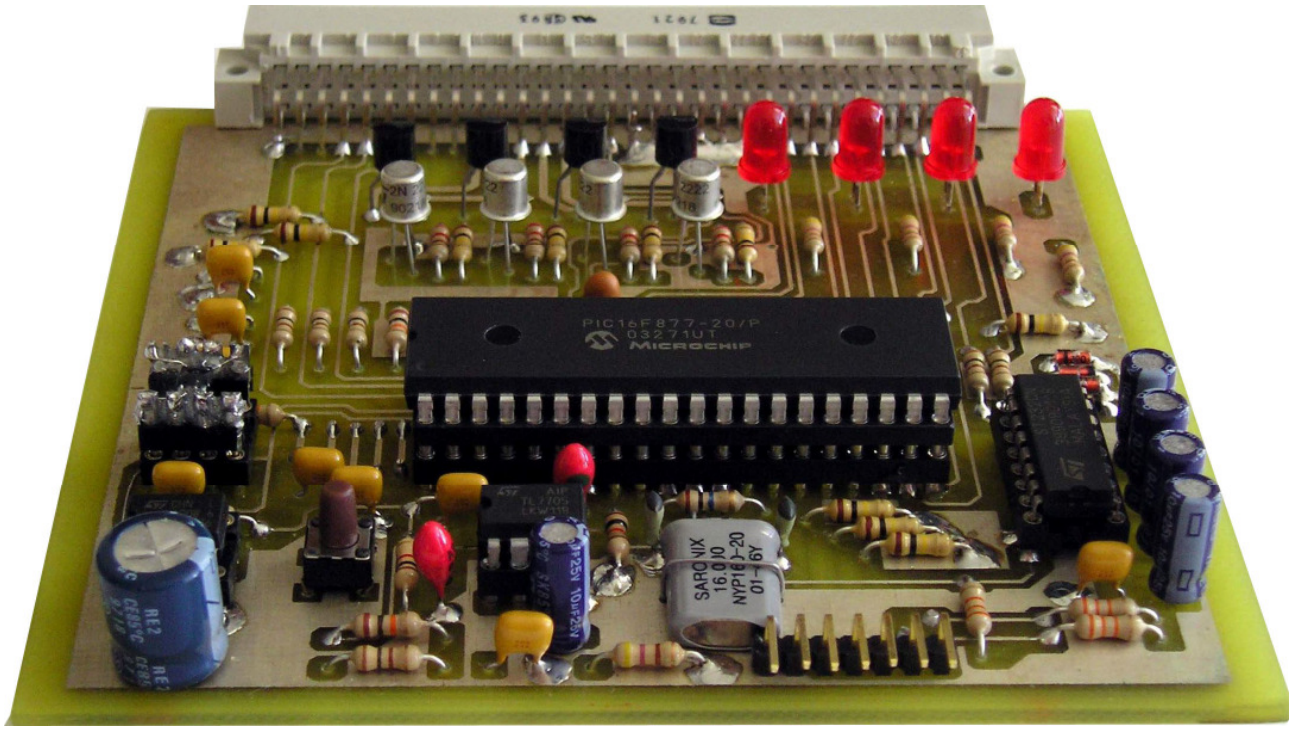
Dato che i segnali della porta seriale del PC non sono direttamente compatibili con i livelli TTL del microcontrollore è necessario impiegare un adattatore di livelli.
Il componente usato è il MAX232 della MAXIM, sviluppato proprio per comunicazioni di questo tipo.
Per le prove fatte si è utilizzata una versione base che impiega capacità esterne dell'ordine del μF.

1.3.5 – Circuito di reset

L'integrato usato è il TL7705 della TEXAS INSTRUMENTS progettato per gestire l'ingresso di reset dei microcontrollori. Le due ultime cifre della sua sigla indicano proprio il funzionamento con un'alimentazione a 5 V con la quale appunto lavora il microcontrollore.

1.3.6 – Transistor per i blocchi di media – bassa potenza

Per implementare il comando di accensione e spegnimento dei circuiti di alimentazione del ricetrasmettitore, sono stati impiegati dei blocchi di media – bassa potenza a transistor. I componenti usati sono dei comuni transistor quali i 2N2222, collegati alle linee del microcontrollore e i BC327, collegati con l'esterno. Essendo data come specifica iniziale una corrente di assorbimento massima per ogni blocco di 0,5 A, tramite un adeguato dimensionamento, tali tipi di transistor sono risultati ottimali.



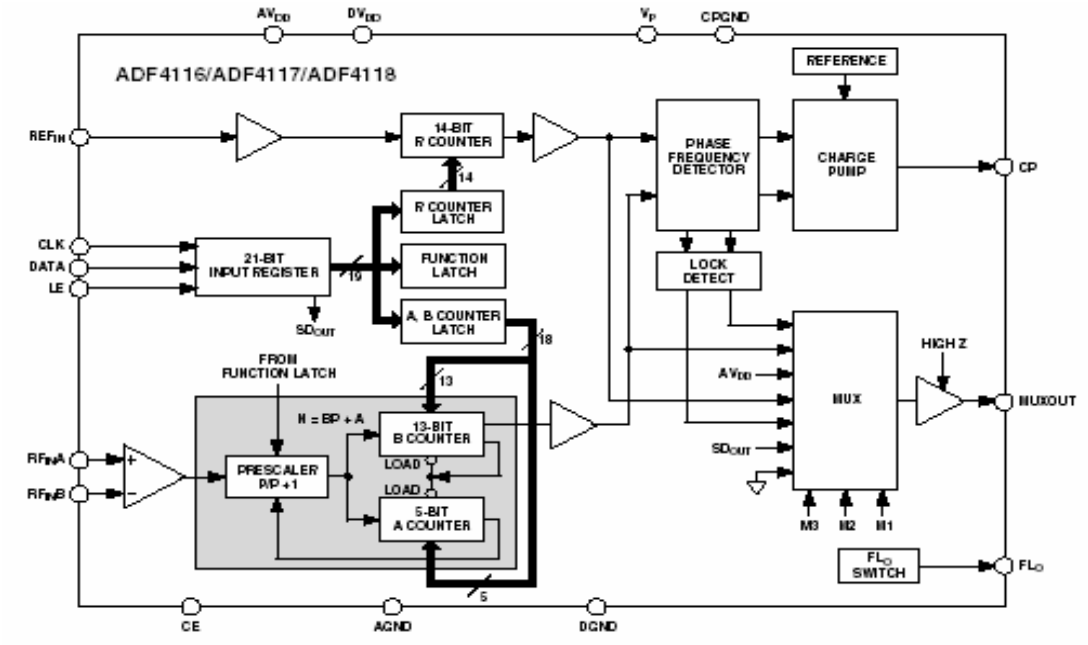
- Un'immagine della scheda -

Al centro si nota il microcontrollore, a sinistra il DAC e la memoria EEPROM, in basso il circuito di reset, il quarzo a 16 MHz e il connettore dorato per il sintetizzatore, a destra l'integrato per l'interfacciamento con il PC, in alto le 4 coppie di transistor e il connettore dove fanno capo tutti i rimanenti collegamenti.

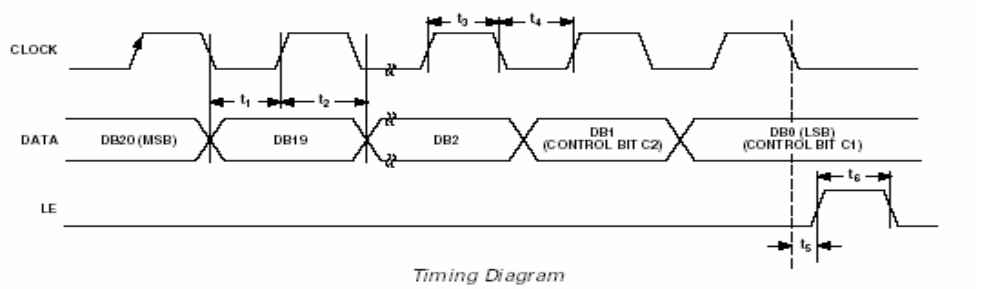
2 – Gestione delle periferiche

In questo capitolo viene descritto in dettaglio il funzionamento delle periferiche che interagiscono con il microcontrollore.

2.1 – Controllo del circuito sintetizzatore di frequenza



Nella figura si vede l'organizzazione interna del circuito sintetizzatore di frequenza ADF 4116 della Analog Device utilizzato nel progetto del PLL. Lo stadio di interesse per la gestione a microcontrollore risulta quello in cui si nota la presenza di uno shift register da 21 bit. Infatti tramite questo shift register si precaricano i dati che successivamente vengono inviati ai tre registri (R-COUNTER, FUNCTION LATCH e A-B COUNTER), i quali a loro volta comandano la sintesi di frequenza. La comunicazione tra questo integrato e il microcontrollore viene pertanto effettuata tramite le tre linee che raggiungono tale shift register: CLK (segnale di clock), DATA (segnale dei dati), LE (Latch Enable, segnale di abilitazione del registro). Sono inoltre necessarie altre due linee: il CE (Chip Enable, abilitazione dell'integrato sintetizzatore) e il MUXOUT o LOCK DETECT (opportunamente configurato risulta utile per verificare l'avvenuto aggancio di fase del sistema PLL).
Il protocollo di comunicazione è SPI compatibile e i dati vengono inseriti nello shift register da 21 bit ogni fronte di salita del clock considerando come bit più significativo il primo che entra.

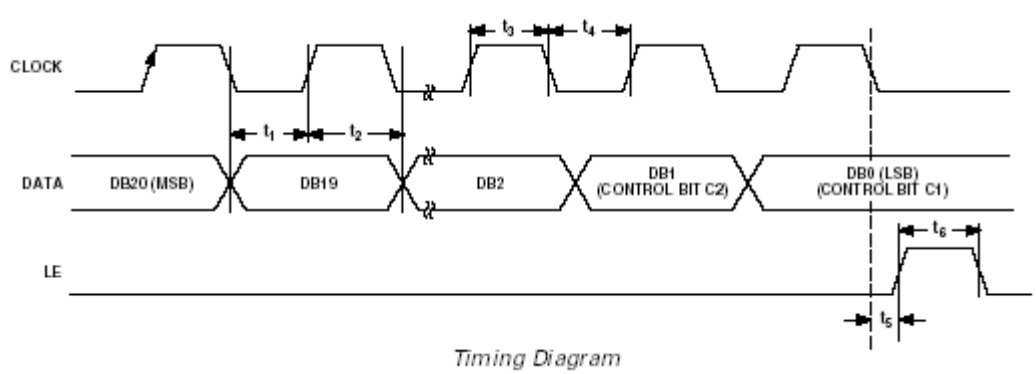


Timing Diagram

Una volta caricato opportunamente lo shift register, per trasferire i dati negli opportuni registri, è necessario abilitare per un breve periodo la linea LE dell'integrato. Per scegliere quale dei tre registri deve essere il destinatario dei dati, si configurano opportunamente i due bit meno significativi del dato precedentemente precaricato nello shift register.
Per quanto riguarda le temporizzazioni è da tener presente che la frequenza del clock non può superare i 20 MHz e che il segnale LE di abilitazione deve andare alto dopo un attesa di almeno 10 ns dall'ultimo impulso di clock per un tempo di almeno 20 ns.

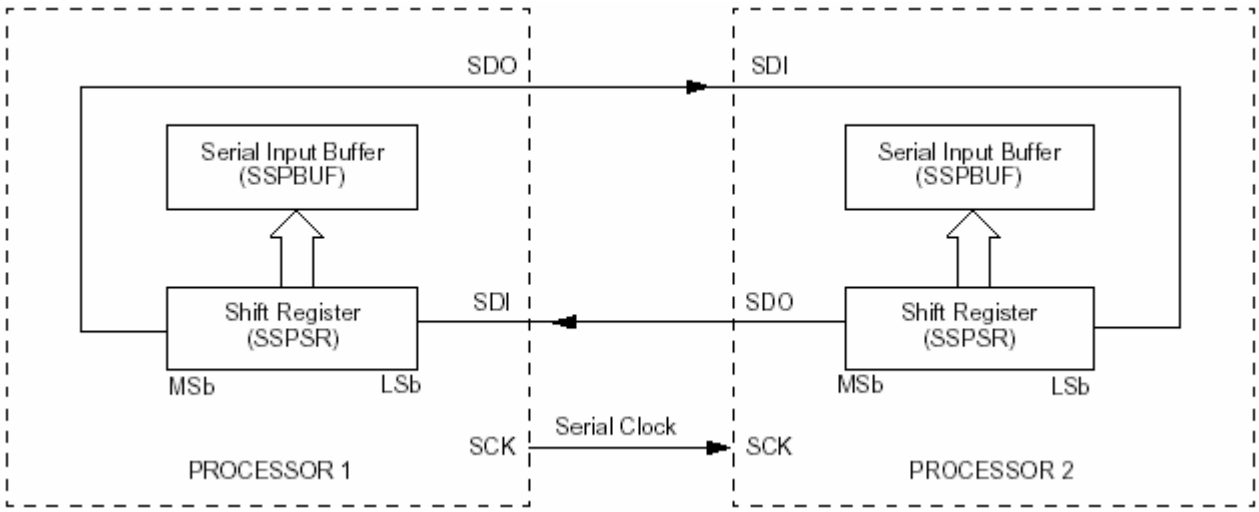
2.1.1 – Protocollo di comunicazione SPI

Il protocollo di comunicazione SPI (Serial Peripheral Interface) è un protocollo sincrono che permette a un dispositivo definito come Master a iniziare una comunicazione con un altro dispositivo definito come Slave.
Il segnale di clock è generato esclusivamente dal dispositivo Master che sincronizza l'intera comunicazione e controlla quando il dato può cambiare e quando può essere pronto per essere letto. I dispositivi Slave quindi non possono manipolare il clock; gli opportuni registri della periferica Master, tramite opportuni bit, controllano come gli Slave reagiscono al segnale di clock.
La comunicazione avviene con uno scambio contemporaneo di dati tra i due dispositivi per mezzo di tre linee, una di clock (CLK), una di dati in uscita (SDO), uno di dati in entrata (SDI) e non quindi con un invio di dati singolarmente da uno dei due dispositivi: come i dati entrano ed escono nel Master, contemporaneamente e rispettivamente escono ed entrano nello Slave.
In generale quando sono presenti più dispositivi Slave si implementa un ulteriore segnale attivo basso definito come SS (Slave Select) che abilita uno Slave alla volta. Se lo Slave è unico allora tale linea può anche essere omessa, anche se una rigorosa implementazione di tale protocollo la impiegherebbe. L'impiego principale di questo piedino consiste nel tenere un dispositivo disabilitato fino poco prima dell'invio dei dati in modo da prevenire malfunzionamenti dovuti a segnali spuri o disturbi in genere.
Tipicamente il dato presente sulla linea cambia durante il fronte di salita o di discesa del clock, in modo da far avvenire la lettura del dato nel fronte opposto al momento di cambiata. A seconda di questi particolari, si ottengono vari tipi di comunicazione SPI, ognuno dei quali deve essere implementato a seconda del dispositivo che si deve interfacciare. Dal diagramma delle temporizzazioni del circuito sintetizzatore si nota che il campionamento del dato avviene sul fronte di salita del clock mentre invece il cambio del dato avviene sul fronte di discesa.



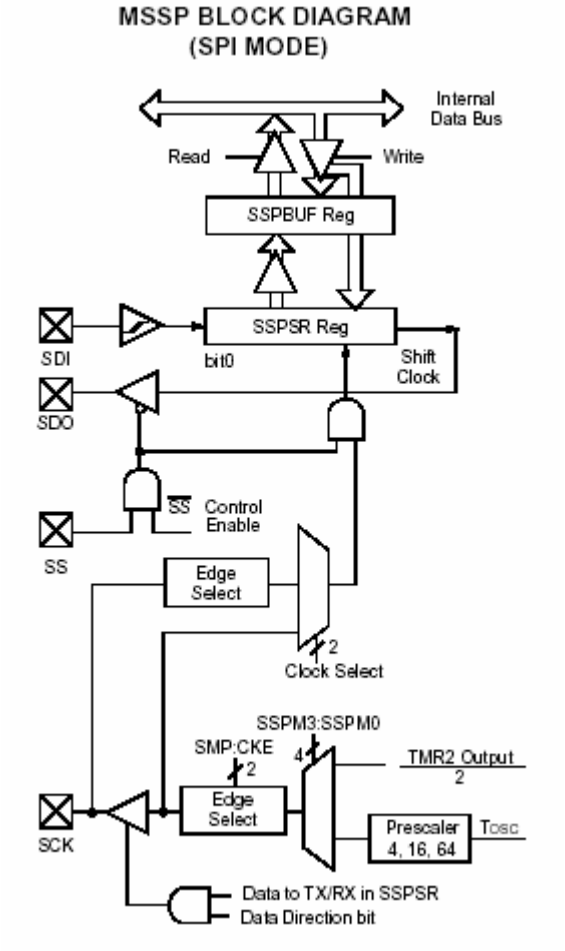
Timing Diagram

Il protocollo SPI crea quindi un anello tra i due dispositivi Master e Slave permettendo ai dati che lasciano il Master di raggiungere lo Slave e viceversa.



2.1.2 – Modulo hardware MSSP Microchip in configurazione SPI

L'SPI è implementato nel microcontrollore Microchip tramite il modulo hardware definito come MSSP (Master Synchronous Serial Port) che permette la comunicazione seriale tra due o più dispositivi.



Riferendosi al modulo hardware Microchip soprastante, il registro SSPSR è lo shift register, che scorrendo ad ogni fronte di clock, manda in uscita sulla linea SDO uno degli otto bit del byte da inviare. Contemporaneamente tramite la linea SDI tale shift register acquisisce i singoli bit che provengono dall'altra periferica Slave. Una volta che l'intero byte è stato scambiato dai dispositivi, allora il byte ricevuto dal Master viene copiato nel buffer SSPBUF che può a sua volta essere letto via software per l'elaborazione. Una semplice scrittura del registro SSPBUF invece abilita la periferica a caricare lo shift register SSPSR e ad iniziare una nuova comunicazione e quindi un nuovo scambio di dati.
Dalla figura si nota inoltre la possibilità di scegliere la sorgente del clock e cioè può essere prelevata da un prescaler a cui fa capo l'oscillatore del microcontrollore, oppure dall'uscita del timer TIMER2 interno al microcontrollore. Con l'uso del TIMER2 è possibile ottenere un periodo di clock decisamente più elevato di quello prelevabile dal prescaler.
E' presente inoltre un circuito che provvede a configurare il fronte del clock.
Il modulo MSSP può essere configurato sia come *SPI* sia come IIC ma non può funzionare contemporaneamente in tutti due i modi.
La configurazione e il controllo della periferica SPI avviene per mezzo di due registri denominati SSPCON (SSP Control) e SSPSTAT (SSP Status).

SSPCON: SYNC SERIAL PORT CONTROL REGISTER (ADDRESS 14h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 WCOL: Write Collision Detect bit

Master mode:

1 = A write to SSPBUF was attempted while the I2C conditions were not valid

0 = No collision

Slave mode:

1 = SSPBUF register is written while still transmitting the previous word (must be cleared in software)

0 = No collision

bit 6 SSPOV: Receive Overflow Indicator bit

In SPI mode:

1 = A new byte is received while SSPBUF holds previous data. Data in SSPSR is lost on overflow. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid overflows. In Master mode, the overflow bit is not set, since each operation is initiated by writing to the SSPBUF register. (Must be cleared in software.)

0 = No overflow

In I^2C mode:

1 = A byte is received while the SSPBUF is holding the previous byte. SSPOV is a "don't care" in Transmit mode. (Must be cleared in software.)

0 = No overflow

bit 5 SSPEN: Synchronous Serial Port Enable bit

In SPI mode,

When enabled, these pins must be properly configured as input or output

1 = Enables serial port and configures SCK, SDO, SDI, and SS as the source of the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

In I^2C mode,

When enabled, these pins must be properly configured as input or output

1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

bit 4 CKP: Clock Polarity Select bit

In SPI mode:

1 = Idle state for clock is a high level

0 = Idle state for clock is a low level

In I^2C Slave mode:

SCK release control

1 = Enable clock

0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In I^2C Master mode:

Unused in this mode

bit 3-0 SSPM3:SSPM0: Synchronous Serial Port Mode Select bits

0000 = SPI Master mode, clock = Fosc/4

0001 = SPI Master mode, clock = Fosc/16

0010 = SPI Master mode, clock = Fosc/64

0011 = SPI Master mode, clock = TMR2 output/2

0100 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control enabled.

0101 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ can be used as I/O pin.

0110 = I^2C Slave mode, 7-bit address

0111 = I^2C Slave mode, 10-bit address

1000 = I^2C Master mode, clock = Fosc / (4 * (SSPADD+1))

1011 = I^2C Firmware Controlled Master mode (slave idle)

1110 = I^2C Firmware Controlled Master mode, 7-bit address with START and STOP bit interrupts enabled

1111 = I^2C Firmware Controlled Master mode, 10-bit address with START and STOP bit interrupts enabled

1001, 1010, 1100, 1101 = Reserved

SSPSTAT: SYNC SERIAL PORT STATUS REGISTER (ADDRESS: 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 SMP: Sample bit

SPI Master mode:

1 = Input data sampled at end of data output time

0 = Input data sampled at middle of data output time

SPI Slave mode:

SMP must be cleared when SPI is used in slave mode

In I^2C Master or Slave mode:

1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)

0 = Slew rate control enabled for high speed mode (400 kHz)

bit 6 CKE: SPI Clock Edge Select (Figure 9-2, Figure 9-3 and Figure 9-4)

SPI mode:

For CKP = 0

1 = Data transmitted on rising edge of SCK

0 = Data transmitted on falling edge of SCK

For CKP = 1

1 = Data transmitted on falling edge of SCK

0 = Data transmitted on rising edge of SCK

In I^2C Master or Slave mode:

1 = Input levels conform to SMBus spec

0 = Input levels conform to I^2C specs

bit 5 D/A: Data/$\overline{\text{Address}}$ bit (I^2C mode only)

1 = Indicates that the last byte received or transmitted was data

0 = Indicates that the last byte received or transmitted was address

bit 4 P: STOP bit

(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)

1 = Indicates that a STOP bit has been detected last (this bit is '0' on RESET)

0 = STOP bit was not detected last

bit 3 S: START bit

(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)

1 = Indicates that a START bit has been detected last (this bit is '0' on RESET)

0 = START bit was not detected last

bit 2 R/W: Read/Write bit Information (I^2C mode only)

This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next START bit, STOP bit or not $\overline{\text{ACK}}$ bit.

In I^2C Slave mode:

1 = Read

0 = Write

In I^2C Master mode:

1 = Transmit is in progress

0 = Transmit is not in progress

Logical OR of this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSSP is in IDLE mode.

bit 1 UA: Update Address (10-bit I^2C mode only)

1 = Indicates that the user needs to update the address in the SSPADD register

0 = Address does not need to be updated

bit 0 BF: Buffer Full Status bit

Receive (SPI and I^2C modes):

1 = Receive complete, SSPBUF is full

0 = Receive not complete, SSPBUF is empty

Transmit (I^2C mode only):

1 = Data transmit in progress (does not include the ACK and STOP bits), SSPBUF is full

0 = Data transmit complete (does not include the $\overline{\text{ACK}}$ and STOP bits), SSPBUF is empty

2.1.3 – Gestione del circuito sintetizzatore

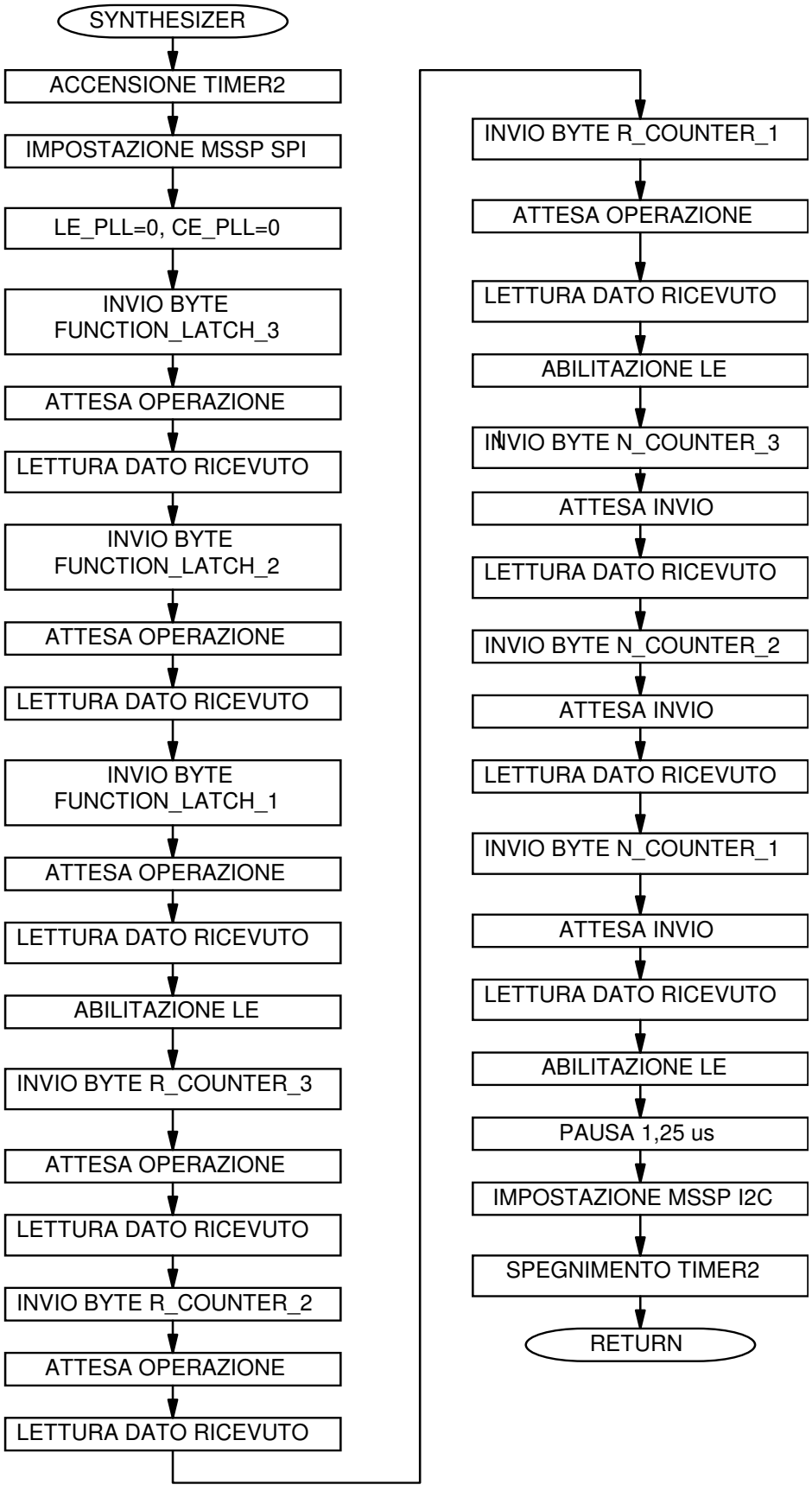
Per far funzionare in modo corretto il sintetizzatore ADF4116, è necessario seguire una ben precisa sequenza di inizializzazione. Questa sequenza non è unica, infatti la casa costruttrice ne propone diverse. Di seguito verrà descritta la sequenza scelta in questo caso e usata nel programma.
Tale sequenza viene denominata come "Il metodo del piedino CE (The CE Pin Method)", in quanto viene utilizzato proprio il piedino CE (Chip Enable) e consiste nel seguente elenco temporale di operazioni:

- applicazione della tensione di alimentazione al circuito sintetizzatore;
- applicazione di un livello logico di tensione pari a 0 al piedino CE (Chip Enable) per portare in power down il circuito sintetizzatore;
- programmazione del registro FUNCTION LATCH;
- programmazione del registro R-COUNTER;
- programmazione del registro N-COUNTER (AB-COUNTER);
- applicazione di un livello logico di tensione pari a 1 al piedino CE (Chip Enable) per portare fuori dal power down il circuito sintetizzatore;
- attesa di 1 μs per consentire di raggiungere uno stato stabile ai circuiti interni del sintetizzatore.

Il piedino CE del sintetizzatore funge in questo caso anche da piedino SS (Slave Select) nella trasmissione SPI.
Dallo schema di funzionamento del sintetizzatore ADF4116, si è visto che per programmare i registri FUNCTION LATCH, R-COUNTER, N-COUNTER è necessario precaricare i dati all'interno di uno shift register da 21 bit (dal n° 20 al n° 0). Essendo l'SPI un protocollo di dati che come minimo invia un byte e quindi 8 bit alla volta, per poter mandare un minimo di 21 bit sono necessari tramite la periferica SPI l'invio di 3 byte e cioè 24 bit (dal n° 23 al n° 0).
A tale scopo si creano all'interno della memoria ram del microcontrollore 3 registri ognuno dei quali contiene uno dei tre byte da inviare. Ad esempio per il registro FUNCTION LATCH si crea un registro FUNCTION_LATCH_3 contenente i bit più significativi dal n° 23 al n° 16, un registro FUNCTION_LATCH_2 contenente i bit dal n° 15 al n° 8 e un registro FUNCTION_LATCH_1 contenente i bit dal n° 7 al n° 0.
Lo stesso si fa per gli altri due registri: per l'R-COUNTER si creano i registri R-COUNTER_3, R_COUNTER_2 e R_COUNTER_1 e per N-COUNTER (o AB-COUNTER) si creano i registri N_COUNTER_3, N_COUNTER_2, N_COUNTER_1.
Il problema dei tre bit inviati in eccesso si risolve automaticamente, in quanto il registro di precaricamento dei dati da 21 bit è uno shift register e pertanto all'invio del 22°, 23°, 24° colpo di clock i 3 bit inviati per primi corrispondenti ai bit 21, 22, 23 vengono scartati.

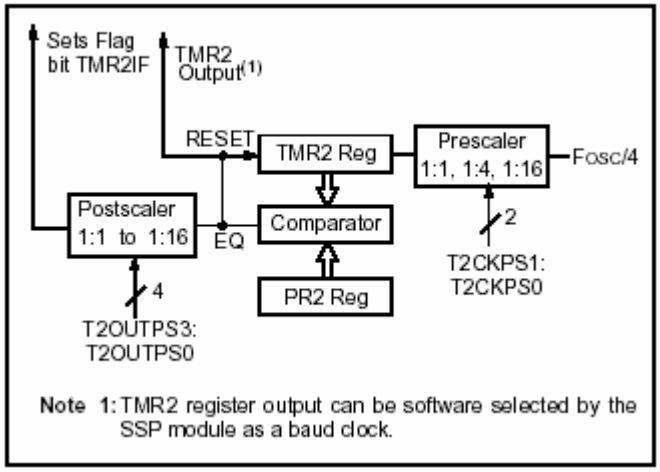
2.1.4 – Analisi della subroutine di controllo del sintetizzatore

Di seguito viene riportato il diagramma di flusso della subroutine:



Per quanto riguarda il segnale di clock si è scelto di lavorare con una frequenza molto bassa di 31 KHz, per garantire una buona qualità dei segnali SPI che viaggiano attraverso la linea.
Si è visto precedentemente che il clock viene fornito dall'uscita di uno dei tre timer disponibili del microcontrollore, in particolare dal TIMER2 per ottenere un'ulteriore divisione rispetto al valore dell'oscillatore a quarzo.
La prima operazione che il codice di controllo del sintetizzatore fa, è l'abilitazione del TIMER2 per avere disponibile la sorgente del segnale di clock. Nello schema sottostante, l'uscita che arriva al modulo SPI è quella denominata "TMR2 Output".

TIMER2 BLOCK DIAGRAM



Il controllo delle varie opzioni del timer avviene per mezzo del registro di controllo T2CON.

T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
0010 = 1:3 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

In particolare tramite il registro T2CON si accende o spegne il timer e si imposta il valore del prescaler e del postscaler.
Nell'applicazione usata il postscaler non viene utilizzato, in quanto si trova dopo l'uscita che raggiunge il modulo SPI, quindi è indifferente il suo valore. Il prescaler invece deve essere impostato per ottenere un ben specifico valore di frequenza in uscita.
E' necessario inoltre impostare il periodo di azzeramento del TIMER2 per mezzo di un altro registro definito come PR2. L'azzeramento avviene infatti ogniqualvolta il conteggio corrente del TIMER2 raggiunge lo stesso valore del registro PR2. Per impostare il periodo di azzeramento è sufficiente caricare il registro PR2 con un valore adeguato.

Se si assegna al prescaler un valore pari 1:16 e al registro PR2 un valore pari a 4 in uscita al TIMER2 trovo una frequenza di:

$$TMR2\_Output : \frac{\frac{Fosc}{4}}{16*4} = \frac{\frac{16000000}{4}}{64} = 62500Hz = 62{,}5KHz$$

La periferica SPI divide però ancora tale valore per 2, quindi la frequenza di lavoro risulta di:

$$\frac{TMR2\_Output}{2} = \frac{62500}{2} = 31250HZ \cong 31KHZ$$

Il codice programma che imposta correttamente il TIMER2 è esterno alla subroutine e risulta essere il seguente:

```
; Timer 2 for SPI clock
    BANKSEL PR2                 ; Bank 1
    MOVLW   4                   ; Set period on PR2
    MOVWF   PR2
    BANKSEL T2CON
    MOVLW   B'00000010'         ; Prescaler 1:16, Timer 2 OFF
    MOVWF   T2CON
```

Si nota l'impostazione del registro PR2 con il valore 4, l'impostazione del prescaler a 1:16 per mezzo del bit T2CKPS1 (bit 1) posto a 1.
Il codice che abilita il TIMER2 è invece:

```
; Turn on Timer2
    BANKSEL T2CON
    BSF     T2CON,TMR2ON
```

dove il bit TMR2ON del registro T2CON viene portato a 1.

Subito dopo viene effettuta l'impostazione dei due registri SSPSTAT e SSPCON per configurare il modulo MSSP in modalità SPI. L'impostazione di tale modulo viene effettuata ad ogni chiamata della subroutine, in quanto di default viene utilizzato come modulo IIC; non potendo utilizzarlo contemporaneamente in entrambi i modi è obbligatorio passare da uno all'altro ogni volta che è necessario.

```
; Change from I2C to SPI for SINTHESIZER
; Synchronous Serial Port Module for SPI BUS
    BANKSEL SSPSTAT             ; Bank 1
    MOVLW   b'01000000'         ; Data transmitted on rising edge of SCK
    MOVWF   SSPSTAT
    BANKSEL SSPCON              ; Bank 0
    MOVLW   b'00100011'         ; SPI ON, CKP=0, TMR2 output/2 CLK --> 31KHz
    MOVWF   SSPCON
```

Per quanto riguarda il registro SSPCON i bit interessati al protocollo SPI o comunque considerati nel programma steso per il microcontrollore sono:

- *SSPEN(bit 5)*:è il bit che serve ad abilitare e mettere in funzionamento la periferica SPI. Dal momento che questo bit viene posto a uno, la periferica configura opportunamente i piedini SCK, SDO, SDI rispettivamente come uscita, uscita ed entrata non badando della configurazione generale posta a inizio programma per quei determinati piedini.
- *CKP(bit 4)*:determina quale deve essere lo stato non attivo del clock. Dal diagramma di trasmissione dei dati del sintetizzatore si è visto che durante il fronte di salita il dato viene campionato mentre su quello di discesa viene cambiato, quindi questo bit va posto a zero.
- *SSPM3-SSPM0(bit3,2,1,0)*.questi bit servono per far funzionare il modulo MSSP del microcontrollore in vari modi. Bisogna scegliere un modo in funzionamento SPI Master Mode. In questo caso si opta per la configurazione 0010, la quale permette di effettuare una scelta della frequenza di clock per mezzo del timer TIMER2.

Per quanto riguarda invece il registro SSPSTAT il bit considerato per il protocollo SPI è solo uno e cioè:

- *CKE(bit 6)*: anche qui come per il bit CKP del registro precedente, si determina il modo in cui avviene la trasmissione. Volendo la trasmissione dei dati sul fronte di salita del clock e avendo posto il bit CKP a 0 ora tale bit va posto a 1.

Il bit SMP(bit 7) non va considerato, in quanto nell'applicazione del sintetizzatore di frequenza non sono presenti dati in entrata al microcontrollore e quindi è del tutto indifferente il modo di campionamento di tali dati in entrata.

```
;Chip Enable, Latch Enable
    BANKSEL PORTC               ; Bank 0
    BCF     PORTC,LE_PLL        ; LE=0
    BCF     PORTC,CE_PLL        ; CE=0
```

Con queste istruzioni non si fa altro che portare ad un livello logico 0 i due piedini esterni del microcontrollore corrispondenti alle linee LE (Latch Enable) e CE (Chip Enable). Da questo momento il sintetizzatore si trova in condizione di power down.

```
;Send first byte FUNCTION LATCH
    MOVF    FUNCTION_LATCH_3,0  ; Put bit from 23 to 16 into W
    MOVWF   SSPBUF              ; Load byte
    BANKSEL SSPSTAT             ; Bank1
    BTFSS   SSPSTAT,BF          ; End trasmission ?
    GOTO    $-1                 ; Test again
    BANKSEL SSPBUF              ; Bank 0
    MOVF    SSPBUF,0            ; Dummy reading
```

A questo punto si inviano in modalità SPI i byte di dati verso il circuito sintetizzatore. La sequenza di invio risulta uguale per tutti i byte inviati con la sola differenza che il nome del primo registro, in questo caso FUNCTION_LATCH_3, cambia ovviamente.
Le prime due istruzioni non fanno altro che copiare il contenuto del registro FUNCTION_LATCH_3 nel registro SSPBUF della periferica SPI e quindi ad iniziare immediatamente la trasmissione di questo byte verso il sintetizzatore. Infatti come detto in precedenza la sola scrittura del registro SSPBUF costituisce anche il comando di inizio trasmissione.
L'operazione successiva consiste in una operazione di attesa di fine trasmissione. Per fare ciò si fa un test ripetuto sul bit 0 del registro SSPSTAT e cioè il BF (Buffer Full). Tale bit infatti si porta in condizione logica 1 ogniqualvolta il registro SSPBUF risulta pieno e in questo caso risulta pieno dopo otto colpi di clock cioè quando l'intero byte in uscita viene completamente trasmesso al sintetizzatore.
Infine si esegue una lettura del registro SSPBUF per simulare la lettura del dato in ingresso.
Infatti se alla fine di ogni scambio di un singolo byte i dati non vengono letti dal Master, allora avviene un'interruzione della comunicazione. Quindi è obbligatorio fare una lettura del dato appena ricevuto in ogni caso, anche se nell'applicazione usata non interessa il dato ricevuto dallo Slave o più semplicemente lo Slave non prevede invio di dati come nel caso del sintetizzatore.
Da notare che le temporizzazione dei dati e del clock avvengono fatti automaticamente dal modulo Microchip MSSP senza l'ausilio di alcuna operazione da parte del programmatore.

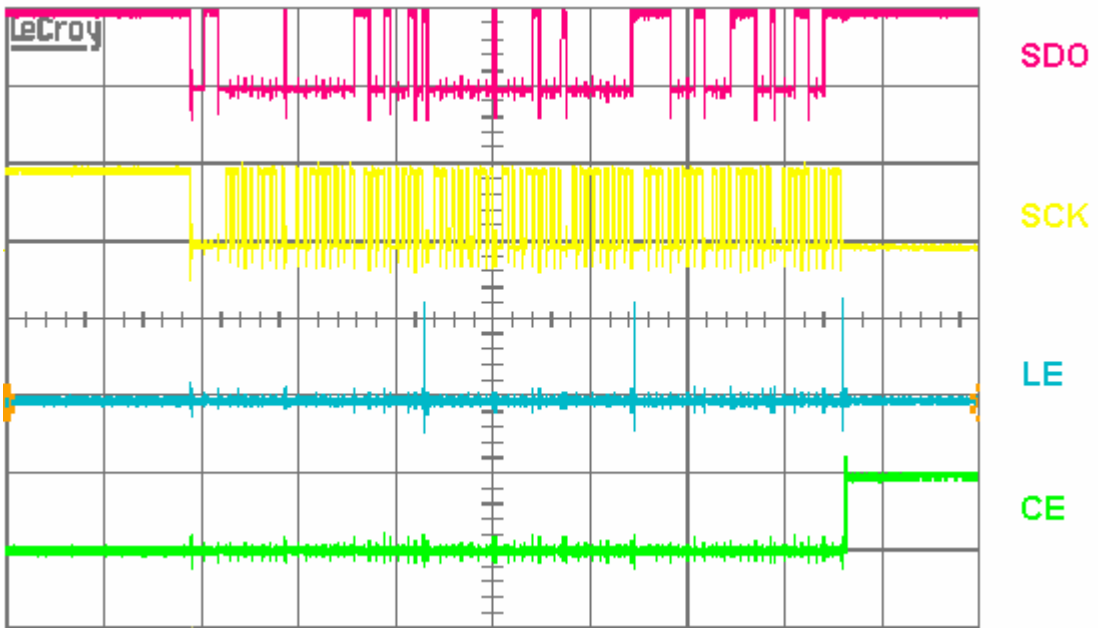
```
; Enable LE for 250 ns (at 16 MHz)
    NOP                         ; Wait for 250 ns (at 16 MHz)
    BSF     PORTC,LE_PLL        ; LE = 1
    NOP                         ; Wait for 250 ns (at 16 MHz)
    BCF     PORTC,LE_PLL        ; LE = 0
```

Inviati i tre byte allo shift register del sintetizzatore si provvede ora a trasferire questo dato a uno dei tre registri R-COUNTER, N-COUNTER, FUNCTION LATCH abilitando opportunamente il piedino del microcontrollore corrispondente alla linea LE (Latch Enable).
Come previsto dal diagramma di temporizzazione del protocollo seriale del circuito sintetizzatore, dopo l'ultimo segnale di clock si attende per almeno 10 ns e poi si porta alto il piedino LE per almeno 20 ns. In verità con il codice scritto le pause sono state rese più alte per garantire una maggior affidabilità del protocollo. La durata di entrambe le temporizzazioni è di 250 ns pari al tempo di esecuzione di una singola istruzione da parte del microcontrollore.

```
;Chip enable
    BSF     PORTC,CE_PLL        ; Enable synthesizer
;Wait 1.25 us
    NOP
    NOP
    NOP
    NOP
    NOP

; Return to I2C MODE like default
; Synchronous Serial Port Module for I2C BUS
    BANKSEL SSPSTAT             ; Bank 1
    MOVLW   b'10000000'         ; Slew rate control disable, Conform I2C levels
    MOVWF   SSPSTAT
    BANKSEL SSPCON              ; Bank 0
    MOVLW   b'00001000'         ; No collision, No overflow, Disable I2C
    MOVWF   SSPCON              ; I2C master mode, BAUD=Fosc/(4*(SSPADD + 1))

; Turn off Timer 2
    BANKSEL T2CON
    BCF     T2CON,TMR2ON
RETURN                          ; Return
```

Una volta caricati i tre registri tramite il protocollo SPI, si provvede a portare fuori dal power down il circuito sintetizzatore portando a livello logico 1 il piedino del microcontrollore corrispondente alla linea CE (Chip Enable) ed ad attendere una pausa pari a 1,25 µs per permettere ai circuiti interni del sintetizzatore di stabilizzarsi.
Infine si riporta il modulo MSSP in configurazione IIC tramite i due registri SSPSTAT e SSPCON, e si disabilita il timer TIMER2, in modo da ridurre i consumi di corrente da parte del microcontrollore.

Per verificare che effettivamente il sistema di trasmissione funzionasse perfettamente, si è visualizzata su un oscilloscopio digitale, un'intera sequenza di trasmissione SPI. Sui quattro canali dell'oscilloscopio sono state collegate le linee di SCK (Clock), SDO (Dato in uscita), LE (Latch Enable), CE (Chip Enable).
Di seguito si riporta la figura dell'intera trasmissione prelevata dall'oscilloscopio digitale usato in laboratorio.

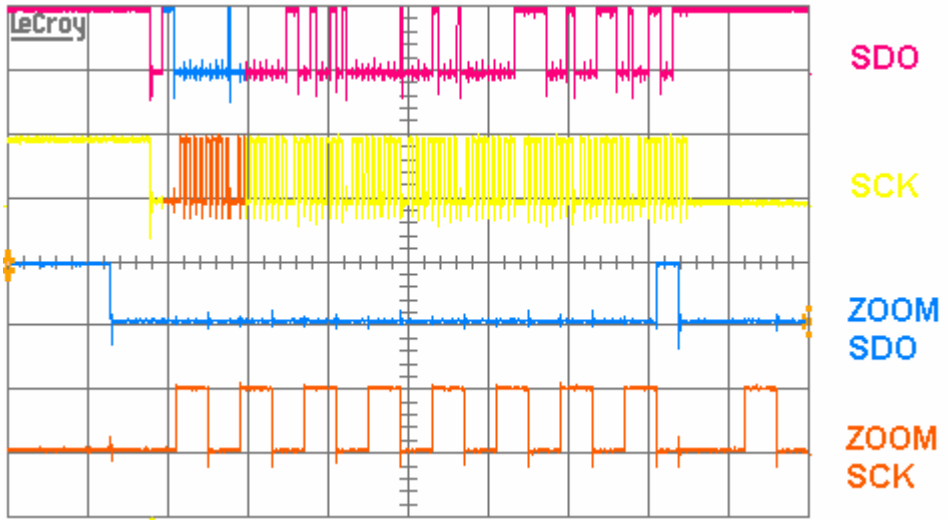


Si possono notare i nove pacchetti del segnale del clock corrispondenti ai nove byte inviati, il segnale LE che si porta alto per un istante ogni tre byte e il segnale CE che si porta alto dopo l'invio dei 9 byte.

Per verificare la correttezza dei dati inviati, si sono caricati i vari registri di lavoro di trasmissione con i seguenti valori:

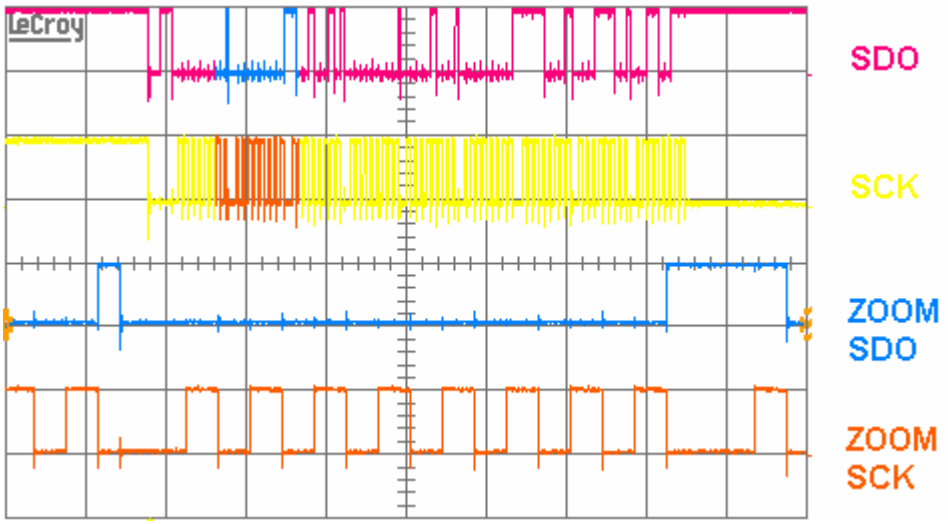
FUNCTION_LATCH_3	*00000000*
FUNCTION_LATCH_2	*00000000*
FUNCTION_LATCH_1	*10010010*
R_COUNTER_3	*00000000*
R_COUNTER_2	*00001000*
R_COUNTER_1	*00000000*
N_COUNTER_3	*11110001*
N_COUNTER_2	*00011100*
N_COUNTER_1	*00110011*

Di seguito si riportano le sequenze ingrandite della trasmissione dei vari byte ordinati nella stessa sequenza della tabella:



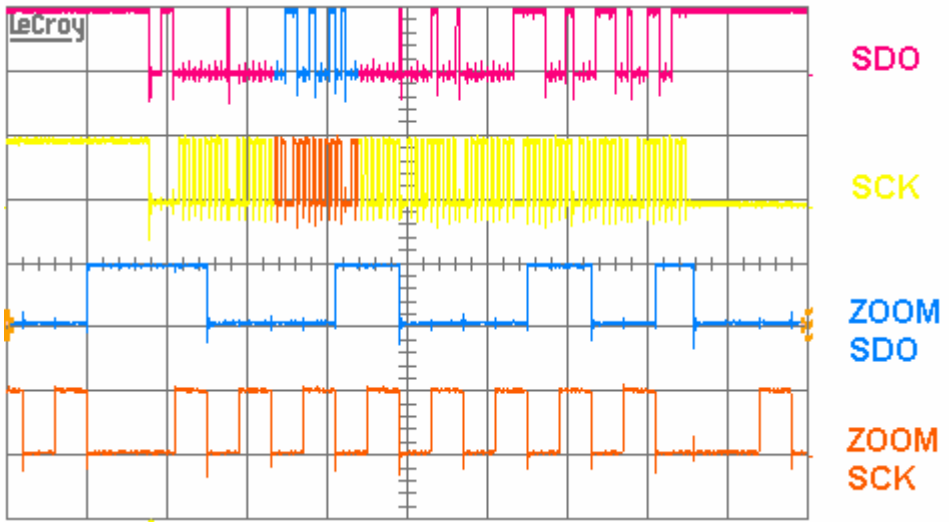
FUNCTION_LATCH_3

In corrispondenza di ogni fronte di salita degli 8 colpi di clock il segnale dati è a livello logico 0.



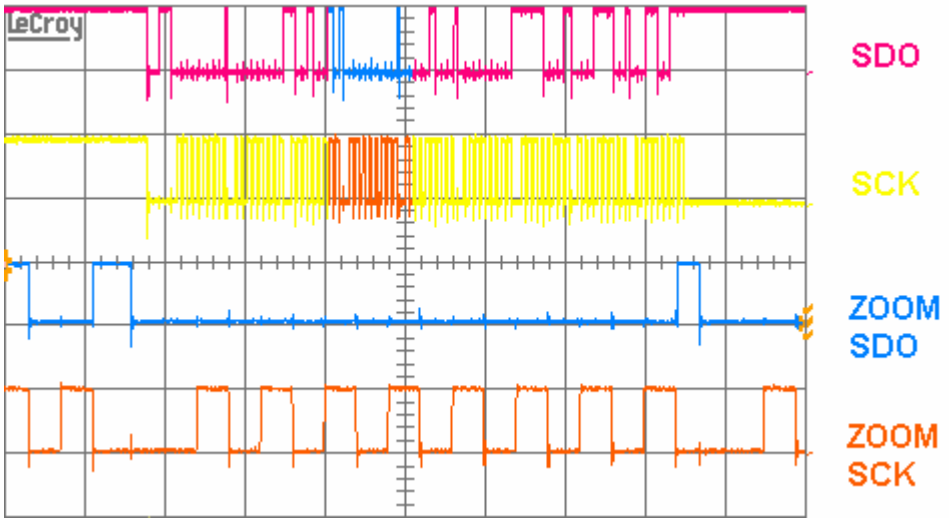
FUNCTION_LATCH_2

In corrispondenza di ogni fronte di salita degli 8 colpi di clock il segnale dati è a livello logico 0.



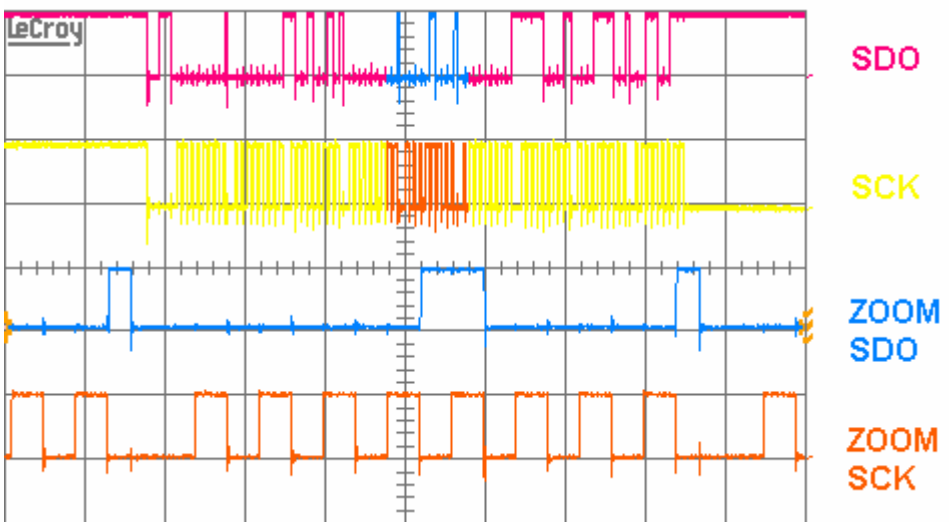
FUNCTION_LATCH_1

In corrispondenza del primo, del quarto, del settimo fronte di salita il segnale dati risulta essere a livello logico 1. Si noti il passaggio da livello logico alto a basso o viceversa nell'istante corrispondente al fronte di discesa del clock opposto appunto all'istante di campionamento.



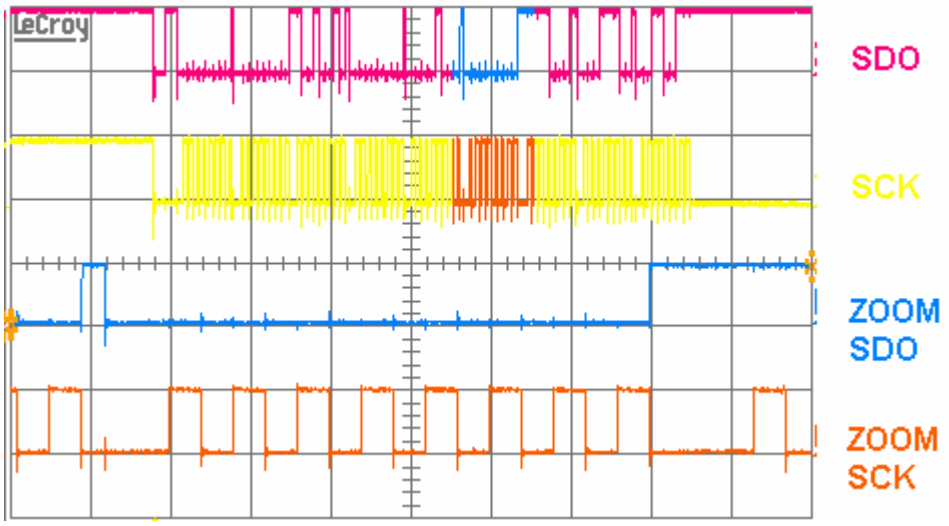
R_COUNTER_3

In corrispondenza di ogni fronte di salita degli 8 colpi di clock il segnale dati è a livello logico 0.



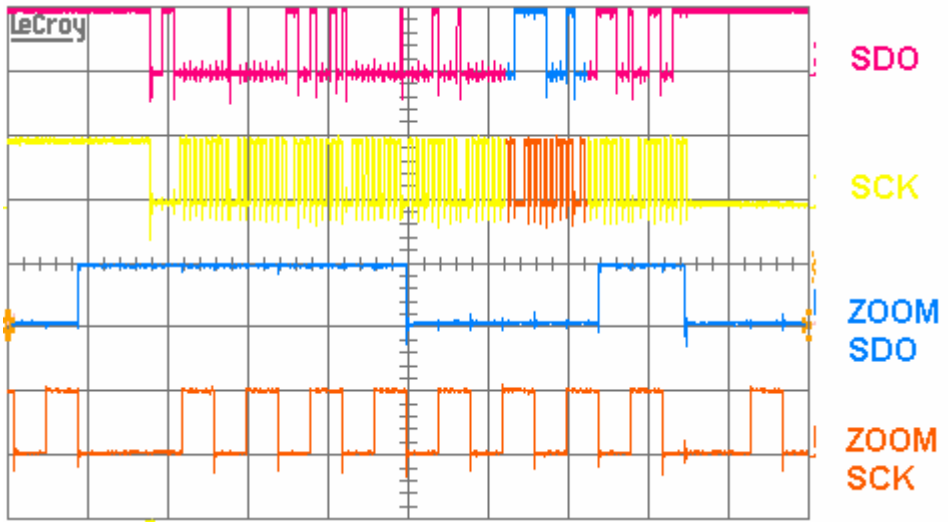
R_COUNTER_2

In corrispondenza del quinto fronte di salita del clock il segnale dati risulta a livello logico 1.



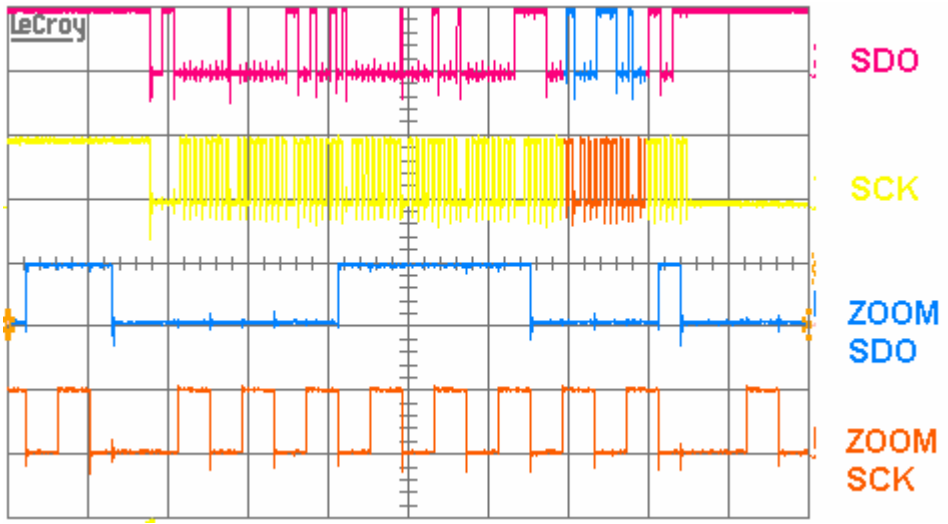
R_COUNTER_1

In corrispondenza di ogni fronte di salita degli 8 colpi di clock il segnale dati è a livello logico 0.



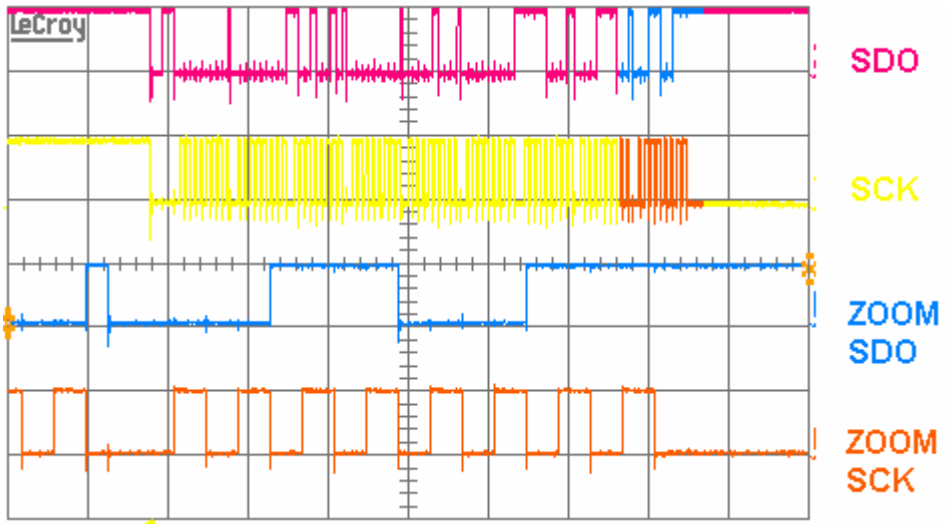
N_COUNTER_3

In corrispondenza del primo, secondo, terzo, quarto e ottavo fronte di salita si nota come il segnale dati risulta essere a livello logico 1.



N_COUNTER_2

In corrispondenza del quarto, quinto e sesto fronte di salita si nota come il segnale dati risulta essere a livello logico 1.

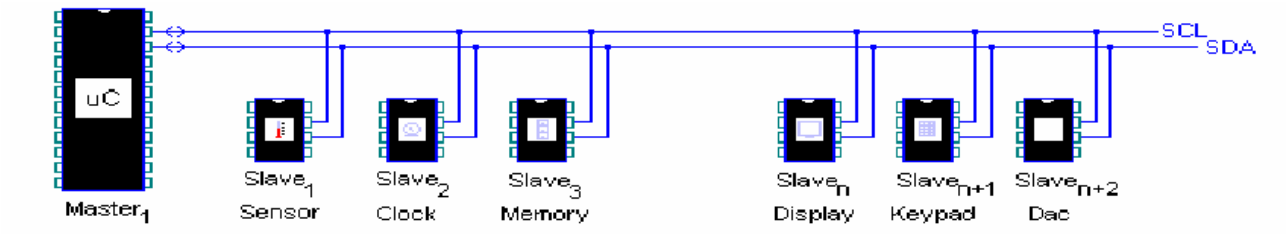


N_COUNTER_1

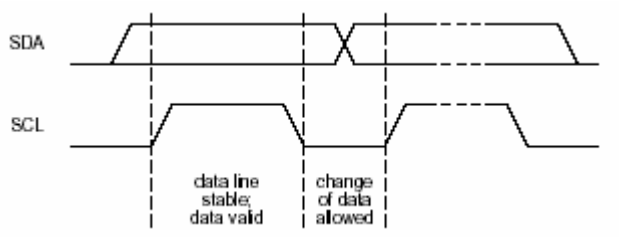
In corrispondenza del terzo, quarto, settimo e ottavo fronte di salita si nota come il segnale dati risulta essere a livello logico 1.

2.2 – Protocollo di comunicazione IIC

Il bus IIC (Inter Integreted Circuit) è stato sviluppato dalla Philips per gestire i controlli interni tra i vari circuiti integrati impiegati in un unico progetto. Quello che si vuole ottenere è la massimizzazione dell'efficienza hardware, minimizzando la complessità dei circuiti.
Tale bus ha infatti la caratteristica principale di poter collegare un elevato numero di dispositivi utilizzando solo due linee, una di dati e una di clock con l'ausilio di eventuali linee di abilitazione.
Oltre alla caratteristica di utilizzare solo due linee, questo sistema di trasmissione non utilizza alcun decodificatore per l'indirizzamento; infatti tutti i dispositivi sono indirizzabili via software e a tale scopo ogni circuito integrato ha un suo codice indirizzo assegnato dal costruttore. Tutti i dispositivi sono legati tra loro con una relazione del tipo master-slave dove il master può sia trasmettere dati sia riceverli ma in ogni caso è solo lui che può iniziare la comunicazione e gestire il segnale di clock. Di solito è presente un solo master e più slave, anche se è possibile la presenza di più master dando luogo al bus di tipo multi-master.
Sono possibili trasferimenti seriali di dati di lunghezza 8 bit a velocità di 100 kbit/s nella modalità standard o velocità dell'ordine del Mbit/s nella modalità High Speed. Questo è possibile grazie anche agli opportuni filtri antidisturbo presenti all'interno delle varie periferiche che garantiscono l'integrità dei dati.
Inoltre il numero di dispositivi che possono essere connessi sullo stesso bus è limitato solamente dalla capacità massima che il bus può raggiungere di valore 400 pF.
Il fatto di avere a disposizione circuiti che funzionano con lo standard IIC permette un rapido passaggio da uno schema a blocchi a un prototipo vero e proprio, dal momento che i circuiti integrati possono essere agganciati al bus senza ricorrere a periferiche aggiuntive di controllo. E' possibile quindi anche modificare un prototipo o semplicemente aggiornarlo, agganciando o sganciando l'integrato dal bus senza che gli altri dispositivi collegati al bus ne risentano minimamente.
Di seguito si riporta uno schema di una possibile applicazione del bus IIC con un solo master e più slave.



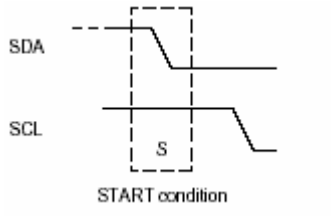
Entrambe le linee SDA e SCK sono bidirezionali e sono connesse al positivo della tensione di alimentazione per mezzo di due resistenze di pull-up. Quando il bus è libero le due linee sono a livello logico 1. Gli stadi di uscita dei vari dispositivi connessi al bus devono avere un'uscita del tipo open-collector o open-drain.
I dati sulla linea SDA deve essere stabile durante il periodo alto del clock mentre può cambiare di valore solamente nel periodo basso.



Per comprendere dettagliatamente una trasmissione secondo lo standard IIC, di seguito vengono analizzate le parti fondamentali.

Condizione di START:

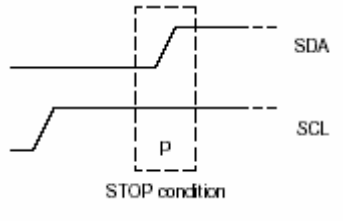
Una condizione di START indica che un dispositivo ha intenzione di trasferire dei dati nel bus. Tale condizione si verifica quando si ha una transizione della linea SDA dal livello logico 1 a livello logico 0 mentre la linea SCK si trova a livello logico 1.



Tale condizione è sempre generata dal master e dal momento che viene generata, il bus viene considerato occupato.

Condizione di STOP:

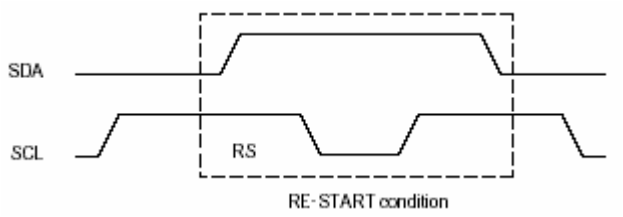
Una condizione di STOP indica che un dispositivo ha finito il trasferimento di dati sul bus. Tale condizione si verifica quando si ha una transizione della linea SDA dal livello logico 0 al livello logico 1 mentre la linea SCK si trova a livello logico 1.



Anche questa condizione viene generata esclusivamente dal master e dal momento che viene generata, il bus risulta essere libero.

Condizione di RE-START:

Questa condizione indica che un dispositivo vuole inviare ancora dati ma non vuole liberare il bus e viene fatto quando si genera una condizione di START senza che prima venga generata una condizione di STOP: questo dal punto di vista software. Dal punto di vista dei segnali sulle due linee SCK e SDA, la condizione di RE-START non è altro che uno STOP seguito immediatamente da uno START.
Una condizione di RE-START è utile anche quando sono necessarie una condizione di STOP e una condizione di START una di seguito all'altra e previene che altri dispositivi si colleghino al bus.

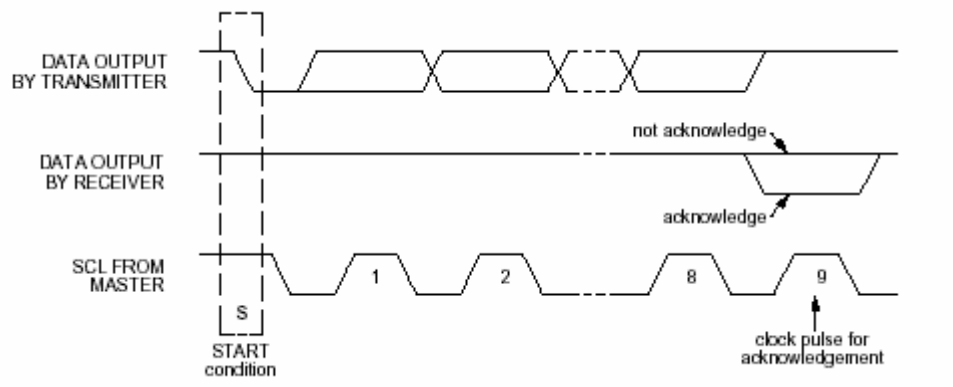


Formato del byte di dati:

Ogni byte inviato sulla linea dei dati SDA deve essere lungo 8 bit e deve essere trasmesso con il bit più significativo per primo. Il numero di byte che possono essere trasmessi invece non ha restrizione alcuna, ma ciascuno deve essere seguito da un opportuno segnale di riconoscimento detto bit di ACKNOWLEDGE.

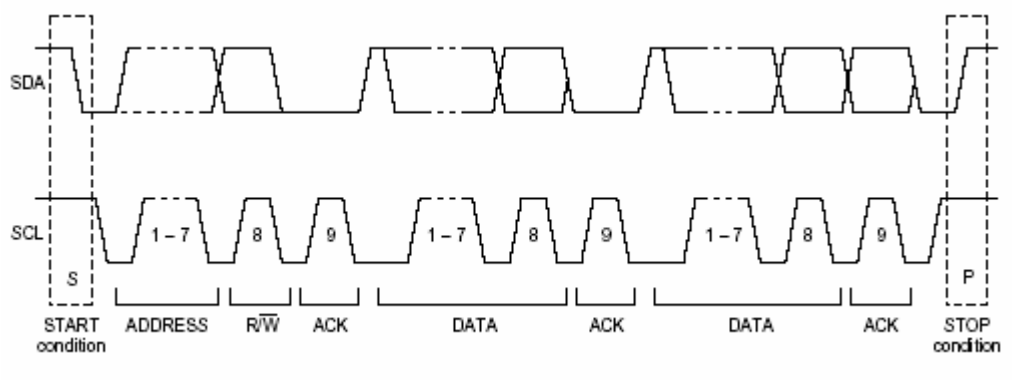
Segnale di ACKNOWLEDGE:

Il trasferimento dei dati con l'accompagnamento del segnale di acknowledge è obbligatorio. Il clock legato a tale segnale è generato dal master e il dispositivo trasmettitore, master o slave che sia, rilascia la linea dati SDA a livello logico 1.
Il ricevitore deve portare a livello logico 0 la linea dati SDA durante l'impulso di clock dell'acknowledge in modo da avere tale condizione stabile durante il suo campionamento sul fronte si salita.
Di solito, un ricevitore, che è stato indirizzato dal trasmettitore è obbligato a generare un acknowledge dopo che ogni byte è stato trasmesso.
Quando uno slave non riconosce l'indirizzo mandato dal master allora deve lasciare la linea dati a livello logico 1.
Il master quindi successivamente può generare una condizione di STOP e interrompere la trasmissione o può generare una condizione di RE-START per ricominciare la trasmissione.
Se lo slave non può ricevere i dati, allora il master deve interrompere comunque la trasmissione generando una condizione di STOP. Questo comunque è indicato dallo stesso slave che non risponde con il segnale di acknowledge dopo il primo byte inviatogli. Quando un dispositivo non genera il segnale di acknowledge si dice anche che esso genera un segnale di NOT-ACKNOWLEDGE.
Quando un dispositivo master sta ricevendo dei dati da parte del dispositivo slave, e quindi ci si trova in una condizione di master-ricevitore e slave-trasmettitore, allora il master deve segnalare l'avvenuta ricezione generando un NOT-ACKNOWLEDGE sull'ultimo byte che è stato inviato dallo slave. Lo slave trasmettitore deve allora rilasciare la linea dati SDA a livello logico 1 in modo da consentire al master di generare la condizione di STOP o eventualmente di RE-START.
Nella figura sottostante si nota parte di una sequenza di trasmissione di un dato, con la presenza del segnale di acknowledge o eventualmente not-acknowledge. Si noti in particolare la presenza del nono impulso di clock, presente sempre nel protocollo di comunicazione IIC, necessario a campionare l'acknowledge



Indirizzamento nel formato 7 bit:

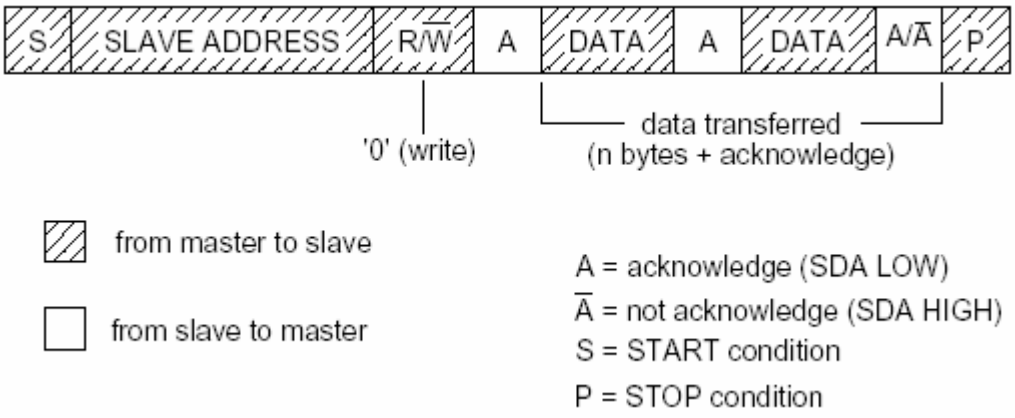
Nelle righe seguenti viene descritta una completa trasmissione dati con indirizzamento nel formato a 7 bit usata in questo progetto. Infatti in questo protocollo è possibile anche avere un indirizzamento a 10 bit nell'eventualità fossero necessari un numero elevato di dispositivi.
Di seguito si mostra un tipico trasferimento dati con indirizzamento a 7 bit:



Dopo la condizione di start, viene inviato l'indirizzo dello slave lungo appunto 7 bit, più un ottavo bit definito come R/W bit, il quale ha lo scopo di definire se si sta iniziando una sequenza di scrittura (R/W= 0) o se si sta iniziando una sequenza di lettura (R/W= 1) determinando quindi il verso del trasferimento.
Ci sono a disposizione vari tipi di combinazioni del bit R/W all'interno di una trasmissione:

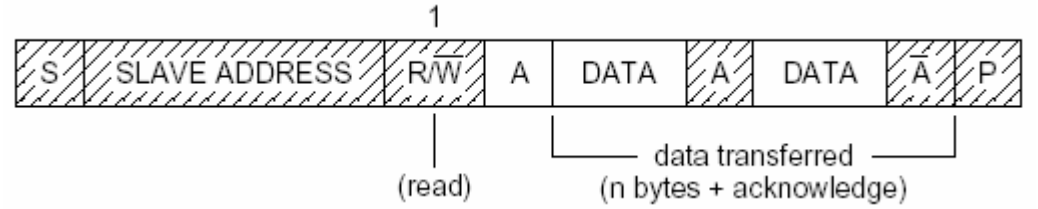
R/W= 0 (scrittura):

In questo caso il master-trasmettitore invia i dati allo slave-ricevitore e la direzione del trasferimento resta immutata per l'intera sequenza.



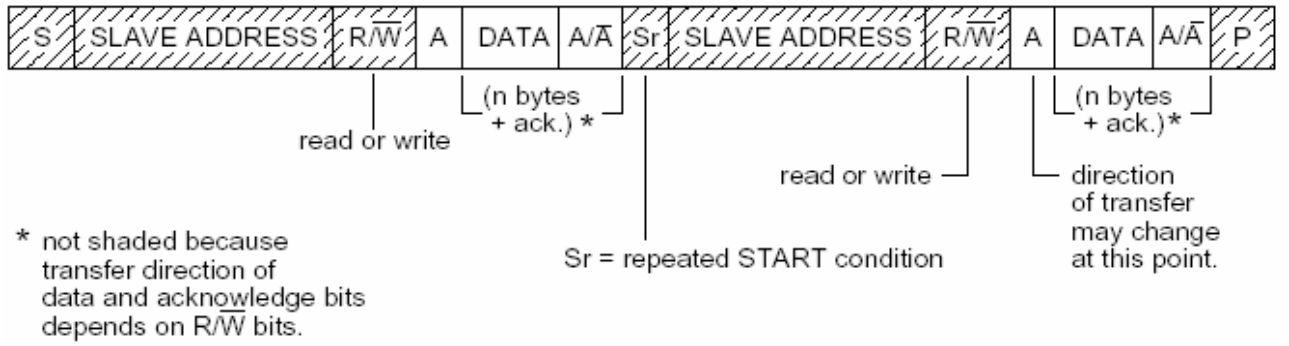
R/W= 1 (lettura):

In questo caso il master legge i dati dello slave immediatamente dopo il primo byte. Al momento del primo acknowledge il master-trasmettitore diventa un master-ricevitore e lo slave-ricevitore diventa lo slave-trasmettitore. Questo segnale di acknowledge è inviato ancora dallo slave mentre i successivi sono inviati dal master. Dopo l'ultimo byte invece il master invia un not-acknowledge.



R/W= 0 (scrittura) e R/W= 1 (lettura):

In questo caso invece si ha una combinazione dei due casi precedenti.
Durante un cambio di direzione all'interno del trasferimento, una condizione di start e un indirizzamento dello slave vengono entrambi ripetuti ma con il bit R/W invertito.



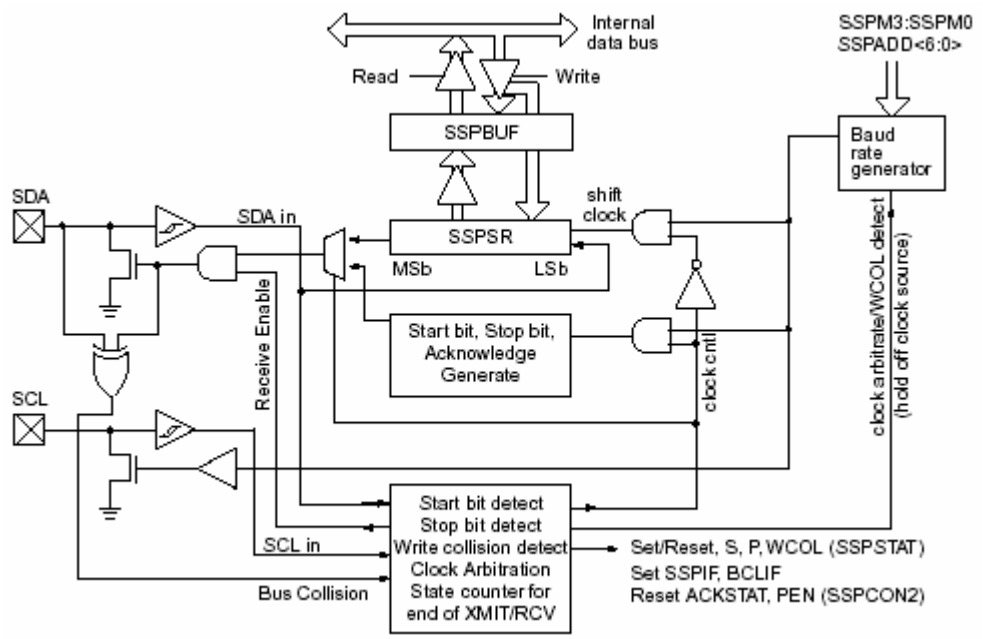
La procedura di indirizzamento per IIC bus è tale quindi che il primo byte dopo la condizione di start di solito determina quale slave sarà selezionato dal master. Questo indirizzo a 7 bit spesso è chiamato come DEVICE SELECT in quanto proprio seleziona il dispositivo con cui iniziare la comunicazione.
Quando è inviato il primo byte ogni dispositivo presente sul bus confronta i primi 7 bit con il proprio indirizzo hardware fornito dal costruttore.
Nel caso della memoria EEPROM, per esempio, il codice DEVICE SELECT è:

M24C08 Select Code	1	0	1	0	E2	A9	A8	R/W

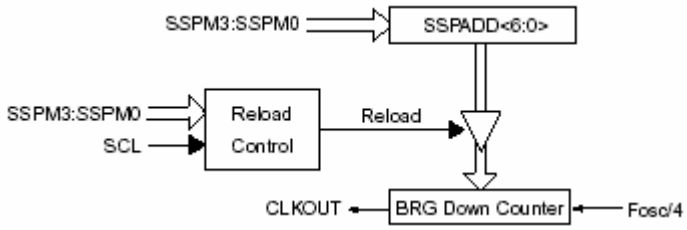
Il bit E2 lega il codice di identificazione della memoria al valore logico corrispondente al piedino di enable della memoria.
Se tale piedino è stato posto via hardware su un livello logico 0 o 1 allora affinché si possa identificare tale memoria, il bit E2 deve essere impostato rispettivamente a 0 o a 1 via software.
I bit A9 e A8 non fanno parte del codice di identificazione ma rappresentano invece i 2 bit più significativi dell'indirizzo di memoria su cui scrivere il dato.

2.2.1 – Modulo hardware MSSP Microchip in configurazione IIC

L'IIC è implementato nel microcontrollore Microchip tramite il modulo hardware definito come MSSP (Master Synchronous Serial Port).



Dal modulo Microchip MSSP in configurazione IIC, si nota la presenza del registro di lavoro SSPBUF, che ha lo scopo di fare da buffer tra il bus interno del microcontrollore e lo shift register SSPSR, necessario al trasferimento dei dati in modo seriale attraverso la linea SDA. Le condizioni di start, stop e acknowledge vengono generate da un blocco la cui uscita è multiplexata con l'uscita dello shift register SSPSR verso la linea di uscita dati SDA. Un altro blocco provvede a rilevare le condizioni di start, stop e collisioni all'interno del bus mandando a livello logico 0 o 1 alcuni bit dei registri di controllo e impostazione SSPSTAT, SSPCON e SSPCON2.
Attraverso questi ultimi tre registri è infatti possibile controllare completamente il modulo MSSP e tutte le impostazioni che riguardano la comunicazione IIC.
Essendo l'IIC un protocollo bidirezionale, si può osservare come entrambe le linee di clock SCK e dati SDA siano predisposte per il passaggio di segnali in entrambi i sensi e ciò avviene grazie alla presenza di opportune porte logiche con l'ausilio di transistor.
Il clock che sincronizza la comunicazione è scandito da un opportuno blocco denominato Baud Rate Generator di cui si riporta lo schema a blocchi.



Quando il blocco BRG è caricato con il valore dei 7 bit meno significativi del registro SSPADD allora il BRG Down Counter conta all'indietro dal valore del SSPADD fino allo zero e si ferma fino a quando non avviene un altro ricaricamento del SSPADD. Tale ricaricamento avviene automaticamente. Il clock principale di riferimento viene fornito dall'oscillatore principale con frequenza Fosc / 4.
La formula che lega il baud rate della trasmissione e il valore del registro con cui caricare il registro SSPADD è la seguente:

$$BaudRate = \frac{Fosc}{4 * (SSPADD + 1)}$$

Di seguito vengono riportati i contenuti dei registri SSPSTAT, SSPCON, SSPCON2.

SSPSTAT: SYNC SERIAL PORT STATUS REGISTER (ADDRESS: 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in slave mode
In I^2C Master or Slave mode:
1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high speed mode (400 kHz)

bit 6 CKE: SPI Clock Edge Select (Figure 9-2, Figure 9-3 and Figure 9-4)
SPI mode:
For CKP = 0
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK
For CKP = 1
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK
In I^2C Master or Slave mode:
1 = Input levels conform to SMBus spec
0 = Input levels conform to I^2C specs

bit 5 D/A: Data/$\overline{\text{Address}}$ bit (I^2C mode only)
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 P: STOP bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a STOP bit has been detected last (this bit is '0' on RESET)
0 = STOP bit was not detected last

bit 3 S: START bit
(I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.)
1 = Indicates that a START bit has been detected last (this bit is '0' on RESET)
0 = START bit was not detected last

bit 2 R/W: Read/Write bit Information (I^2C mode only)
This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next START bit, STOP bit or not $\overline{\text{ACK}}$ bit.
In I^2C Slave mode:
1 = Read
0 = Write
In I^2C Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress
Logical OR of this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSSP is in IDLE mode.

bit 1 UA: Update Address (10-bit I^2C mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

bit 0 BF: Buffer Full Status bit
Receive (SPI and I^2C modes):
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty
Transmit (I^2C mode only):
1 = Data transmit in progress (does not include the $\overline{\text{ACK}}$ and STOP bits), SSPBUF is full
0 = Data transmit complete (does not include the $\overline{\text{ACK}}$ and STOP bits), SSPBUF is empty

SSPCON: SYNC SERIAL PORT CONTROL REGISTER (ADDRESS 14h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 WCOL: Write Collision Detect bit

Master mode:

1 = A write to SSPBUF was attempted while the I2C conditions were not valid

0 = No collision

Slave mode:

1 = SSPBUF register is written while still transmitting the previous word (must be cleared in software)

0 = No collision

bit 6 SSPOV: Receive Overflow Indicator bit

In SPI mode:

1 = A new byte is received while SSPBUF holds previous data. Data in SSPSR is lost on overflow. In Slave mode, the user must read the SSPBUF, even if only transmitting data, to avoid overflows. In Master mode, the overflow bit is not set, since each operation is initiated by writing to the SSPBUF register. (Must be cleared in software.)

0 = No overflow

In I^2C mode:

1 = A byte is received while the SSPBUF is holding the previous byte. SSPOV is a "don't care" in Transmit mode. (Must be cleared in software.)

0 = No overflow

bit 5 SSPEN: Synchronous Serial Port Enable bit

In SPI mode,

When enabled, these pins must be properly configured as input or output

1 = Enables serial port and configures SCK, SDO, SDI, and SS as the source of the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

In I^2C mode,

When enabled, these pins must be properly configured as input or output

1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

bit 4 CKP: Clock Polarity Select bit

In SPI mode:

1 = Idle state for clock is a high level

0 = Idle state for clock is a low level

In I^2C Slave mode:

SCK release control

1 = Enable clock

0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In I^2C Master mode:

Unused in this mode

bit 3-0 SSPM3:SSPM0: Synchronous Serial Port Mode Select bits

0000 = SPI Master mode, clock = Fosc/4

0001 = SPI Master mode, clock = Fosc/16

0010 = SPI Master mode, clock = Fosc/64

0011 = SPI Master mode, clock = TMR2 output/2

0100 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control enabled.

0101 = SPI Slave mode, clock = SCK pin. $\overline{SS}$ pin control disabled. $\overline{SS}$ can be used as I/O pin.

0110 = I^2C Slave mode, 7-bit address

0111 = I^2C Slave mode, 10-bit address

1000 = I^2C Master mode, clock = Fosc / (4 * (SSPADD+1))

1011 = I^2C Firmware Controlled Master mode (slave idle)

1110 = I^2C Firmware Controlled Master mode, 7-bit address with START and STOP bit interrupts enabled

1111 = I^2C Firmware Controlled Master mode, 10-bit address with START and STOP bit interrupts enabled

1001, 1010, 1100, 1101 = Reserved

SSPCON2: SYNC SERIAL PORT CONTROL REGISTER2 (ADDRESS 91h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

bit 7 **GCEN**: General Call Enable bit (In I^2C Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT**: Acknowledge Status bit (In I^2C Master mode only)
In Master Transmit mode:
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT**: Acknowledge Data bit (In I^2C Master mode only)
In Master Receive mode:
Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN**: Acknowledge Sequence Enable bit (In I^2C Master mode only)
In Master Receive mode:
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence idle

bit 3 **RCEN**: Receive Enable bit (In I^2C Master mode only)
1 = Enables Receive mode for I^2C
0 = Receive idle

bit 2 **PEN**: STOP Condition Enable bit (In I^2C Master mode only)
SCK Release Control:
1 = Initiate STOP condition on SDA and SCL pins. Automatically cleared by hardware.
0 = STOP condition idle

bit 1 **RSEN**: Repeated START Condition Enable bit (In I^2C Master mode only)
1 = Initiate Repeated START condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated START condition idle

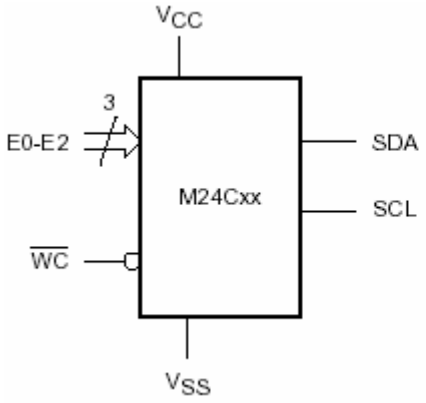
bit 0 **SEN**: START Condition Enable bit (In I^2C Master mode only)
1 = Initiate START condition on SDA and SCL pins. Automatically cleared by hardware.
0 = START condition idle

2.3 – Gestione della memoria EEPROM

Caratteristiche generali della memoria:

- capacità 8 Kbit;
- alimentazione compresa tra 4,5 e 5,5 V;
- assorbimento di corrente pari a 2 mA;
- interfaccia seriale a 2 fili (IIC compatibile);
- protezione scrittura tramite l'ingresso Write Control;
- scrittura BYTE e PAGE;
- lettura RANDOM e SEQUENZIALE;
- incremento automatico dell'indirizzo;
- circuito Power-On-Reset (POR);
- package 8 pin DIP.

Oltre le due linee, una di clock (SCK) e una di dati (SDA) utilizzate per il bus IIC, in aggiunta sono presenti una linea di abilitazione della scrittura WC (Write Control) e una linea E (Enable) per l'abilitazione dell'intero dispositivo. Il circuito interno di Power-On-Reset citato tra le caratteristiche generali provvede a mantenere la memoria in stato di reset fino a quando la tensione di alimentazione non ha raggiunto la soglia prestabilita dal POR. Questo per evitare perdite accidentali di dati.



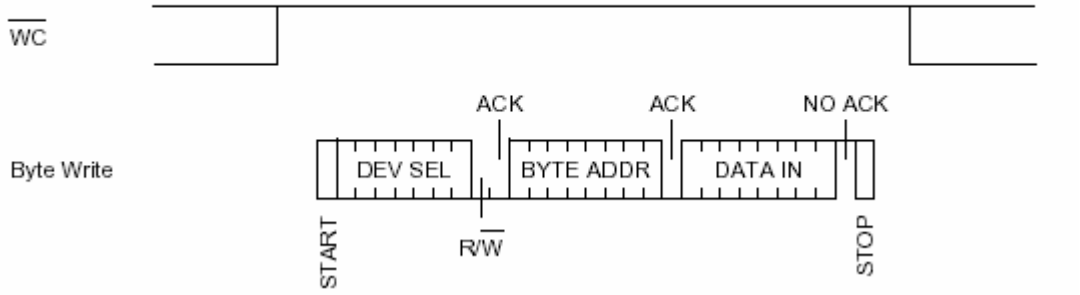
E0, E1, E2	Chip Enable
SDA	Serial Data
SCL	Serial Clock
$\overline{WC}$	Write Control
V_{CC}	Supply Voltage
V_{SS}	Ground

2.3.1 – Analisi delle subroutine di scrittura della memoria EEPROM

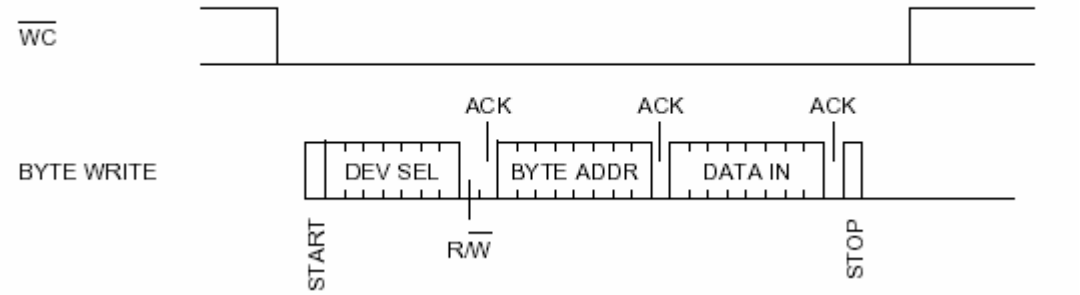
Per la gestione della memoria EEPROM sono state create due subroutine separate, una dedicata alla scrittura del dato e una dedicata alla lettura del dato.
Si comincia con la descrizione della scrittura del dato in memoria.
Questa subroutine provvede a scrivere nella memoria un singolo byte per ogni ciclo di scrittura (Byte Write), infatti tale tipo di memoria consente eventualmente anche una memorizzazione di più byte consecutivi nello stesso ciclo di scrittura (Page Write).
Dopo che il codice di Device Select e l'indirizzo su cui memorizzare il byte sono stati inviati dal master, quest'ultimo provvede inoltre a inviare il byte di dati.
Se la linea WC (Write Control) è stata portata a livello logico 1 durante il periodo compreso tra la condizione di start e la fine del byte dell'indirizzo, allora il dispositivo risponde con un not-acknowledge e il contenuto della memoria resta immutato.
Se invece la linea WC (Write Control) risulta a livello logico 0, il dispositivo risponde con un acknowledge e la scrittura viene effettuta.

Le figure riportate mettono in evidenza la sequenza di scrittura di un byte nella memoria considerando la linea Write Control prima a livello logico 1 e poi 0.

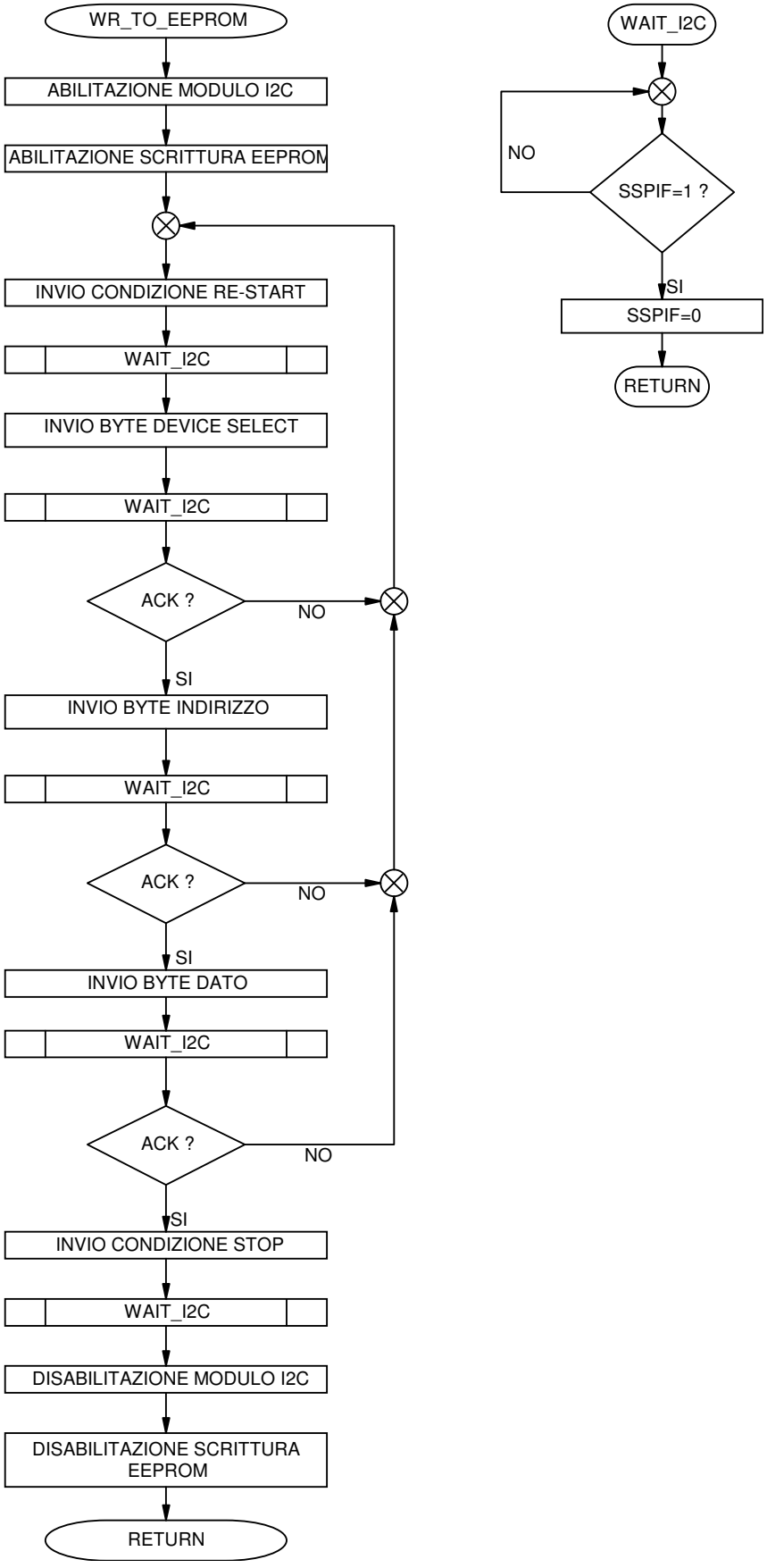
Write Mode Sequences with $\overline{WC}$=1 (data write inhibited)



Write Mode Sequences with $\overline{WC}$=0 (data write enabled)



Il diagramma di flusso seguente mostra le varie operazioni svolte dalla subroutine presente nel programma per la scrittura nella memoria eeprom.



Prima di analizzare dettagliatamente la subroutine, bisogna dire che la configurazione della periferica MSSP in modalità IIC, viene fatta non all'interno della subroutine, ma esternamente nella parte dedicata all'impostazione generale delle varie periferiche del microcontrollore. Il codice che fa queste operazioni è:

```
; Synchronous Serial Port Module for I2C BUS
    BANKSEL SSPSTAT             ; Bank 1
    MOVLW   b'10000000'         ; Slew rate control disable, Conform I2C levels
    MOVWF   SSPSTAT
    MOVLW   ( FOSC / (4 * BAUD_I2C ) ) - 1
    MOVWF   SSPADD              ; 124 (decimal)
    BANKSEL SSPCON              ; Bank 0
    MOVLW   b'00001000'         ; No collision, No overflow, Disable I2C
    MOVWF   SSPCON              ; I2C master mode, BAUD=Fosc/(4*(SSPADD + 1))
```

Per quanto riguarda il registro SSPSTAT si considerano i 2 bit più significativi:

- *SMP(bit 7):* viene posto a 1 lavorando infatti con velocità di comunicazione standard.
- *CKE(bit 6):* viene posto a 0 volendo lavorare con segnali d'ingresso conformi alle specifiche IIC.

Tramite il registro SSPCON si impostano solo i 4 bit meno significativi:

- *SSPM3_SSPM0(bit 3-0):* vengono posti rispettivamente a 1000 scegliendo così per il modulo MSSP una modalità IIC master e con un baud rate dato dalla formula Fosc/(4*(SSPADD +1)).

Essendo quindi il baud rate della comunicazione dato dalla formula,

$$BaudRate = \frac{Fosc}{4*(SSPADD+1)}$$

il valore con cui bisogna caricare il registro SSPADD è dato da:

$$SSPADD = \frac{Fosc}{4*BaudRate} - 1$$

Nel codice, il caricamento del registro SSPADD, viene fatto lasciando tale formula nel codice e definendo le costanti Fosc e BaudRate (Baud_I2C) a inizio programma, in modo da consentire una più agevole eventuale modifica.
Per il protocollo IIC è stato scelto un baud rate relativamente basso per non avere troppe distorsioni dei relativi segnali lungo la linea e cioè è stato impostato a 32 KHz.

Le prime due operazioni eseguite invece dalla subroutine di scrittura e presenti nel diagramma di flusso sono:

```
WR_TO_EEPROM

;Enable I2C and EEPROM
    BANKSEL SSPCON              ; Bank 0
    BSF     SSPCON,SSPEN        ; Enable I2C
    BCF     PORTE,ENABLE_EEPROM ; Enable EEPROM (WC)
```

Si provvede cioè ad accendere la periferica MSSP tramite l'impostazione del relativo bit di abilitazione SSPEN(bit 5) a livello logico 1 del registro SSPCON e si provvede ad abilitare la scrittura sulla memoria portando a livello logico 0 il piedino del microntrollore che comanda la linea del WC (Write Control).

A questo punto si è pronti ad occupare il bus IIC mandando la relativa condizione di START, anche se in questo caso in realtà si genera una condizione di RE-START.

```
;Send START
START_EEPROM_WR
    BANKSEL SSPCON2
    BSF     SSPCON2,RSEN        ; RE-START condition bit enable
;Wait
    CALL    WAIT_I2C
```

Se si deve infatti iniziare una comunicazione IIC, una condizione di re-start invece di una condizione di start non implica, a parte le tempistiche, nessuna differenza: infatti si ricorda che inviare un segnale di re-start è equivalente a inviare un segnale di stop e immediatamente dopo un segnale di start. Quindi in caso di bus libero la comunicazione ha inizio a tutti gli effetti. Nel caso invece il bus sia occupato, il segnale di re-start interrompe la comunicazione in corso e successivamente ne inizia un'altra.
Ciò è solo un artificio per non dover creare un'ulteriore subroutine con presente il codice di generazione della condizione di stop nel caso i segnali di acknowledge non siano rivelati. Si nota infatti dal diagramma di flusso che tutte e tre le diramazioni degli eventuali non avvenuti acknowledge vengono portate nel punto immediatamente superiore alla condizione di re-start. Se solo uno dei segnali di acknowledge non verrebbe rilevato sarebbe necessario generare uno stop e poi iniziare nuovamente la comunicazione con uno start.
La generazione dei segnali di start, re-start e stop vengono fatte automaticamente dalla periferica MSSP dal momento che si porta a livello logico 1 un determinato bit presente nel registro SSPCON2.
I tre bit meno significativi servono proprio a generare una delle tre condizioni; in particolare il bit RSEN (bit1) genera la condizione di re-start.
Successivamente all'abilitazione di tale bit viene richiamata la subroutine WAIT_I2C che fa attendere al programma il tempo necessario al completamento della condizione di re-start.
A fianco al diagramma di flusso principale è riportato il diagramma di flusso della subroutine di attesa che altro non fa che un test sul bit SSPIF (bit 3) del registro PIR1.

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag
1 = The SSP interrupt condition has occurred, and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
- SPI
  - A transmission/reception has taken place.
- I^2C Slave
  - A transmission/reception has taken place.
- I^2C Master
  - A transmission/reception has taken place.
  - The initiated START condition was completed by the SSP module.
  - The initiated STOP condition was completed by the SSP module.
  - The initiated Restart condition was completed by the SSP module.
  - The initiated Acknowledge condition was completed by the SSP module.
  - A START condition occurred while the SSP module was idle (Multi-Master system).
  - A STOP condition occurred while the SSP module was idle (Multi-Master system).

0 = No SSP interrupt condition has occurred.

Tale bit usualmente viene utilizzato in applicazioni di subroutine di interrupt e infatti il registro PIR1 è un registro dedicato a contenere vari flag di eventuali avvenuti interrupt delle varie periferiche del microcontrollore. La periferica MSSP prevede infatti in automatico la generazione di un flag, e quindi il passaggio del bit SSPIF da livello logico 0 a livello logico 1, ogniqualvolta una condizione di start, re-start, stop e avvenuto acknowledge viene completata. Per sapere quindi, quando l'operazione di re-start è terminata, basta fare un polling continuo sul bit SSPIF.

Terminata l'operazione di start si è pronti per inviare sul bus il codice dell'indirizzo Device Select più il bit di *R/W*, per fare ciò è sufficiente caricare il registro buffer SSPBUF con il valore del codice. Dal momento che il registro buffer viene caricato la trasmissione del byte avviene subito dopo.

```
;Send DEVICE SELECT
    BCF     DEV_SEL_EEPROM,0    ; First bit always to 0 to write operation
    MOVF    DEV_SEL_EEPROM,0    ; DEVICE SELECT
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
```

Si sottolinea il fatto che il bit *R/W* viene impostato automaticamente dal software indipendentemente dal valore impostato nel registro DEV_SEL_EEPROM.
Ciò è fatto ponendo a 0 il primo bit del suddetto registro, prima di copiarlo nel buffer (prima riga del codice precedente):

```
BCF     DEV_SEL_EEPROM,0    ; First bit always to 0 to write operation
```

Per sapere poi quando questa operazione è terminata allora si richiama di nuovo la subroutine WAIT_I2C per sapere quando la periferica ha riconosciuto l'avvenuto nono colpo di clock. Non ha importanza se effettivamente la memoria ha inviato l'acknowledge oppure no, quello che si verifica tramite il bit SSPIF in questo momento, è che siano stati trasmessi gli otto bit del dato più un bit di acknowledge o eventualmente not-acknowledge.
Adesso tramite questo test si verifica il valore logico campionato dal nono segnale di clock e quindi se effettivamente la memoria ha abbassato la linea dei dati SDA inviando l'acknowledge.

```
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_EEPROM_WR     ; no else...
```

Per fare ciò si controlla il valore del bit ACKSTAT del registro SSPCON2 che si porta a livello logico 0 solo se lo slave invia l'acknowledge e in caso ciò non avvenga allora tramite un salto incondizionato si ritorna ad inizio subroutine subito prima dell'invio della condizione di re-start.
Le operazioni successive inviano sul bus l'indirizzo della locazione di memoria sulla quale scrivere il dato e inviano il valore del dato stesso da memorizzare. Ciò avviene tramite le operazioni descritte precedentemente caricando opportunamente in questo caso il registro SSPBUF.

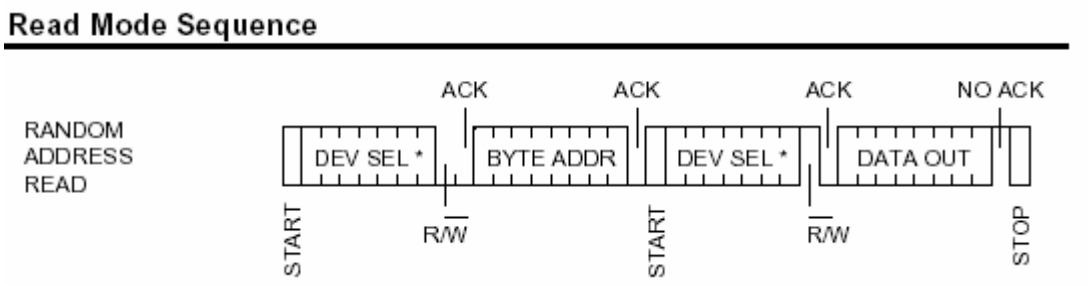
```
;Send ADDRESS
    BANKSEL ADDRESS_EEPROM      ; Bank 0
    MOVF    ADDRESS_EEPROM,0    ; ADDRESS
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_EEPROM_WR     ; no else...
;Send DATA
    BANKSEL DATA_EEPROM         ; Bank 1
    MOVF    DATA_EEPROM,0       ; BYTE
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_EEPROM_WR     ; no else...
```

Una volta effettuata la scrittura di una particolare locazione di memoria si provvede a terminare la comunicazione inviando il segnale di stop. Ciò avviene abilitando il bit PEN (bit 2) del registro SSPCON2 e successivamente richiamando la subroutine WAIT_I2C per attendere la fine dell'operazione.
Infine si spegne la periferica MSSP portando a livello logico basso il bit SSPEN (bit 5) del registro SSPCON e si disabilita la memoria dalla scrittura portando al livello logico 1 il piedino del microcontrollore corrispondente alla linea di Write Control.

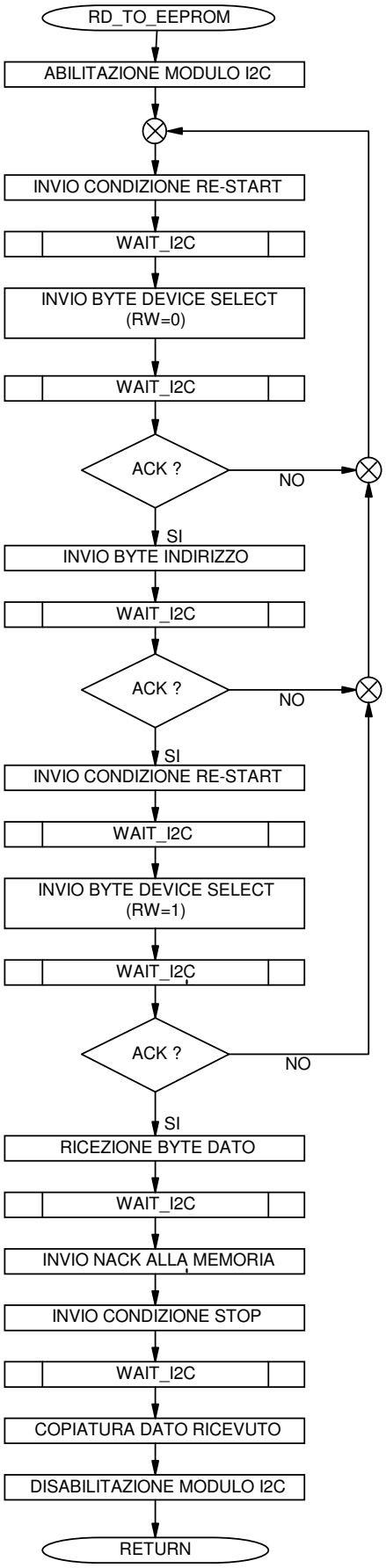
```
;Send STOP
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,PEN         ; STOP condition bit enable
;Wait
    CALL    WAIT_I2C
;Disable I2C and EEPROM
    BANKSEL SSPCON              ; Bank 0
    BCF     SSPCON,SSPEN        ; Disable I2C
    BSF     PORTE,ENABLE_EEPROM ; Disable EEPROM (WC)
RETURN
```

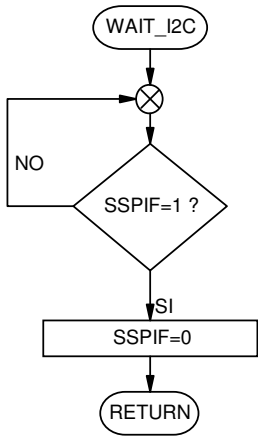
2.3.2 – Analisi della subroutine di lettura della memoria EEPROM

Al contrario di quanto accade per la scrittura della memoria, la lettura viene eseguita correttamente indipendentemente dallo stato della linea Write Control.
La memoria ha un contatore interno di indirizzo che incrementa ogni volta che un byte viene letto.
Ci sono varie modalità di lettura della memoria tra le quali quella di leggere un solo byte alla volta per ciclo di lettura partendo dall'indirizzo zero (Current Address Read), quella di leggere più byte alla volta per ciclo di lettura partendo dall'indirizzo zero (Sequential Current Read), quella di leggere un solo byte alla volta per ciclo di lettura ad una determinata locazione di memoria (Random Address Read) e quella infine di leggere più byte per ciclo di lettura partendo da una determinata locazione di memoria (Sequential Random Read).
La subroutine di lettura implementata consiste nella Random Address Read e quindi si specifica l'indirizzo del byte da voler leggere e successivamente lo si preleva.
Per fare ciò si comincia la trasmissione mandando il codice Device Select con il bit *R/W* posto a zero in modo da iniziare una scrittura fittizia e subito dopo si invia il byte di indirizzo della locazione di memoria sulla quale si vuole leggere il dato. In questo modo il contatore interno di indirizzo conterrà proprio questo valore e sarà possibile prelevare un ben determinato byte ad una determinata locazione. A questo punto si manda uno start, ovvero uno re-start, in quanto non è stato inviato alcun stop precedentemente, e successivamente di nuovo il codice Device Select, ma questa volta con il bit *R/W* posto a 1 per consentire la lettura del dato.
E' obbligatorio che dopo la trasmissione del byte da parte della memoria, il microcontrollore mandi un not-acknowledge, in quanto il byte ricevuto essendo l'unico inviato è anche l'ultimo inviato.
Di seguito si riporta una schematizzazione della descrizione appena fatta in modo da agevolare la comprensione.



Il diagramma di flusso seguente mostra le varie operazioni svolte dalla subroutine presente nel programma per la lettura della memoria eeprom.





Quasi tutte le singole parti del codice che compongono la subroutine di lettura della memoria è stato già descritto per la subroutine di scrittura. Le condizioni di start e stop, l'invio di un appropriato byte, le attese di fine esecuzione delle varie operazioni vengono create nello stesso modo di prima. Si crea la giusta sequenza di operazioni mettendo insieme le parti di codice fondamentali che creano il protocollo.
Le uniche operazioni non svolte precedentemente sono la ricezione del dato da parte della memoria slave, l'invio del segnale di acknowledge dal microcontrollore master alla memoria slave e la copiatura in un opportuno registro di salvataggio del dato trasmesso dalla memoria.
Rispettivamente i codici che eseguono tali operazioni sono:

```
;Receive DATA from slave
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,RCEN        ; Enable receive mode bit RCEN
```

Prima di ricevere i dati da parte della memoria è necessario abilitare l'appropriata funzione di ricezione portando a livello logico 1 il bit RCEN (bit 3) del registro SSPCON2. Successivamente il master si trova in condizioni di poter ricevere il dato.

```
;Send NACK to slave
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,ACKDT       ; Enable ACKDT (NACK)
    BSF     SSPCON2,ACKEN       ; Start NACK
```

Tramite il bit ACKDT (bit 5) del registro SSPCON2 si imposta se mandare un acknowledge o un not-acknoledge e in questo caso volendo ottenere il secondo bisogna impostare a 1 tale bit.
Tramite invece l'abilitazione del bit ACKEN (bit 4) si inizia l'operazione di acknowledge o not-acknowledge a seconda del bit ACKDT e in questo caso di not-acknowledge.
Si ricorda infatti che ora è il master a inviare un segnale di acknowledge, in questo caso di not-acknowledge, e non lo slave, in quanto la trasmissione va dallo slave al master dato che il bit *R/W* è a 1. L'impostazione del bit *R/W* è fatto automaticamente dal software indipendentemente da che valore è stato impostato nel registro, come avveniva nella subroutine di scrittura.

```
;Save received data
    BANKSEL SSPBUF
    MOVF    SSPBUF,0
    MOVWF   DATA_EEPROM
```

Il dato ricevuto presente nel registro buffer viene copiato infine in un registro creato appositamente in modo da poter essere elaborato successivamente.

Si riporta il codice completo di lettura dalla memoria EEPROM.

```
RD_FROM_EEPROM
;Enable I2C and EEPROM
    BANKSEL SSPCON              ; Bank 0
    BSF     SSPCON,SSPEN        ; Enable I2C

START_EEPROM_RD
;Send STOP
    BANKSEL SSPCON2              ; Bank 1
    BSF     SSPCON2,RSEN         ; START condition bit enable
;Wait
    CALL    WAIT_I2C
;Send DEVICE SELECT
    BCF     DEV_SEL_EEPROM,0    ; First bit always to 0 to write operation
    MOVF    DEV_SEL_EEPROM,0    ; DEVICE SELECT
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_EEPROM_RD     ; no else...
;Send ADDRESS
    BCF     STATUS,RP0
    MOVF    ADDRESS_EEPROM,0    ; ADDRESS
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_EEPROM_RD     ; no else...
;Send START
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,RSEN        ; RE-START condition bit enable
;Wait
    CALL    WAIT_I2C
;Send DEVICE SELECT
    BCF     STATUS,RP0
    BSF     DEV_SEL_EEPROM,0    ; First bit always to 1 to read operation
    MOVF    DEV_SEL_EEPROM,0    ; DEVICE SELECT
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_EEPROM_RD     ; no else...
;Receive DATA from slave
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,RCEN        ; Enable receive mode bit RCEN
;Wait
    CALL    WAIT_I2C
```

```
;Send NACK to slave
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,ACKDT       ; Enable ACKDT (NACK)
    BSF     SSPCON2,ACKEN       ; Start NACK
;Send STOP
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,PEN         ; STOP condition bit enable
;Wait
    CALL    WAIT_I2C
;Save received data
    BANKSEL SSPBUF
    MOVF    SSPBUF,0
    MOVWF   DATA_EEPROM
;Disable I2C and EEPROM
    BANKSEL SSPCON              ; Bank 0
    BCF     SSPCON,SSPEN        ; Disable I2C
RETURN
```

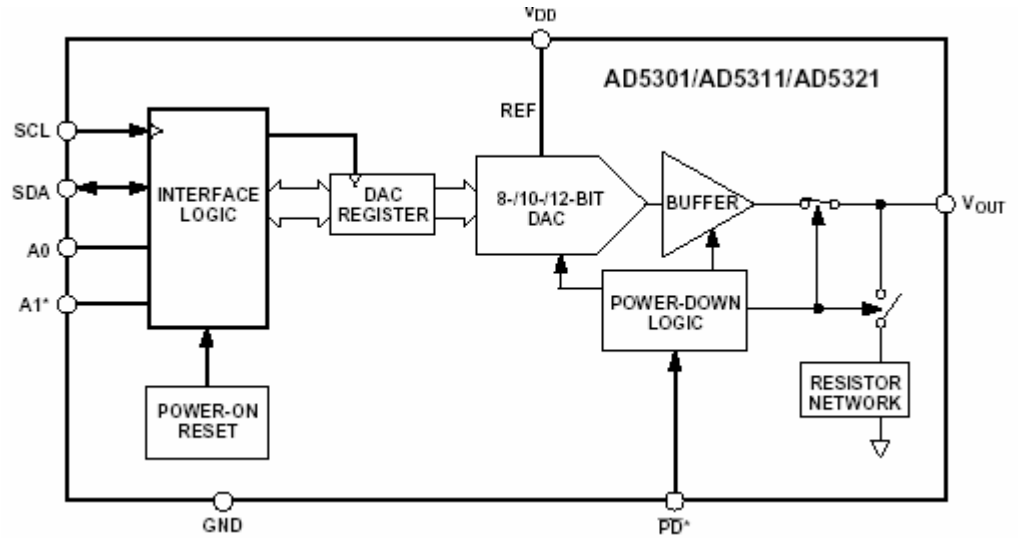
2.3.3 – Architettura hardware per la memoria EEPROM

Per quanto riguarda l'architettura hardware impiegata per il bus di comunicazione si rimanda il lettore al paragrafo dedicato, presente successivamente.
Per quanto riguarda invece l'alimentazione della memoria, essa non viene prelevata direttamente dall'alimentazione della scheda ma viene filtrata da una resistenza da 10 Ω posta in serie e mediante un condensatore da 470 µF. Tale accorgimento è stato introdotto dal momento che si sono verificati dei malfunzionamenti della memoria, come ad esempio la cancellazione imprevista e indesiderata di dati, dovuti all'improvvisa applicazione della tensione di alimentazione.

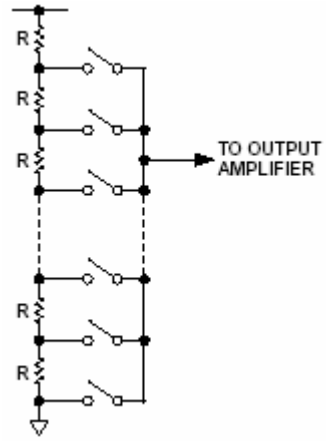
2.4 – Conversione digitale – analogica

Caratteristiche generali del convertitore:

- conversione con risoluzione a 12 bit;
- alimentazione compresa tra 2,5 e 5,5 V;
- assorbimento di corrente pari a 120 μA (a 3 V);
- interfaccia seriale a 2 fili (IIC compatibile);
- riferimento di conversione pari alla tensione di alimentazione;
- circuito di Power-On-Reset;
- buffer d'uscita rail to rail;
- tre diverse funzioni di power down con assorbimento di 50 nA (a 3 V);
- package a 8 piedini μSOIC.



Lo schema mostra i blocchi principali che costituiscono il convertitore digitale-analogico. Si nota la presenza di un'interfaccia logica che gestisce gli ingressi dedicati al controllo tramite protocollo IIC; tra questi sono presenti le due linee che si agganciano al bus, SDA e SCK, e due ingressi di identificazione del dispositivo A0 e A1 che determinano lo stato di due bit del codice di Device Select.



I valori dei vari bit dedicati alla conversione sono trasferiti al registro di conversione Dac Register che provvede a gestire il blocco che forma il vero convertitore. Infatti quest'ultimo consiste in una fila di resistenze collegate in serie in modo da formare un partitore di tensione. Su ogni nodo è presente un interruttore che provvede a collegare un determinato punto della rete verso l'uscita.

Il codice digitale caricato nel Dac Register determina proprio quale di questi interruttori deve essere chiuso in modo da ottenere in uscita una determinata tensione direttamente proporzionale. Il valore della tensione di riferimento della rete viene prelevato proprio da quello di alimentazione.

A questo punto il valore di tensione prelevato dalla rete resistiva viene inviato all'ingresso di un buffer separatore il quale consente di prelevare tensioni in un range da 1 mV a Vdd – 1mV.

Dallo schema a blocchi si nota inoltre la presenza di un piedino denominato PD che consente di mandare il dispositivo in stato di power down. Quando il dispositivo è mandato in power down tramite questo piedino, l'uscita viene posta in una condizione di Three State. In tale condizione, se il dispositivo è alimentato a 5 V, i consumi si mantengono entro i 200 nA.

È possibile mandare il dispositivo in power down anche via software, configurando opportunamente due bit presenti in uno dei registri configurabili tramite IIC, ma ciò verrà analizzato più dettagliatamente dopo quando si descriveranno le funzioni di tali registri.

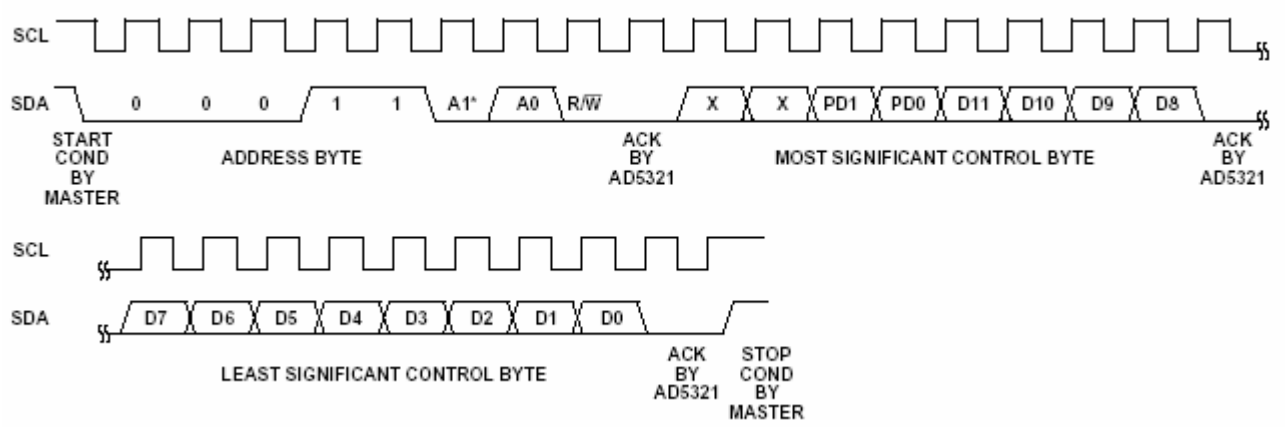
Per calcolare quale tensione si preleva dall'uscita del convertitore per una data configurazione binaria in entrata, è possibile sfruttare la seguente formula:

$$V_{OUT} = \frac{V_{DD} * D}{2^N}$$

dove N indica la risoluzione del DAC (in questo caso 12), D indica il numero equivalente decimale del codice binario a 12 bit.

2.4.1 – Gestione del convertitore e analisi della subroutine di controllo

Il dispositivo consente oltre che operazioni di scrittura, anche operazioni di lettura. Infatti nella fase di scrittura si inviano i dati in modo da ottenere in uscita una ben determinata tensione e nella fase di lettura è possibile ricevere gli stessi dati inviati precedentemente in modo da effettuare per esempio un'operazione di controllo.
Per la gestione di questo convertitore è stata creata solamente una subroutine di scrittura, in modo da non effettuare alcuna lettura di ritorno.
Come tutti i dispositivi IIC compatibili è necessario identificare il destinatario dei dati inviando per primo un byte di indirizzo Device Select. Per questo convertitore i cinque bit più significativi del codice di Device Select sono "00011", mentre i tre meno significativi sono determinati dallo stato degli ingressi A0, A1 e dal bit *R/W* che va posto a 0 in modo da consentire una scrittura.
I due byte inviati successivamente, denominati MSB CONTROL BYTE e LSB CONTROL BYTE, oltre a impostare i 12 bit da voler convertire, consentono, tramite altri due bit dedicati PD1 e PD0, di selezionare il modo di funzionamento del dispositivo.
Di seguito si riporta una sequenza di scrittura completa.

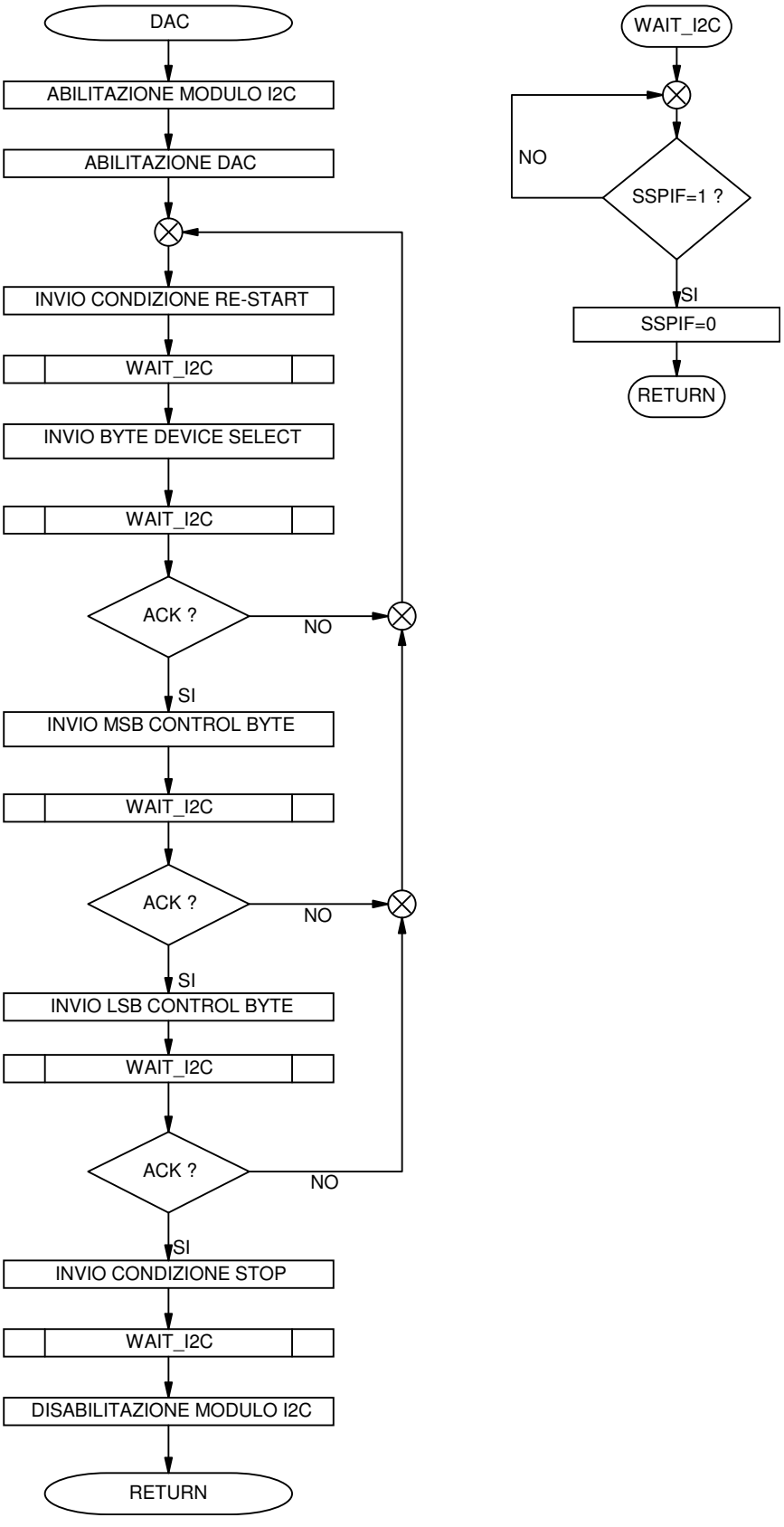


Le varie combinazioni dei bit PD1 e PD0 consentono di far lavorare il convertitore in modo normale oppure di mandarlo in tre condizioni di power down diverse tra loro per quanto riguarda la configurazione hardware dell'uscita:

PD1	PD0	Operating Mode
0	0	Normal Operation
0	1	Power-Down (1 kΩ Load to GND)
1	0	Power-Down (100 kΩ Load to GND)
1	1	Power-Down (Three-State Output)

infatti con il primo modo è possibile interporre tra l'uscita e massa una resistenza da 1KΩ, con il secondo modo è possibile interporre tra l'uscita e massa una resistenza da 100KΩ e infine con il terzo modo è possibile lasciare l'uscita in condizione di Three State.
È da tener presente che qualsiasi delle quattro modalità di funzionamento viene ignorata se si manda il dispositivo in power down via hardware tramite il piedino PD.

Il diagramma di flusso seguente mostra le varie operazioni svolte dalla subroutine presente nel programma per la scrittura del convertitore.



Come si può notare il diagramma di flusso è praticamente uguale a quello di gestione della scrittura della memoria eeprom e tutte le operazioni sono le stesse descritte precedentemente. Le uniche cose che differenziano i diagrammi sono i tre byte inviati e la non disabilitazione del dispositivo IIC a fine subroutine. Infatti se si provvede a disabilitare il dispositivo l'uscita andrebbe in alta impedenza e si preleverebbe una tensione di 0 volt costante in uscita. Quindi una volta effettuta la conversione, se si desidera spegnere il dispositivo, è necessario pilotare a livello logico 0 la linea di PD successivamente.
C'è da dire inoltre che ogniqualvolta si vuole aggiornare il valore di tensione in uscita è necessario richiamare tale subroutine effettuando quindi nuovamente l'indirizzamento del dispositivo tramite il Device Select. Il dispositivo consente anche l'aggiornamento dei dati effettuando un solo indirizzamento: dopo l'invio del Device Select ogni multiplo di due byte sarà identificato come i due MSB e LSB CONTROL BYTE. In ogni modo tale modalità di funzionamento non è stata implementata in questo caso ma può essere sfruttata in una eventuale revisione futura del software.
Di seguito si riporta il codice che effettua la conversione digitale analogica.

```
DAC
;Enable I2C and DAC
    BANKSEL SSPCON              ; Bank 0
    BSF     SSPCON,SSPEN        ; Enable I2C
    BSF     PORTE,ENABLE_DAC    ; Enable DAC (PD)
;Send STOP
START_DAC
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,RSEN        ; RE-START condition bit enable
;Wait
    CALL    WAIT_I2C
;Send DEVICE SELECT
    MOVF    DEV_SEL_DAC,0       ; DEVICE SELECT
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_DAC           ; no else...
;Send DATA_MSB
    BANKSEL DAC_MSB             ; Bank 0
    MOVF    DAC_MSB,0           ; MSB BYTE
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_DAC           ; no else...
;Send DATA_LSB
    BANKSEL DAC_LSB             ; Bank 0
    MOVF    DAC_LSB,0           ; LSB BYTE
    BANKSEL SSPBUF              ; Bank 0
    MOVWF   SSPBUF              ; Load byte
;Wait
    CALL    WAIT_I2C
;Check ACK from slave
    BANKSEL SSPCON2             ; Bank 1
    BTFSC   SSPCON2,ACKSTAT     ; received ACK from slave ?
    GOTO    START_DAC           ; no else...
;Send STOP
    BANKSEL SSPCON2             ; Bank 1
    BSF     SSPCON2,PEN         ; STOP condition bit enable
;Wait
    CALL    WAIT_I2C
;Disable I2C and DAC
    BANKSEL SSPCON              ; Bank 0
    BCF     SSPCON,SSPEN        ; Disable I2C
RETURN
```

2.4.2 – Architettura hardware per il convertitore digitale - analogico

Le accortezze hardware per quanto riguarda il bus di comunicazione vengono spiegate nel paragrafo successivo.
Per quanto riguarda la linea di uscita del convertitore è stata posta una resistenza da 10 KΩ con in parallelo un condensatore da 100 nF per costituire un filtro passa basso.
Per eliminare la radiofrequenza in parallelo all'alimentazione dell'integrato è presente inoltre un condensatore da 100 nF.

2.4.3 – Architettura hardware per il protocollo SPI e IIC

Come visto il modulo interno di generazione di questi due protocolli viene condiviso e per tale motivo la linea del clock risulta essere la stessa usata sia per il protocollo SPI e sia per il protocollo IIC.
Dallo schema elettrico infatti è possibile constatare che tale linea fa capo sia ai due dispositivi IIC e sia al sintetizzatore di frequenza.
In ogni modo tale vincolo non pregiudica affatto il funzionamento dei vari dispositivi, in quanto, oltre che funzionando con protocolli diversi, il modulo MSSP del microcontrollore non può per sua progettazione funzionare contemporaneamente nei due diversi modi.
L'unico accorgimento hardware per questi due protocolli risulta quello di implementare una resistenza di pull-up per ogni linea. Visto il numero ridotto di dispositivi presenti, la buona qualità dei fronti dei segnali di comunicazione verificati con l'oscilloscopio e l'effettivo funzionamento dei bus, si è visto che il valore di resistenza di 33 KΩ è risultato adeguato.

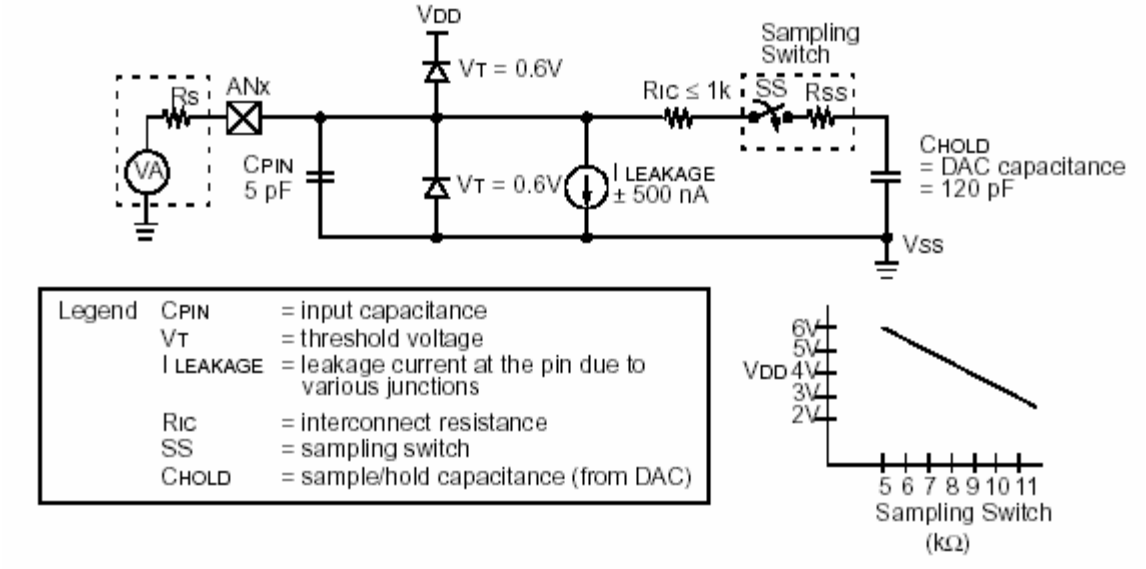
2.5 – Conversione analogica-digitale

La conversione analogica – digitale viene fatta sfruttando l'apposito modulo A/D presente all'interno del microcontrollore e non è quindi necessaria l'aggiunta di alcuna periferica esterna.
Tale microcontrollore permette l'acquisizione contemporanea su otto ingressi differenti usando un solo convertitore, grazie alla presenza di un multiplexer.
L'operazione di multiplexing viene fatta automaticamente dal microcontrollore in modo del tutto trasparente rispetto al programmatore. Infatti il programmatore scrive il relativo codice come se ci fossero all'interno ben otto convertitori indipendenti l'uno dall'altro.
L'ingresso analogico carica un condensatore che funge da circuito sample e hold e la cui uscita fa capo al convertitore. Quest'ultimo poi genera il risultato per approssimazioni successive in un numero digitale a 10 bit.
Le tensioni di riferimento alta e bassa del convertitore possono essere anche prelevate dall'esterno.
Affinché il convertitore possa effettuare un'accurata conversione, è necessario che il condensatore Chold si carichi completamente al livello della tensione proveniente dall'ingresso analogico.
L'impedenza Rs della sorgente e l'impedenza interna Rss del circuito che esegue il campionamento influenzano direttamente il tempo richiesto per la carica del condensatore Chold e quindi l'acquisizione. A tale scopo si raccomanda che l'impedenza Rs per la sorgente analogica non superi i 10KΩ.
Dopo che si seleziona il canale sul quale effettuare la conversione bisogna attendere il tempo di acquisizione prima che la conversione possa partire.
In particolare il tempo viene calcolato considerando il tempo di stabilizzazione dell'amplificatore interno, il tempo di carica del condensatore Chold e un coefficiente di temperatura:

$$
\begin{aligned}
T_{ACQ} &= \text{Amplifier Settling Time} + \\
&\quad \text{Hold Capacitor Charging Time} + \\
&\quad \text{Temperature Coefficient} \\
&= T_{AMP} + T_C + T_{COFF} \\
&= 2\mu s + T_C + [(\text{Temperature} - 25^\circ C)(0.05\mu s/^\circ C)] \\
T_C &= C_{HOLD}(R_{IC} + R_{SS} + R_S)\ \ln(1/2047) \\
&= -120pF\,(1k\Omega + 7k\Omega + 10k\Omega)\ \ln(0.0004885) \\
&= 16.47\mu s \\
T_{ACQ} &= 2\mu s + 16.47\mu s + [(50^\circ C - 25^\circ C)(0.05\mu s/^\circ C) \\
&= 19.72\mu s
\end{aligned}
$$

Dal datasheet del microcontrollore il tempo di acquisizione viene standardizzato a 19,72µs.

ANALOG INPUT MODEL



Il tempo di conversione per ogni singolo bit viene definito come TAD. Il convertitore richiede un tempo minimo pari a 12*TAD per effettuare la conversione completa.
La sorgente del clock usata per la conversione è selezionabile via software e deve essere scelta opportunamente per assicurare un minimo tempo TAD di 1,6 μs.
Il clock usato per la conversione determina con quale velocità avviene il campionamento e quindi bisogna sceglierlo opportunamente in base alla frequenza del segnale d'ingresso.
A tal proposito è disponibile la seguente tabella:

AD Clock Source (TAD)		Maximum Device Frequency
Operation	ADCS1:ADCS0	Max.
2TOSC	00	1.25 MHz
8TOSC	01	5 MHz
32TOSC	10	20 MHz
RC(1, 2, 3)	11	(Note 1)

Il modulo convertitore AD viene impostato per mezzo di due registri denominato ADCON0 e ADCON1 di cui si riportano di seguito i contenuti:

ADCON0 REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0**: A/D Conversion Clock Select bits
00 = Fosc/2
01 = Fosc/8
10 = Fosc/32
11 = FRC (clock derived from the internal A/D module RC oscillator)

bit 5-3 **CHS2:CHS0**: Analog Channel Select bits
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN4)
101 = channel 5, (RE0/AN5)(1)
110 = channel 6, (RE1/AN6)(1)
111 = channel 7, (RE2/AN7)(1)

bit 2 **GO/$\overline{\text{DONE}}$**: A/D Conversion Status bit
If ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion)
0 = A/D conversion not in progress (this bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1 **Unimplemented**: Read as '0'

bit 0 **ADON**: A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shut-off and consumes no operating current

Tramite il registro ADCON0 si seleziona il clock di campionamento e il canale sul quale effettuare la conversione. Inoltre è possibile accendere o spegnere il modulo A/D e dare inizio alla conversione.

ADCON1 REGISTER (ADDRESS 9Fh)

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. 6 Most Significant bits of ADRESH are read as '0'.
0 = Left justified. 6 Least Significant bits of ADRESL are read as '0'.

bit 6-4 **Unimplemented:** Read as '0'

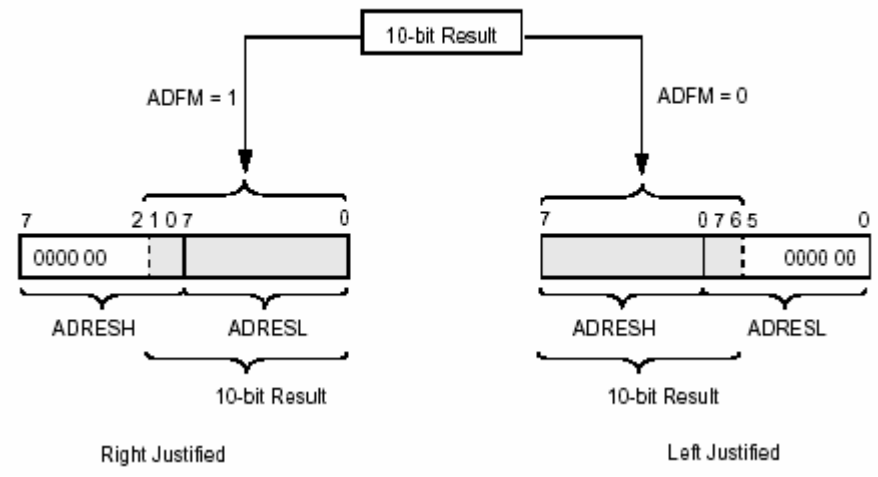
bit 3-0 **PCFG3:PCFG0**: A/D Port Configuration Control bits:

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs(2)
0000	A	A	A	A	A	A	A	A	VDD	Vss	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	Vss	7/1
0010	D	D	D	A	A	A	A	A	VDD	Vss	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	Vss	4/1
0100	D	D	D	D	A	D	A	A	VDD	Vss	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	Vss	2/1
011x	D	D	D	D	D	D	D	D	VDD	Vss	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	Vss	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	Vss	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	Vss	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

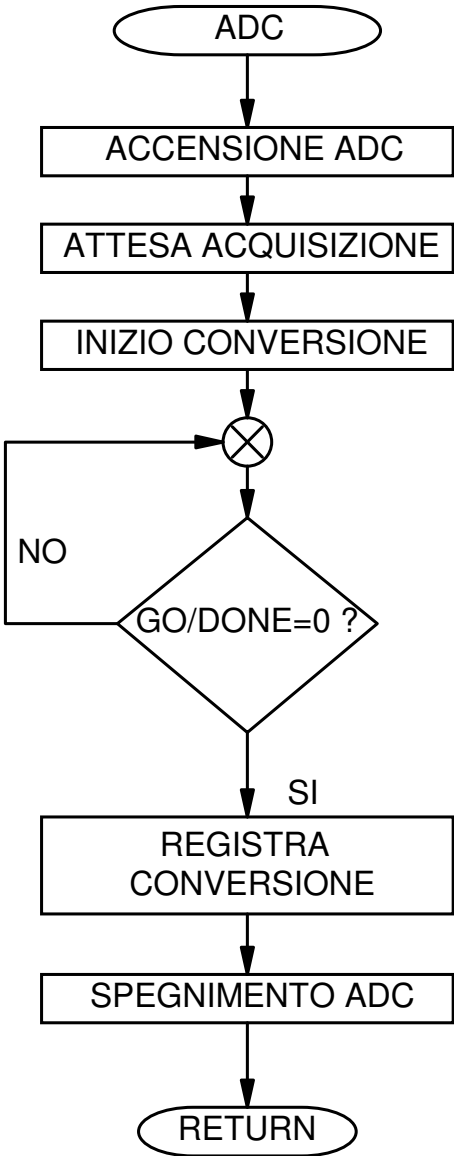
A = Analog input D = Digital I/O

Con il registro ADCON1 si impostano invece opportunamente le porte A e E del microcontrollore per poter usare i singoli piedini come analogici o digitali in base alle esigenze. Si possono scegliere varie combinazioni nelle quali tra l'altro è possibile selezionare le sorgenti delle tensioni di riferimento positiva e negativa. Tali tensioni infatti si possono prelevare sia dall'interno del microcontrollore sia dall'esterno.
Tramite questo registro si controlla anche in che modo il risultato digitale convertito viene salvato: a tal scopo sono disponibili due altri registri ADRESL e ADRESH nei quali è presente il valore binario a 10 bit.
Le due alternative disponibili consentono di salvare gli otto bit meno significativi nel registro ADRESL e i due restanti più significativi nel registro ADRESH oppure gli otto più significativi nel registro ADRESH e i due meno significativi nel registro ADRESL.

A/D RESULT JUSTIFICATION



2.5.1 – Gestione del convertitore e analisi della subroutine di controllo



Il diagramma di flusso mostra che dopo l'operazione di accensione dell'opportuno modulo si attende il tempo di acquisizione per la conversione. Infatti il modulo A/D risulta essere già impostato prima di richiamare tale subroutine, con il codice presente nella parte generale del programma dedicata alle varie periferiche.
Successivamente si è pronti per iniziare la conversione, attendere il suo termine, salvare il valore convertito e spegnere la periferica AD.
Il codice che imposta inizialmente i registri ADCON0 e ADCON1 è il seguente:

```
;Analog to Digital Converter Module
    BANKSEL ADCON1              ; Bank 1
    MOVLW   b'00000011'         ; Left justified, RA0,RA1,RA2,RA5 Analog input, +Vref=RA3
    MOVWF   ADCON1
    BANKSEL ADCON0
    MOVLW   b'00000000'         ; Fosc/2, Channel 0 (RA0), ADC disable
    MOVWF   ADCON0
```

Per quanto riguarda l'ADCON1 si impostano i vari bit in tale modo:

- *ADFM (bit 7):* viene posto a 0 in modo da ottenere una giustificazione sinistra del dato e quindi gli otto bit più significativi vengono salvati nel registro ADRESH. Tale impostazione può comunque essere cambiata in ogni momento a seconda dell'utilità non essendo infatti un'impostazione determinante al funzionamento del convertitore.
- *PCFG3:PCFG0 (bit 3,2,1,0):* volendo avere a disposizione tre canali analogici e desiderando una tensione positiva di riferimento del convertitore esterna si impostano tali bit nella configurazione "0011". In tale modalità un quarto canale analogico resta inutilizzato.

Per quanto riguarda l'ADCON0 invece:

- *ADCS1:ADCS0 (bit 7,6):* si impostano entrambi a 0 in modo da ottenere un periodo di campionamento pari a Fosc/2.
- CHS2:CHS0 (bit 5,4,3): vengono posti tutti a 0 in modo da selezionare come ingresso iniziale il canale 0. Quando è necessario effettuare il campionamento su un altro canale tali bit devono essere cambiati.
- *GO/DONE (bit 2):* inizialmente tale bit va posto a 0 ma va portato a 1 quando si desidera iniziare la conversione.
- *ADON (bit 0):* anche questo bit inizialmente va posto a zero in modo da mantenere il convertitore momentaneamente spento.

Il codice invece presente nella subroutine di conversione è il seguente:

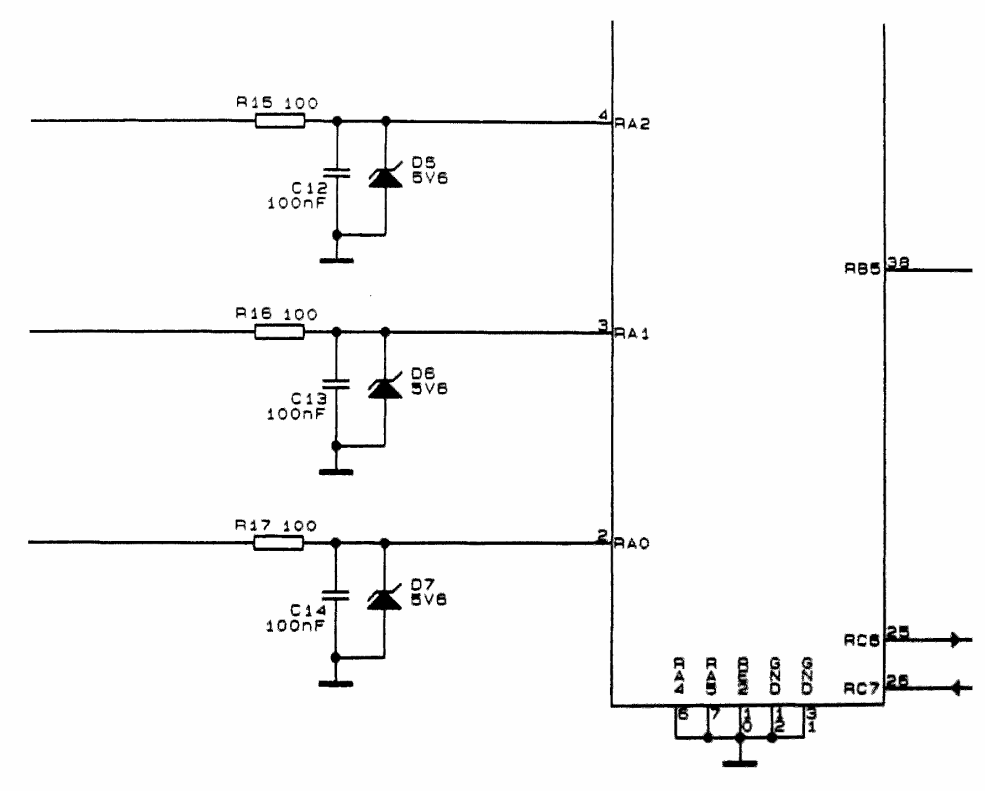
```
ADC
    BANKSEL ADCON0              ; Bank 0
    BSF     ADCON0,ADON         ; ADC enable
;Wait 20 us (at 16 MHz) for acquisition time (MAX input impedance for source = 10 KOhm)
    MOVLW   20
    MOVWF   DELAY1
    DECFSZ  DELAY1
    GOTO    $-1
```

L'accensione del convertitore viene fatta settando il bit ADON del registro ADCON0. La pausa di acquisizione approssimata a 20 µs viene fatta decrementando un registro provvisorio DELAY1 e facendo un test su di esso per verificare se è decrementato fino a 0, tutto ciò per venti volte.
Dato che le operazioni di decremento DECFSZ e GOTO impiegano due cicli di clock per essere eseguite, il tempo di esecuzione di una singola operazione, cioè 250 ns, viene moltiplicata per 20*2*2 dando origine a una pausa di 20 µs.

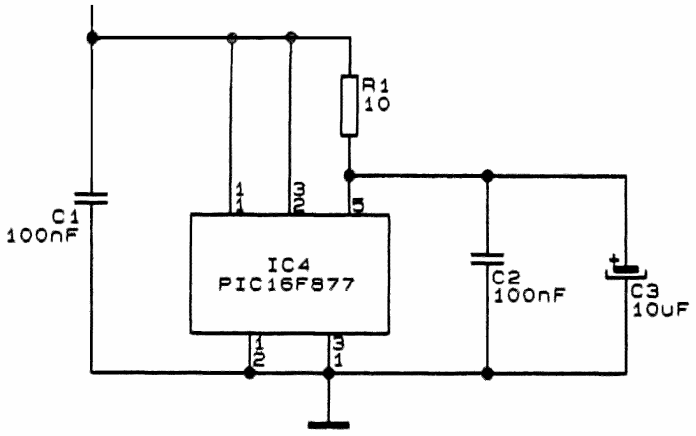
```
;Conversion
    BSF     ADCON0,GO           ; Start conversion
    BTFSC   ADCON0,GO           ; A/D over ?
    GOTO    $-1                 ; no then loop
    MOVF    ADRESH,0
    MOVWF   ADC_MSB             ; Save 2 MSB into ADC_MSB
    BANKSEL ADRESL              ; Bank 1
    MOVF    ADRESL,0
    BANKSEL ADC_LSB             ; Bank 0
    MOVWF   ADC_LSB             ; Save 8 LSB into ADC_LSB
    BCF     ADCON0,ADON         ; ADC disable
RETURN                          ; Return
```

A questo punto si setta il bit GO/DONE in modo da avviare la conversione e si fa un test continuo, sempre su questo bit, in modo da verificare quando si resetta. Infatti la periferica resetta questo bit automaticamente quando la conversione termina.
A conversione finita si provvede a salvare i risultati dei registri ADRESH e ADRESL in altri due registri di lavoro denominati ADC_MSB e ADC_LSB contenenti rispettivamente i bit più e meno significativi della conversione.
L'ultima operazione invece resetta il bit ADON in modo da spegnere il modulo convertitore e ridurre i consumi di corrente da parte del microcontrollore.

2.5.2 – Architettura hardware per il convertitore analogico - digitale



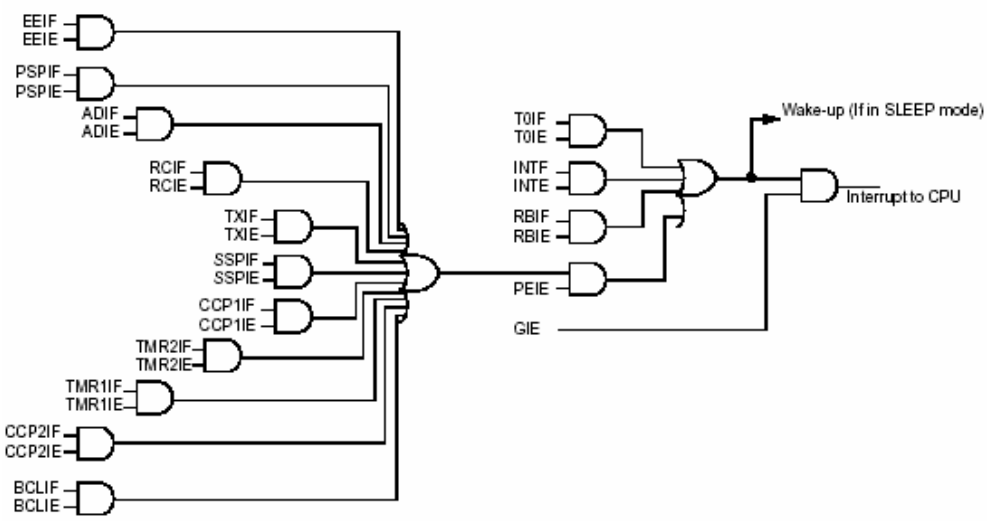
Le linee del convertitore analogico – digitale fanno capo direttamente al microcontrollore e sono a tutti gli effetti delle linee analogiche alle quali applicare le relative tensioni da convertire.
Le varie tensioni prima di essere applicate ai vari ingressi del microcontrollore vengono prima filtrate da un filtro passa basso costituito da una rete passiva R-C e poi vengono limitate a circa 5 V per evitare di danneggiare il convertitore. I diodi zener sono stati scelti con una tensione un po' superiore ai 5 V per garantire l'effettiva conversione anche a fondo scala. A causa delle tolleranze infatti, se si implementano dei diodi zener da 5,1 V, si rischia di limitare la tensione d'ingresso ad un valore minore di 5 V.



Al fine di garantire una conversione analogica – digitale accurata e precisa, si è deciso di filtrare opportunamente la sorgente della tensione di riferimento. Dalle opzioni di configurazione del convertitore infatti si è scelto che la tensione di riferimento venga prelevata esternamente dal microcontrollore e il piedino 5 ha proprio questo scopo.
La presenza della resistenza R1 in serie alla linea della tensione di riferimento e i due condensatori in parallelo C2 e C3 costituiscono proprio il suddetto filtro.

2.6 – Gestione dell'INTERRUPT

Come condizione di partenza si è supposto di avere a disposizione un unico pulsante esterno privo di rimbalzi che quando viene premuto si porta basso e abilita in maniera sequenziale le quattro coppie di transistor, e quando viene rilasciato ritorna alto e le disabilita. Risulta indispensabile quindi cambiare via software la sensibilità del fronte della linea esterna ogniqualvolta il pulsante viene premuto e rilasciato.
A tal scopo il microcontrollore dedica un particolare ingresso denominato RB0/INT a cui fa capo anche il piedino 0 della porta B. Se si sceglie di usare tale ingresso come sorgente dell'interrupt allora non è possibile usarlo anche contemporaneamente come una qualsiasi uscita o come un qualsiasi ingresso.
Il microcontrollore supporta 14 sorgenti diverse di interrupt la cui organizzazione è la seguente:



Dal momento che si decide di utilizzare almeno uno dei quattordici interrupt, il bit GIE (Global Interrupt Enable) deve essere posto a 1 per consentire l'abilitazione della porta AND la cui uscita fa capo direttamente al microcontrollore.
Nel caso di interesse, cioè di interrupt esterno, inoltre è necessario abilitare il bit INTE (Interrupt Enable) per consentire al bit INTF (Interrupt Flag) di arrivare a destinazione.
Tutti i flag delle varie periferiche in ogni caso non dipendono dai rispettivi bit di abilitazione e quindi vengono settati indipendentemente.

Per controllare l'interrupt, in particolare l'interrupt esterno, è necessario configurare opportunamente alcuni bit dei registri INTCON e OPTION_REG di cui si riportano i contenuti.

INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 GIE: Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 PEIE: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 T0IE: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 INTE: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 RBIE: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 T0IF: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 INTF: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 RBIF: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Per quanto riguarda questo registro i bit da settare sono i bit INTE (bit 4) e il bit GIE (bit 7), che consentono di abilitare l'interrupt sulla linea esterna RB0/INT.

OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 $\overline{\text{RBPU}}$: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 INTEDG: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 T0CS: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 T0SE: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 PSA: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

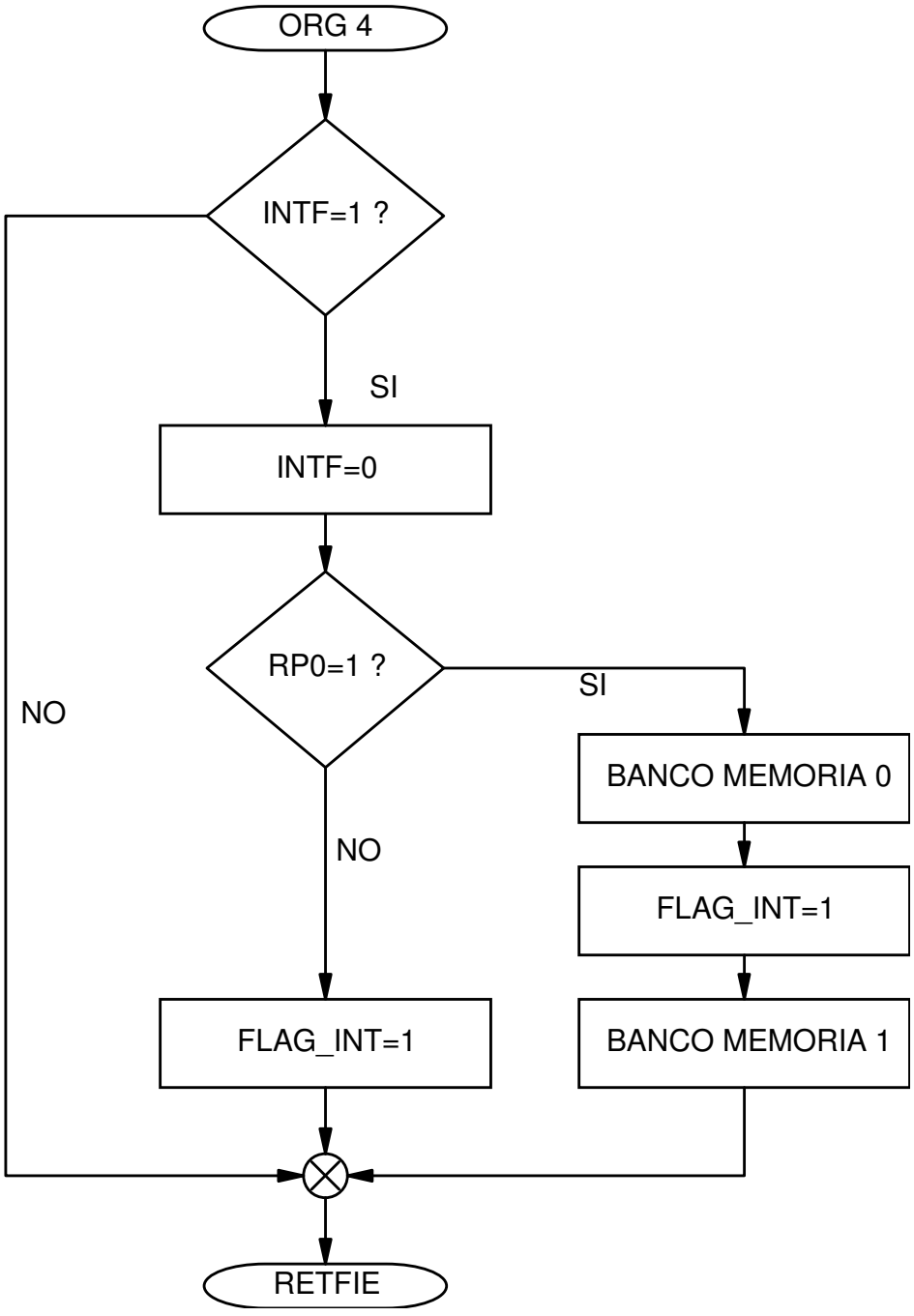
bit 2-0 PS2:PS0: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Qui il bit da considerare è il bit INTEDG (bit 6) che consente di selezionare su quale fronte del segnale d'ingresso deve intervenire l'interrupt.
Inizialmente tale bit va posto a 0 dato che la linea esterna è alta in stato di riposo.

La strategia scelta per la gestione dell'interrupt, non è quella di eseguire immediatamente tutte le operazioni di interrupt dopo che ne è stato identificato uno tramite la linea esterna. Quello che si fa con la subroutine di interrupt consiste solamente nel portare alto un bit di un opportuno registro e quindi di fissare un flag.
Ciò viene fatto per minimizzare il più possibile il tempo di esecuzione della subroutine di interrupt con lo scopo di non creare problemi di temporizzazione alle altre subroutine del programma, in particolare quelle di scambio dati quali memoria eeprom, convertitore digitale-analogico e sintetizzatore di frequenza. Quello che si cerca di evitare è eseguire una parte di codice, la cui durata risulti elevata, nel mezzo di una trasmissione dati.
Quindi in sostanza una volta che la linea esterna si porta in una condizione tale da generare un interrupt, la subroutine di gestione setta solamente un flag, impiegando pochi cicli macchina.
Sarà solo poi successivamente un'altra parte di codice presente nel programma principale, eseguita quando non è in corso alcun altra operazione, a verificare lo stato di tale flag e di generare il codice completo vero e proprio di interrupt.

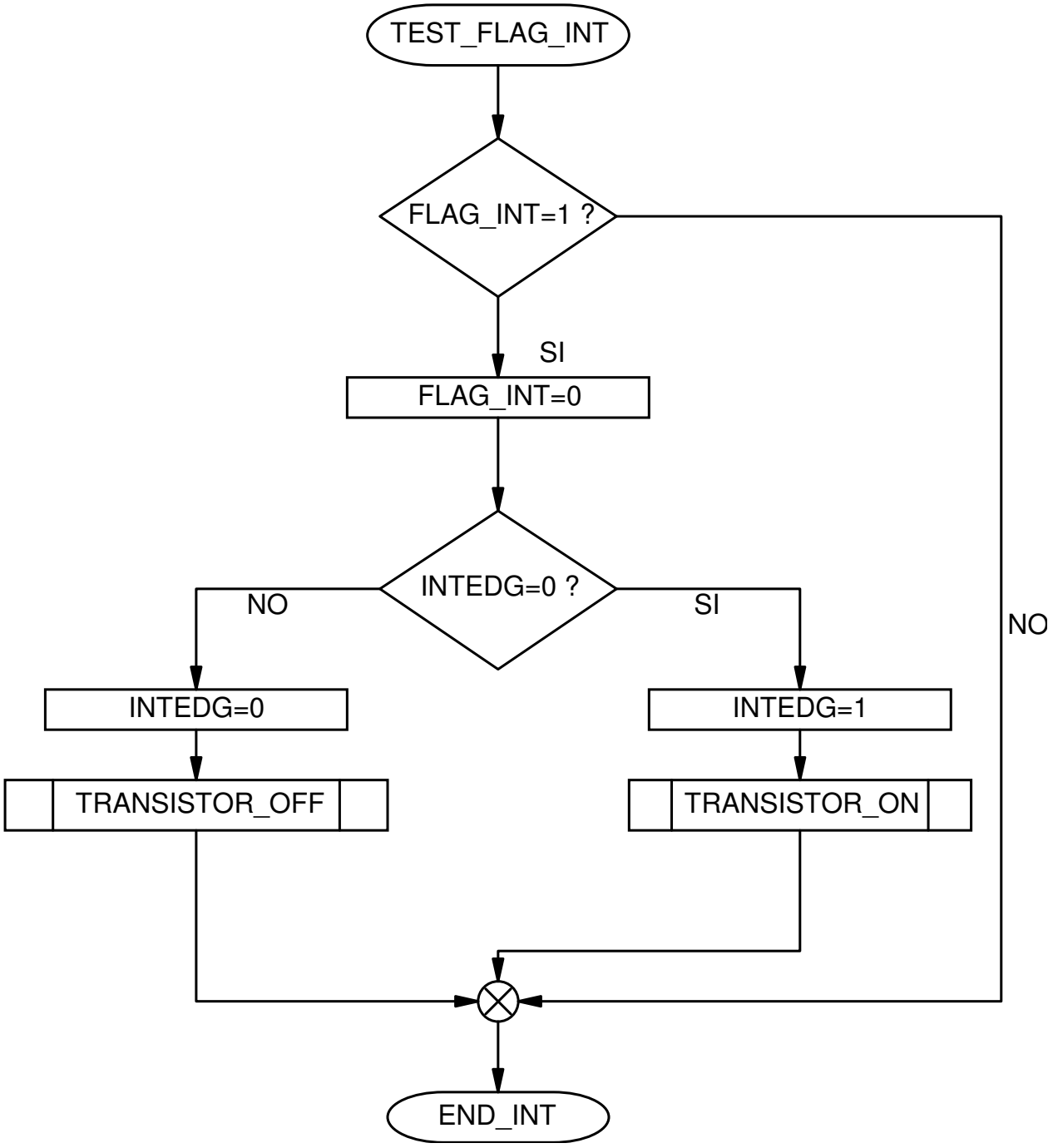
Si riporta di seguito il diagramma di flusso di gestione dell'interrupt.



Questo diagramma rappresenta quello che avviene nel codice di interrupt vero e proprio e cioè cosa avviene immediatamente dopo che la linea si porta da alta a bassa.
La prima cosa che si fa è verificare se effettivamente è avvenuto un interrupt proveniente dalla linea INT/RB0 testando lo stato del flag interrupt INTF (bit 1 del registro INTCON).
Se tale bit risulta 0 nessun interrupt è avvenuto e quindi si ritorna al punto del programma in cui si era precedentemente, se al contrario tale flag è a 1, allora è necessario eseguire la subroutine di interrupt.
Subito dopo il test è necessario resettare lo stesso bit in modo da consentire eventuali altri interrupt successivi.
A questo punto si setta il flag creato opportunamente via software denominato FLAG_INT.
La presenza di un altro test sul bit RP0 del registro STATUS si rende necessario in quanto si vuole che dopo la conclusione di tale subroutine, si riporti il banco di memoria nella stessa posizione precedente alla chiamata dell'interrupt, in modo da non creare malfunzionamenti al programma principale che è in corso.
Infatti il flag FLAG_INT è solamente il primo bit di un registro creato nella ram del microcontrollore e quindi in un opportuno banco.
Dal momento che si vuole accedere a tale registro è necessario portarsi al banco 0 ma se prima di richiamare la subroutine di interrupt ci si trovava nel banco 1, è necessario ritornarvi prima di uscire da quest'ultima.
Quindi se RP0 è 0 e ci si trova sul banco 0, si setta il flag, se RP0 è 1 ci si trova sul banco 1 e quindi bisogna prima accedere al banco 0, poi settare il flag e poi ritornare al banco 1.
Avendo usato in questo programma solamente registri compresi tra il banco 0 e 1, si verificano solo questi due casi.
Il codice che esegue la subroutine di interrupt è il seguente.

```
    ORG     0x0004
    BTFSS   INTCON,INTF         ; Interrupt test from RB0/INT pin
RETFIE                          ; Return from Interrupt vector
    BCF     INTCON,INTF         ; Turn off INTF bit
    BTFSS   STATUS,RP0          ; Memory bank test
    GOTO    INT1                ; Goto subroutine bank0
    GOTO    INT2                ; Goto subroutine bank1

INT1
    BSF     FLAG_INT,0          ; Turn on Interrupt Flag
RETFIE                          ; Return from Interrupt vector

INT2
    BANKSEL FLAG_INT
    BSF     FLAG_INT,0          ; Turn on Interrupt Flag
    BSF     STATUS,RP0          ; Return on bank 1
RETFIE                          ; Return from Interrupt vector
```

Il diagramma di flusso che gestisce il flag FLAG_INT presente nel programma principale è invece il seguente:



Questa subroutine fa un test sul flag FLAG_INT eventualmente settato dalla subroutine di interrupt.
In caso negativo esce dalla subroutine, in caso positivo si passa al test per verificare se i transistor vanno abilitati o disabilitati. Per fare ciò si parte dalla condizione che inizialmente il fronte del segnale di interrupt sia basso e che la linea si porti da alta a bassa. Il test consiste nel verificare se il bit INTEDG (bit 6 del registro OPTION_REG) cioè il bit che seleziona il fronte è 1 o 0. Quindi se è 0 bisogna abilitare i transistor e cambiare il fronte di sensibilità da basso ad alto per permettere lo spegnimento successivamente al momento del rilascio del pulsante.
Al momento del rilascio del pulsante, visto il cambio di sensibilità del fronte, verrà generato un altro interrupt e nuovamente poi verrà settato il FLAG_INT ed eseguita tale subroutine percorrendo però l'altra via, cioè quella che disabilita i transistor.

Il codice che esegue questo diagramma di flusso è il seguente:

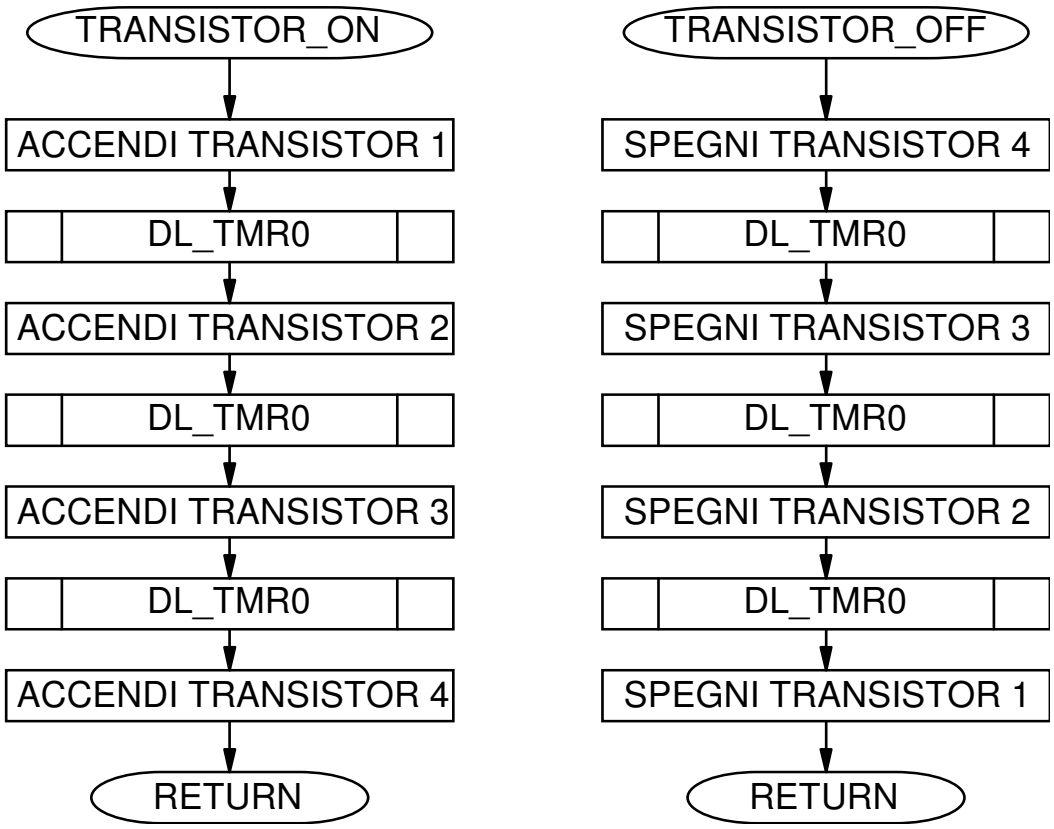
```
TEST_INT_FLAG                   ; Interrupt Flag Test

    BANKSEL  FLAG_INT           ; Bank 0
    BTFSS    FLAG_INT,0         ; Interrupt Flag Test
    GOTO     END_INT            ; Go out
    BCF      FLAG_INT,0         ; Turn off Interrupt Flag
    BANKSEL  OPTION_REG         ; Bank1
    BTFSC    OPTION_REG,INTEDG  ; Test edge of Interrupt
    GOTO     TR_OFF             ; Rising edge
    GOTO     TR_ON              ; Falling edge

TR_OFF
    BCF      OPTION_REG,INTEDG  ; Change Interrupt edge
    CALL     TRANSISTOR_OFF     ; Turn off transistors sequence
    GOTO     END_INT

TR_ON
    BSF      OPTION_REG,INTEDG  ; Change Interrupt edge
    CALL     TRANSISTOR_ON      ; Turn on transistors sequence

END_INT
```

2.6.1 – Architettura hardware per l'ingresso interrupt

La linea di interrupt del microcontrollore è tenuta in condizione di riposo a livello logico alto mediante una resistenza di pull up da 10 KΩ. Tale linea è anche collegata alla massa mediante un condensatore da 1 nF per costituire un basilare circuito antirimbalzo per il dispositivo di comando esterno.

2.7 – Accensione e spegnimento dei blocchi di media – bassa potenza

Di seguito si riportano i diagrammi di flusso del codice che esegue l'accensione e lo spegnimento delle quattro coppie di transistor. Il codice non fa altro che abilitare, nel caso dell'accensione, o disabilitare, nel caso dello spegnimento, le quattro linee del microcontrollore in maniera sequenziale, intervallate da un certo ritardo definibile via software.

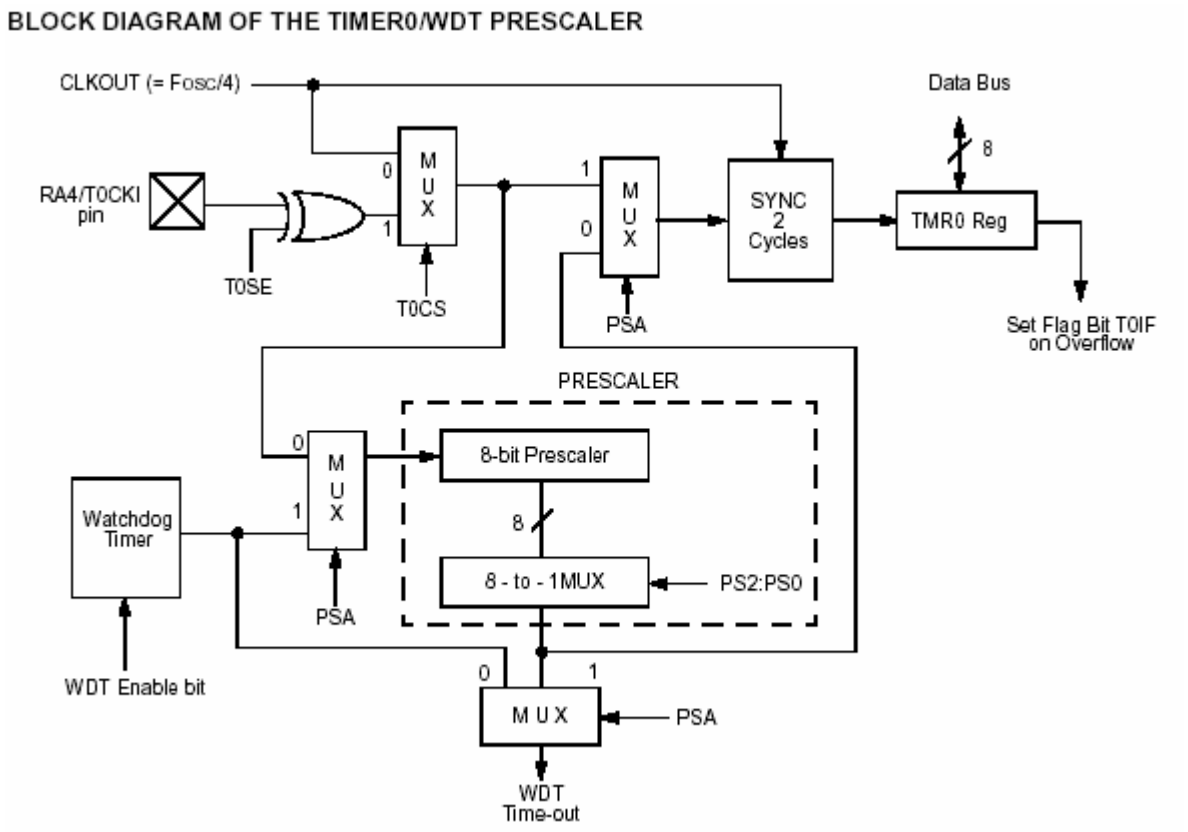


Le righe di codice che rappresentano i diagrammi di flusso sono le seguenti:

```
TRANSISTOR_ON
    BANKSEL PORTB               ; Bank 0
    BSF     PORTB,1             ; RX/TX_1 ON
    CALL    DL_TMR0             ; Delay
    BSF     PORTB,2             ; RX/TX_2 ON
    CALL    DL_TMR0             ; Delay
    BSF     PORTB,4             ; RX/TX_3 ON
    CALL    DL_TMR0             ; Delay
    BSF     PORTB,5             ; RX/TX_4 ON
RETURN                          ; Return
```

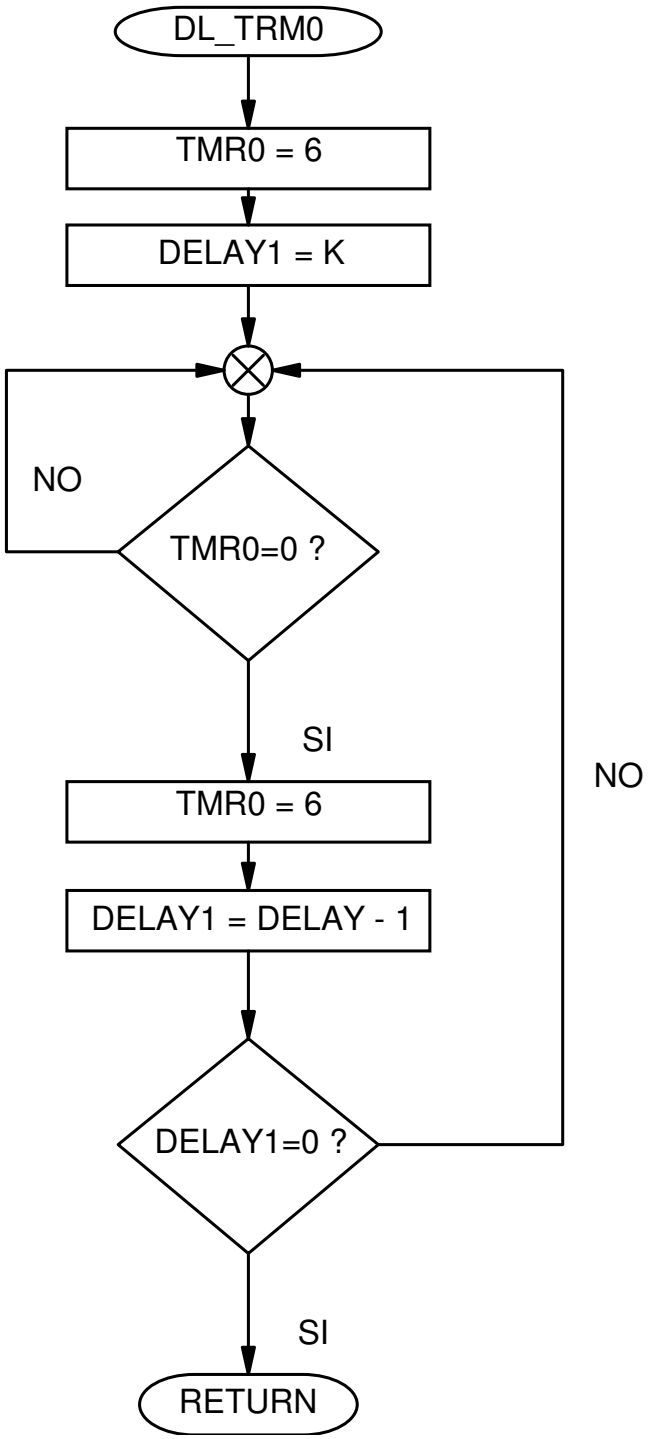
```
TRANSISTOR_OFF
    BANKSEL PORTB               ; Bank 0
    BCF     PORTB,5             ; RX/TX_4 OFF
    CALL    DL_TMR0             ; Delay
    BCF     PORTB,4             ; RX/TX_3 OFF
    CALL    DL_TMR0             ; Delay
    BCF     PORTB,2             ; RX/TX_2 OFF
    CALL    DL_TMR0             ; Delay
    BCF     PORTB,1             ; RX/TX_1 OFF
RETURN                          ; Return
```

Il ritardo viene gestito dalla subroutine DL_TMR0 che provvede, in base ad una costante modificabile via software, di creare una precisa pausa in modo da temporizzare opportunamente le sequenze di accensione e spegnimento.
Per creare una temporizzazione di precisione si è sfruttato il TIMER0 del microcontrollore, controllato dal clock interno del microcontrollore scandito dall'oscillatore a quarzo.



Il TIMER0 è un registro a otto bit leggibile e scrivibile in qualsiasi momento, che può incrementare il suo conteggio in base ad un clock esterno o interno al microcontrollore.
È presente inoltre un prescaler a 8 bit, condiviso tra l'altro con il blocco watchdog, che permette di dividere il clock d'ingresso di un certo fattore.

La subroutine di ritardo esegue le seguenti operazioni:



La subroutine non fa altro che dividere il clock principale di un certo valore.
Partendo dalla frequenza dell'oscillatore di 16 MHz essa viene già divisa sempre per 4 dal microcontrollore stesso in modo da fornire una frequenza di macchina di 4 MHz. Impostando il prescaler del timer con un fattore di divisione 1:256 la frequenza presente all'ingresso del timer è di 15,625 KHz.
Caricando inizialmente il timer con un valore pari a 6, il microcontrollore impiega 256 – 6 =250 cicli macchina prima di tornare al valore 0 e prima di superare il primo blocco di test. Quindi, quello che è stato fatto, è di aver diviso il clock ancora di un fattore 250 ottenendo una frequenza di 62,6 Hz. Se il registro DELAY1 risulta ad esempio inizialmente precaricato con un valore 62 quello che si ottiene tramite il secondo blocco di test è un ritardo totale pari a 1,008 s. Modificando il valore del registro DELAY1, tramite la costante K, si possono ottenere altre temporizzazioni.

Quindi la relazione che lega la costante K al tempo di ritardo è la seguente:

$$T\_di\_ritardo = \frac{K}{62{,}5}$$

I vari parametri di gestione del timer sono selezionabili configurando opportunamente il registro OPTION_REG di cui si riporta di seguito il contenuto.

OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 $\overline{\text{RBPU}}$: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 INTEDG: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 T0CS: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 T0SE: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 PSA: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 PS2:PS0: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

I bit interessati alla gestione del TIMER0 sono:

- *T0CS (bit 5):* questo bit viene posto a 0 in modo da garantire un incremento del timer con il clock interno di macchina del microcontrollore.
- *T0SE (bit 4):* avendo scelto un clock interno, la configurazione di tale bit è indifferente e per default viene posto a 0.
- *PSA (bit 3):* con questo bit si seleziona se assegnare il prescaler al timer oppure al watchdog. Volendo usare in tale applicazione il prescaler tale bit va a 0.
- *PS2:PS0 (bit 2,1,0):* in base alla tabella con questi tre bit si seleziona il valore da assegnare al prescaler. Volendo ottenere un fattore di divisione pari a 1:256 tutti i tre bit vengono posti a 1.

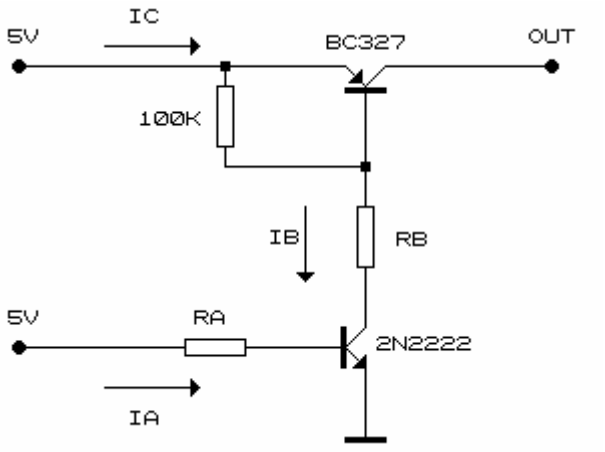
Questo registro viene configurato nella parte di programma dedicata alla configurazione generale delle varie impostazioni, mentre il codice programma che esegue la subroutine di ritardo è il seguente.

```
DL_TMR0
    MOVLW   6
    MOVWF   TMR0                ; Move 6 into TMR0
    MOVLW   K
    MOVWF   DELAY1              ; Move K into DELAY1
DL
    MOVF    TMR0,0              ; Move TMR0 into W
    BTFSS   STATUS,Z            ; Test ZERO bit
    GOTO    DL
    MOVLW   6
    MOVWF   TMR0                ; Move 6 into TMR0
    DECFSZ  DELAY1,1            ; Decrement DELAY1, skip if 0
    GOTO    DL
RETURN
```

Il primo test viene fatto andando a verificare il valore assunto dal bit Z (Zero bit) del registro STATUS. Questo bit infatti si porta alto ogniqualvolta il risultato di un'operazione aritmetica o logica è zero e può quindi essere pienamente sfruttato in questo caso.
Il secondo test invece sfrutta il comando assembler "DECFSZ", che oltre a decrementare il registro DELAY1, subito dopo controlla se il valore raggiunto e zero.

2.7.1 – Architettura hardware per i blocchi di media - bassa potenza

Per ogni blocco di potenza è stato previsto una coppia di transistor di tipo NPN e PNP. La base del transistor NPN viene comandata dalla linea del microcontrollore, mentre la base del transistor PNP viene comandata dal transistor NPN come da schema. Entrambi i transistor funzionano come interruttori elettronici e lavorano quindi di conseguenza in zona di interdizione o in zona di saturazione.
Il collegamento diretto con il dispositivo esterno è effettuato tramite il transistor PNP, che garantisce una bassa caduta di tensione tra collettore ed emettitore. Una resistenza da 100 KΩ, tra emettitore e base, è presente per consentire una maggior stabilità termica.
Il transistor NPN invece, è introdotto per non sollecitare troppo le linee del microcontrollore. Infatti, senza la presenza di quest'ultimo, ogni linea dovrebbe poter fornire la corrente di base IB che è di valore troppo elevato.
Di seguito vengono riportati i calcoli effettuati per il dimensionamento delle resistenze di base dei transistor, per una corrente di assorbimento massima dei dispositivi esterni di 0,5 A.



E' noto il β dei due transistor e la corrente massima che scorre nell'emettitore del transistor PNP:

$$\beta_{2N2222} = \beta_{BC327} = 50\,,\ IC = 0{,}5A$$

Quindi dalla formula seguente ricavo la corrente di base IB:

$$IB = \frac{IC}{\beta} = \frac{0{,}5}{50} = 10mA$$

Scrivendo l'equazione della maglia d'ingresso del transistor PNP

$$Vcc = RB * IB + 0{,}7$$

trovo il valore della resistenza di base RB:

$$RB = \frac{5 - 0{,}7}{IB} = \frac{4{,}3}{10 * 10^{-3}} = 430\Omega$$

Nota la corrente di base IB è possibile ricavare la corrente IA per mezzo della formula:

$$IA = \frac{IB}{\beta} = \frac{10 * 10^{-3}}{50} = 0{,}2mA$$

Quindi scrivendo la maglia d'ingresso per il transistor NPN:

$$Vcc = RA * IA + 0{,}7$$

trovo RA:

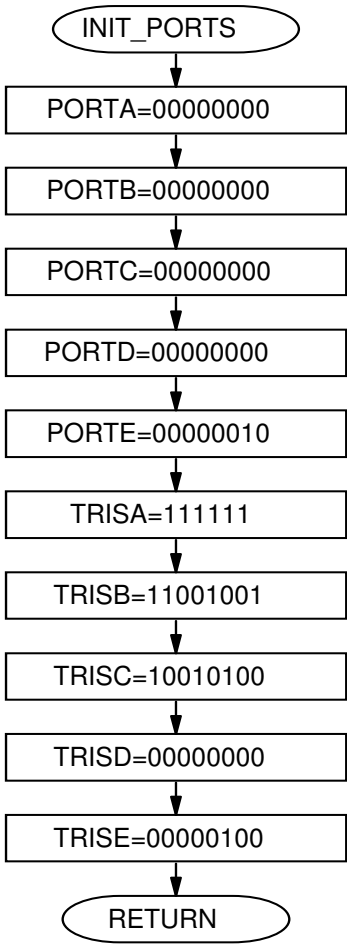
$$RA = \frac{5 - 0{,}7}{0{,}2 * 10^{-3}} = 21{,}5K\Omega$$

Per garantire una più marcata saturazione di entrambi i transistor si sono scelti i seguenti valori di resistenza: $RA = 10K\Omega$ e $RB = 220\Omega$.

In parallelo ad ogni uscita è presente un led con la sua rispettiva resistenza di limitazione della corrente, al fine di monitorare l'effettivo stato dei blocchi di media - bassa potenza.

2.8 – Inizializzazione delle porte

In base alle funzioni che devono svolgere i vari piedini del microcontrollore, essi devono essere configurati come ingressi o come uscite. Di seguito si riporta il diagramma di flusso dell'inizializzazione delle porte del microcontrollore in modo da poter identificare quali piedini vengono configurati come ingressi e quali come uscite.



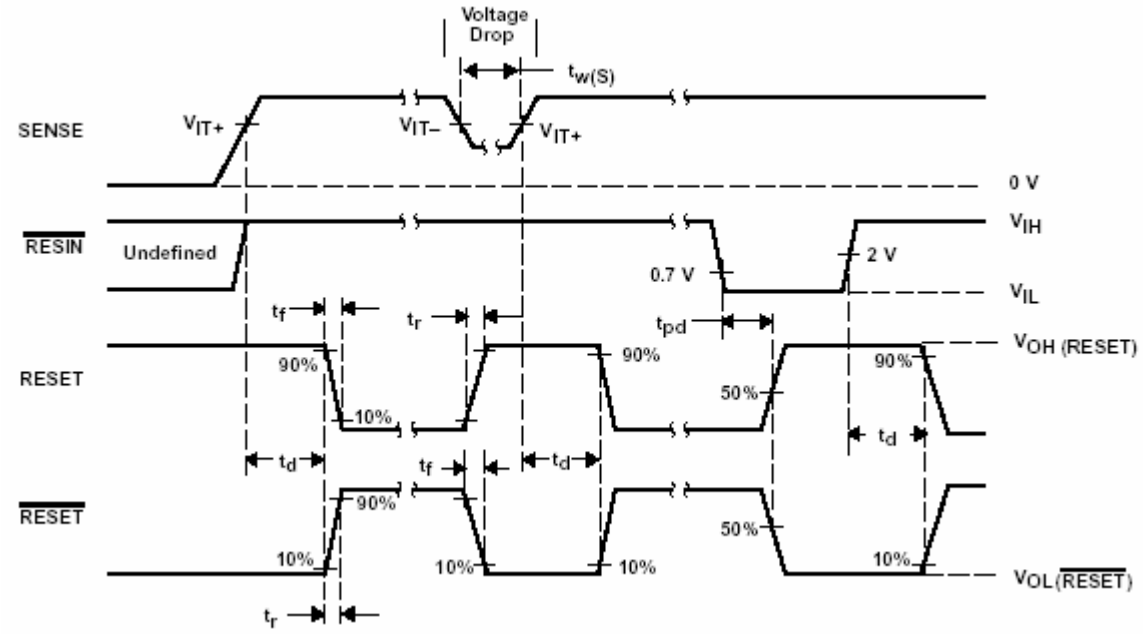
Immediatamente dopo l'alimentazione tutte le linee vengono settate su una configurazione standard di tipo spento in modo da non attivare alcun dispositivo esterno.
Subito dopo si provvede a scegliere le linee come ingressi o come uscite. La porta A essendo analogica, deve in ogni caso essere impostata come ingresso, dato che tutte le linee acquisiscono dei segnali, in particolare segnali analogici. Sulla porta B le linee che pilotano le basi dei transistor sono per tale motivo poste come uscite, mentre la linea di interrupt dovendo acquisire il segnale di trasmissione è scelta come ingresso.
Sulla porta C vengono settati come ingressi la linea MUXOUT del circuito sintetizzatore, che segnala l'eventuale avvenuto aggancio in fase da parte del PLL, la linea dei dati in ingresso del circuito RS232 e la linea dei dati del bus IIC, anche se quest'ultima in ogni caso viene controllata interamente dalla periferica MSSP.
Per quanto riguarda la porta D, essa è impostata tutta come uscita, in quanto a lei fa capo la linea del segnale MUTE che è un'uscita verso il trasmettitore e altre sette linee libere scelte anch'esse come uscite.
La porta E invece, viene configurata come uscita per quanto riguarda le due linee che consentono l'abilitazione della memoria e del convertitore DAC.
Il terzo piedino della porta E viene posto come ingresso dato che è collegata a massa e pertanto si vuole evitare di portare tale piedino erroneamente a livello logico 1 mandando in corto circuito il microcontrollore. Per lo stesso motivo sono stati posti come ingressi anche i piedini RA4 e RA5 della porta A.
È importante quindi per non danneggiare la scheda, non configurare questi tre piedini come uscite e portarli successivamente a livello logico alto.

2.9 – Sistema di reset

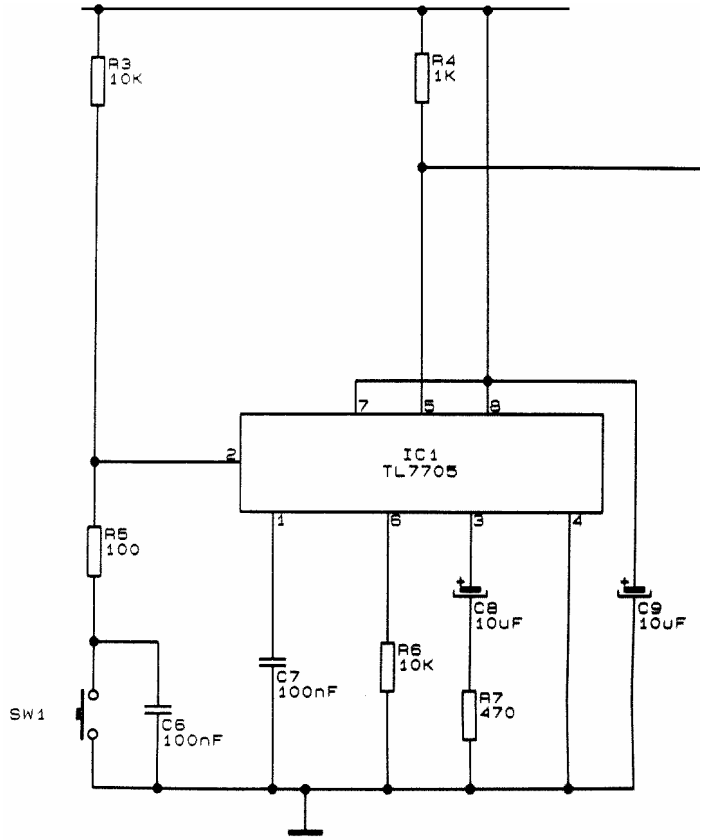
Questo sistema viene utilizzato per mantenere in stato di reset il microcontrollore al momento dell'applicazione della tensione di alimentazione dell'intero circuito fino al raggiungimento di un'opportuna soglia. Questo viene fatto in modo da evitare malfunzionamenti del circuito nei primi istanti dall'accensione per l'eventuale stato non corretto delle porte del microcontrollore.
Sull'integrato di questo sistema è collegato un pulsante per poter effettuare in qualsiasi momento un reset manuale del microcontrollore da parte dell'utente. Un circuito antirimbalzo interno all'integrato provvede a mandare un solo impulso di reset all'apposito piedino del microcontrollore quando il pulsante viene premuto.
Di seguito si riporta un'analisi dettagliata del funzionamento dell'integrato.
Il livello della tensione di alimentazione viene controllato attraverso il piedino di $SENSE$.

Dal momento che si fornisce tensione al circuito, il piedino di reset $\overline{RESET}$ si porta attivo, cioè a livello logico basso solo dopo il raggiungimento di una tensione di alimentazione pari a 3,6 V. Prima infatti il circuito si trova in una condizione non definita.
Mentre la tensione continua a salire, viene raggiunta la tensione di soglia V_{IT} e da questo istante inizia un ritardo pari a T_D dopo il quale il piedino $\overline{RESET}$ si porta alto sbloccando il microcontrollore dallo stato di reset.
Nel caso in cui successivamente la tensione scendesse al di sotto della soglia V_{IT}, anche per un breve istante, il piedino $\overline{RESET}$ si porta basso resettando il microcontrollore. Anche in questo caso dal momento che la soglia viene nuovamente raggiunta deve trascorrere un tempo T_D affinché si sblocchi la situazione di reset.

Premendo il pulsante esterno di reset si porta basso il piedino $\overline{RESIN}$, il quale comanda il circuito interno all'integrato, che a sua volta porta basso il piedino $\overline{RESET}$ consentendo il reset del microcontrollore.
I rimbalzi del pulsante non influiscono sul reset in quanto dopo il primo impulso interviene il ritardo T_D che mantiene in ogni caso comunque la linea $\overline{RESET}$ bassa.
Il diagramma seguente provvede a facilitare la comprensione del funzionamento appena descritto.



Il tempo di ritardo T_D è determinato dal valore della capacità esterna collegata al piedino CT per mezzo della formula: $T_D = 1{,}3 \cdot 10^4 \cdot C_T$. In questo caso che la capacità vale 10 µF il tempo di ritardo vale 13 ms.



Lo schema usato è quello fornito dal datasheet dell'integrato con la modifica per l'applicazione del pulsante di reset. La tensione del piedino d'ingresso per il segnale di reset viene portata sotto la soglia mediante un partitore di tensione attivato dalla chiusura del pulsante stesso.
In parallelo al pulsante è posto un condensatore da 100 nF per limitare anche via hardware i rimbalzi.

3 - Comunicazione con il Personal Computer tramite la porta RS232

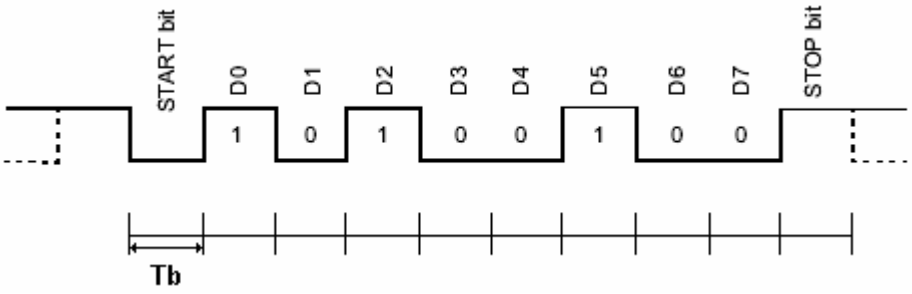
Tale comunicazione è stata implementata per consentire all'utente esterno di poter accedere ai dati del microcontrollore tramite una facile interfaccia quale appunto può essere quella di un PC.
Questa interfaccia risulta molto utile in fase di debug dell'intera scheda per controllare l'effettivo funzionamento delle varie periferiche e dei vari componenti che comunicano con il microcontrollore stesso. Infatti è possibile trasmettere a quest'ultimo, tramite un banale software, dei byte di dati e memorizzarli in alcuni registri utilizzabili successivamente in qualsiasi modo, oppure al contrario è possibile ricevere dei byte di dati memorizzati in alcuni registri e visualizzarne il contenuto sul PC.
In tal modo è possibile ad esempio scrivere dei byte di dati nella memoria EEPROM 24C08 digitando il loro valore per mezzo della tastiera del PC. Poi è possibile prelevare gli stessi dati, visualizzarli sullo schermo del PC e verificarne l'effettiva uguaglianza constatando l'effettivo funzionamento della memoria.

3.1 – Protocollo seriale asincrono RS232

Il collegamento seriale RS232 prevede un protocollo di comunicazione asincrono. Il fatto che sia asincrono significa sostanzialmente che, al contrario di quanto avviene per esempio nella trasmissione IIC o *SPI,* non è presente alcuna linea esterna di clock che sincronizza il cambio e il campionamento dei dati.
Il collegamento RS232 è bidirezionale e presenta due linee di dati, una in trasmissione e l'altra in ricezione, più il segnale di massa.
Nella comunicazione seriale asincrona, in generale, il numero di bit che vengono trasmessi alla volta si aggira intorno ai 7-12 bit e la durata del tempo di ogni singolo bit è costante.
L'informazione utile contenuta è composta da 5-8 bit mentre gli altri bit costituiscono segnali di gestione della comunicazione. In particolare i bit di gestione sono il bit di Start, il bit di Stop e l'eventuale bit di controllo di parità.
Una trama completa comprende quindi in ordine:

- un bit di Start che segnala l'inizio di una trasmissione portando la linea da livello logico 1 a livello logico 0;
- l'informazione utile, costituita da una sequenza di 5-8 bit;
- un bit di parità che tramite il suo valore permette la verifica della corretta trasmissione del segnale utile;
- uno o più bit di Stop che segnalano la fine della trasmissione ripristinando il livello logico da 0 a 1.

La comunicazione seriale RS232 utilizzata dal PC, o comunque sfruttata in questo caso, è costituita da un bit di Start, 8 bit di dati e un bit di Stop per un totale di 10 bit.
Di seguito si riporta una forma d'onda tipica di una trasmissione asincrona.



Si noti lo stato di riposo della linea a livello logico 1 e si noti la durata costante di ogni singolo bit di durata Tb, dove il suo inverso indica la velocità di trasmissione o più precisamente il baud rate:

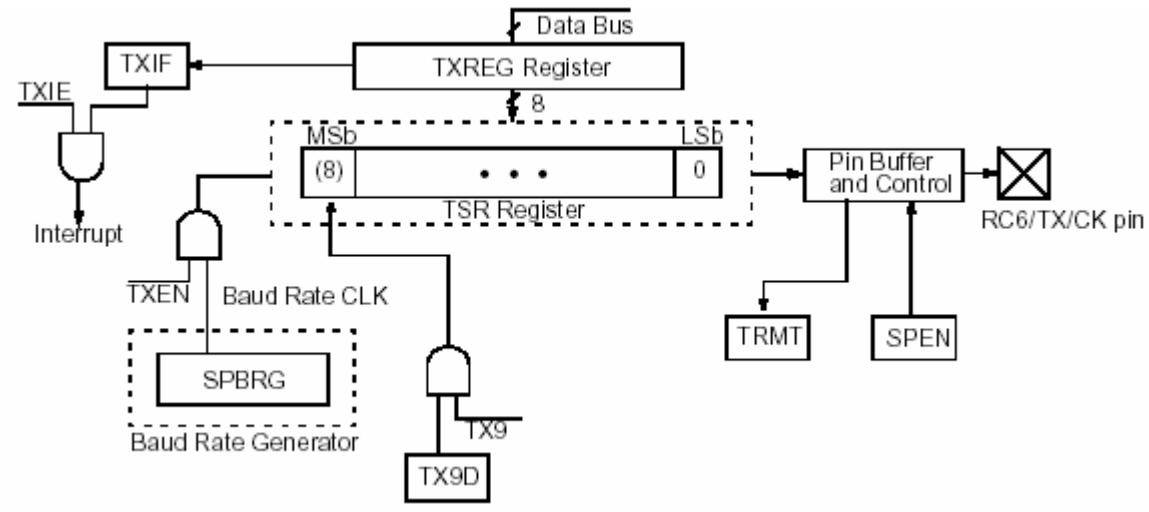
$$baud\_rate = \frac{1}{Tb} bit/s$$

Valori tipici di baud rate per questo tipo di trasmissioni sono 110, 300, 600, 1200, 2400, 4800, 9600 e 19600 baud.

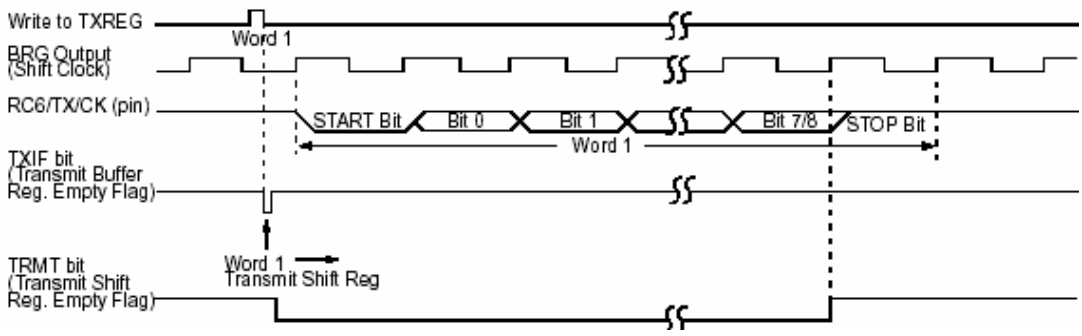
Nonostante la comunicazione sia asincrona, sia il trasmettitore che il ricevitore debbono essere provvisti di clock interno avente *periodo* uguale al tempo di ogni singolo bit Tb. Il campionamento dello stato della linea in ricezione viene fatto in corrispondenza di frazioni di periodo Tb (di solito 1/16, 1/32 o 1/64). Supponendo ad esempio che il clock di campionamento sia 32 volte maggiore di quella del bit, si verificherà che a partire dall'istante di inizio trasmissione il bit di Start verrà campionato dopo 16 colpi di clock. Ora se lo stato della linea viene ancora a trovarsi a livello logico 0 allora verrà riconosciuto l'effettivo bit di Start.

3.2 – Modulo hardware USART

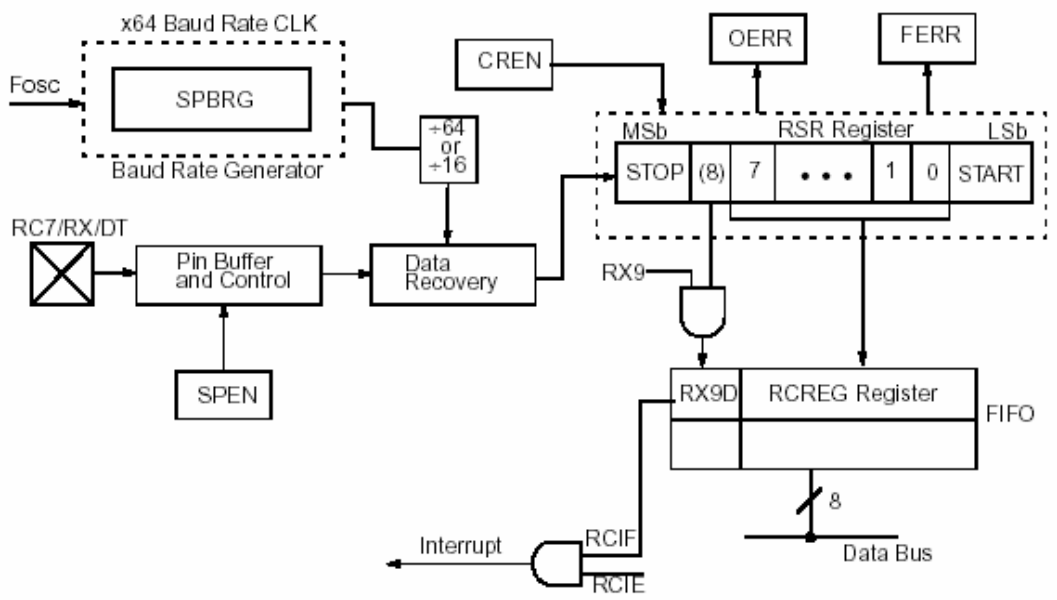
Il microcontrollore in dotazione è provvisto della periferica hardware USART (Universal Synchronous Asynchronous Receiver Trasmitter) il cui comune impiego consiste proprio in comunicazioni tramite il protocollo RS232.
Come si può constatare dal nome della periferica, è possibile anche l'impiego in trasmissioni sincrone che comunque non sono di interesse in questa applicazione.
La periferica nell'impiego asincrono usa il formato non-return-to-zero (NRZ) che consiste nell'invio di un bit di start, otto o nove bit di dati e un bit di stop.
Un generatore di clock a otto bit integrato nella periferica, denominato Baud Rate Generator BRG, può essere usato per prelevare standard baud rate derivati dal clock principale del microcontrollore stesso. Tale generatore di baud rate produce un clock con frequenza 16 o 64 volte maggiore rispetto alla frequenza di shift del singolo bit. Il trasmettitore e il ricevitore sono funzionalmente indipendenti ma sono legati l'un l'altro dallo stesso formato di dati impostato e dallo stesso baud rate; inoltre ricevono i dati con il bit meno significativo per primo.
Il controllo della parità non è eseguito in automatico dalla periferica USART, ma se lo si vuole implementare, è necessario farlo separatamente via software. A tal scopo la periferica salva in un opportuno registro il valore del nono bit trasmesso per poter essere processato successivamente.
Di seguito si analizza il funzionamento del trasmettitore della periferica USART e tal scopo si riporta uno schema a blocchi.



Il registro TXREG è il buffer dove viene caricato via software il dato da trasmettere. Dal momento che viene rilevato il bit di stop della precedente trasmissione, allora il contenuto di questo registro viene copiato nel registro Transmit Shift Register TSR che viene utilizzato proprio per trasmettere sulla linea di uscita ogni singolo bit.
Una volta che il registro TXREG scarica i dati nel TSR, il TXREG è vuoto e il flag di interrupt TXIF (bit 4 del registro PIR1) viene settato. Un altro bit denominato TRMT (bit 1 del registo TXSTA), quando è a livello logico 1, indica che il registro TSR è vuoto e quindi una trasmissione è finita.

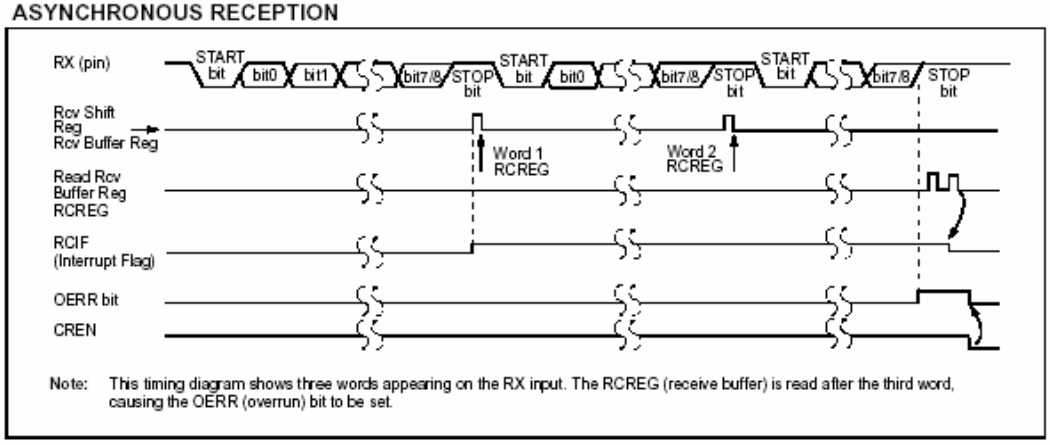


Si analizza a questo punto il ricevitore della periferica USART.



Il dato viene ricevuto dal piedino RC7 il quale pilota il blocco di recupero dati Data Recovery. Questo blocco è un shift register ad alta velocità che opera con un baud rate 16 volte superiore a quello impostato nel BRG. Dopo il campionamento del segnale di stop, il dato ricevuto presente nel registro Receive Shift Register RSR, viene trasferito al registro RCREG se è vuoto. Se il trasferimento è completo, il flag RCIF (bit 5 del registro PIR1) viene settato.
Il registro RCREG è un doppio buffer nel quale è anche possibile salvare due byte di dati e ricevere allo stesso momento un terzo byte sul RSR.
Si riporta una figura di una ricezione di un byte in cui si vuole evidenziare lo stato dei bit di interrupt RCIF.

ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

3.3 - Gestione della periferica USART e analisi della subroutine di controllo

La comunicazione tra la periferica USART del microcontrollore e il PC è stata prevista senza nessun controllo di parità sui vari byte scambiati. Il protocollo creato prevede quindi un bit di start, un byte di dati e un bit di stop.
L'impostazione della periferica USART viene fatta caricando opportunamente i tre registri TXSTA, RCSTA e SPBRG i quali contengono rispettivamente lo stato e il controllo del trasmettitore, lo stato e il controllo del ricevitore e la velocità di comunicazione tramite il Baud Rate Generator.
Si riportano di seguito le figure che mostrano i contenuti degli appena citati registri:

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be parity bit

RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN**: Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Don't care

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN**: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D**: 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

La configurazione della periferica USART viene fatta nella parte del programma apposita dedicata all'impostazione delle varie periferiche.
Il caricamento dei tre registri avviene per mezzo delle seguenti righe di codice:

```
;USART Asynchronous Mode RS_232
   BANKSEL SPBRG                ; Bank 1
   MOVLW   BAUD_USART           ; 9600 baud
   MOVWF   SPBRG
   MOVLW   b'00000000'          ; 8 bit trasmission, Asynchronous mode, Low speed
   MOVWF   TXSTA
   BANKSEL RCSTA
   MOVLW   b'00000000'          ; Serial port disable, 8 bit reception, Address detection disable
   MOVWF   RCSTA
```

Il registro TXSTA viene impostato nel seguente modo:

- *TX9 (bit 6):* questo bit imposta la trasmissione a 9 bit o a 8 bit. Portandolo a zero si imposta la modalità a 8 bit come deciso precedentemente non implementando nessun controllo di parità.
- *TXEN (bit 5):* se viene settato, serve ad iniziare la trasmissione. Momentaneamente questo bit deve rimanere a 0.
- *SYNC (bit 4):* tramite questo bit si seleziona il funzionamento della periferica USART in modalità sincrona o asincrona. Quindi volendo una modalità asincrona tale bit va posto a 0.
- *BRGH (bit 2):* è possibile far lavorare la periferica impostandola su alte o basse velocità di trasmissione. Non essendoci particolari esigenze di alte velocità in questo tipo di comunicazione si pone tale bit a 0 scegliendo una velocità di lavoro bassa.

Il registro RCSTA invece, viene impostato nel seguente modo:

- *SPEN (bit 7):* momentaneamente, questo bit va posto a zero disabilitando la periferica. Verrà abilitato successivamente nelle due subroutine di comunicazione.
- *RX9 (bit 6):* resettando questo bit si provvede a selezionare il ricevitore sulla modalità a 8 bit.
- *CREN (bit 4):* avendo progettato la comunicazione con un trasferimento di un singolo byte alla volta, questo bit va posto a 0 disabilitando ricezione continue di dati.
- *ADDEN (bit 3):* anche questo bit si pone a 0 in quanto la trasmissione come già detto avviene con 8 bit.

Il registro SPBRG in base al valore con cui è caricato, determina la velocità di trasmissione. Il valore da caricare dipende da come si è configurato il bit BRGH del registro TXSTA: infatti a seconda della modalità di funzionamento "veloce" o "lenta" per la stessa velocità è necessario caricare il registro SPBRG con due valori differenti.
Avendo imposto il bit BRGH a 0 la formula che lega il Baud Rate al valore del SPBRG è la seguente:

$$Baud\_Rate = \frac{Fosc}{16*(SPBRG+1)}$$

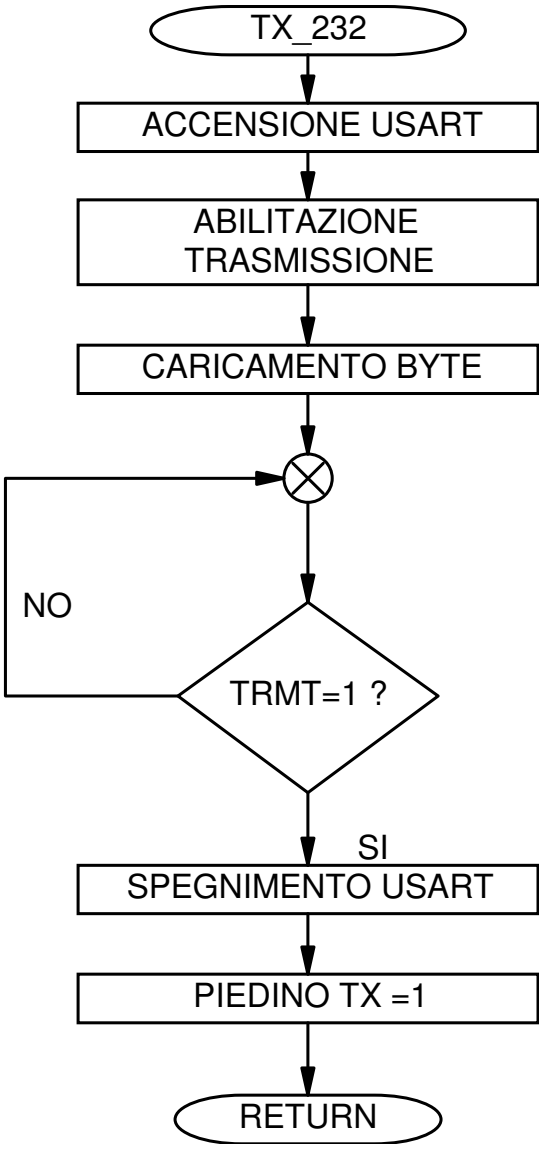
In ogni caso, dal datasheet del microcontrollore sono disponibili delle tabelle che illustrano i vari valori con cui caricare il registro SPBRG in funzione della velocità che si desidera ottenere.

BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	-	-	-	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	207	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	103	2.404	0.17	64
9.6	9.766	1.73	31	9.615	0.16	25	9.766	1.73	15
19.2	19.531	1.72	15	19.231	0.16	12	19.531	1.72	7
28.8	31.250	8.51	9	27.778	3.55	8	31.250	8.51	4
33.6	34.722	3.34	8	35.714	6.29	6	31.250	6.99	4
57.6	62.500	8.51	4	62.500	8.51	3	52.083	9.58	2
HIGH	1.221	-	255	0.977	-	255	0.610	-	255
LOW	312.500	-	0	250.000	-	0	156.250	-	0

La tabella mostra vari valori a seconda anche della frequenza dell'oscillatore a quarzo del microcontrollore.
La velocità di funzionamento scelta per la periferica USART è di 9600 Baud che assicura un buon compromesso tra prestazioni ed errori di trasmissione. Dato che il microcontrollore lavora con un quarzo da 16 MHz, il valore decimale del registro SPBRG deve essere pari a 25.
Nel codice tale valore viene memorizzato in una costante dichiarata a inizio programma denominata *BAUD_USART.*

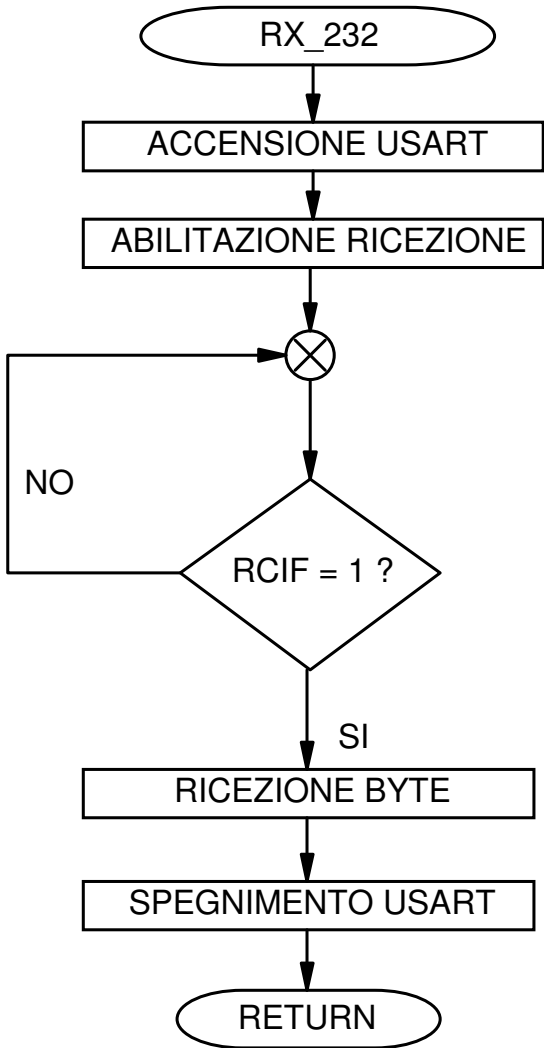
Si passa ora all'analisi del diagramma di flusso e del relativo codice di trasmissione di un singolo byte.



```
TX_232
    BANKSEL RCSTA               ; Bank 0
    BSF     RCSTA,SPEN          ; Enable USART
    BANKSEL TXSTA               ; Bank 1
    BSF     TXSTA,TXEN          ; Transmit enable
    BANKSEL BYTE_232_TX         ; Bank 0
    MOVF    BYTE_232_TX,0       ; Move TX value into W
    MOVWF   TXREG               ; Load data
    BANKSEL TXSTA               ; Bank 1
    BTFSS   TXSTA,TRMT          ; TSR empty ?
    GOTO    $-1                 ; no, else....
    BANKSEL RCSTA               ; Bank 0
    BCF     RCSTA,SPEN          ; Disable USART
    BSF     PORTC,6             ; Put high TX pin
RETURN                          ; Return
```

La prima operazione che si fa è quella di accendere la periferica USART settando il bit SPEN (bit 7) del registro RCSTA. Subito dopo si abilita la periferica per la modalità di trasmissione settando il bit TXEN (bit 5) del registro TXSTA.
A questo punto un apposito registro creato nella memoria ram del microcontrollore denominato BYTE_232_TX, contenente il byte che si desidera inviare, viene trasferito nel registro buffer TXREG in modo da iniziare la trasmissione. La fine della trasmissione viene verificata facendo un test sul bit TRMT (bit 1) del registro TXSTA, che si porta da 0 a 1 quando il registro TSR si è vuotato e cioè quando l'intero byte è stato shiftato in uscita.
Terminata la trasmissione si provvede a disabilitare la periferica resettando il bit SPEN e a portare a livello logico 1 il relativo piedino del microcontrollore facente capo alla linea di trasmissione in modo da evitare rimbalzi indesiderati.

Si passa ora all'analisi del diagramma di flusso e del relativo codice di ricezione di un singolo byte.

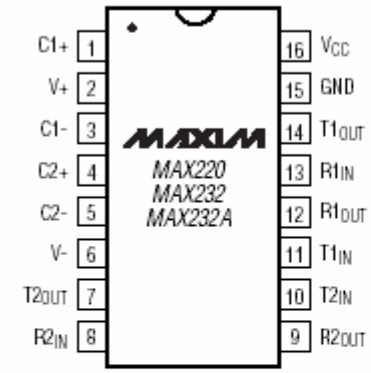



```
RX_232
    BANKSEL RCSTA               ; Bank 0
    BSF     RCSTA,SPEN          ; Enable USART
    BSF     RCSTA,CREN          ; Enable reception
    BTFSS   PIR1,RCIF           ; Test Interrupt Flag for end of transmission
    GOTO    $-1
    MOVF    RCREG,0             ; Move received byte into W
    MOVWF   BYTE_232_RX         ; Save byte
    BCF     RCSTA,SPEN          ; Disable USART
RETURN                          ; Return
```

Come nella subroutine di trasmissione, la prima e l'ultima operazione consistono rispettivamente nell'accensione e nello spegnimento della periferica USART.
La ricezione del byte viene fatta abilitando la periferica alla ricezione per mezzo del bit CREN (bit 4) del registro RCSTA.
La verifica del termine del trasferimento viene fatta testando il flag usato di solito nelle applicazioni di interrupt e cioè il bit RCIF (bit 5) del registro PIR1.
Questo bit infatti viene settato dalla periferica ogniqualvolta il buffer di ricezione risulta pieno e quindi quando l'intero byte è stato ricevuto.
Infine si provvede a trasferire il dato appena ricevuto presente nel registro RCREG in un altro registro di lavoro denominato BYTE_*232*_RX.

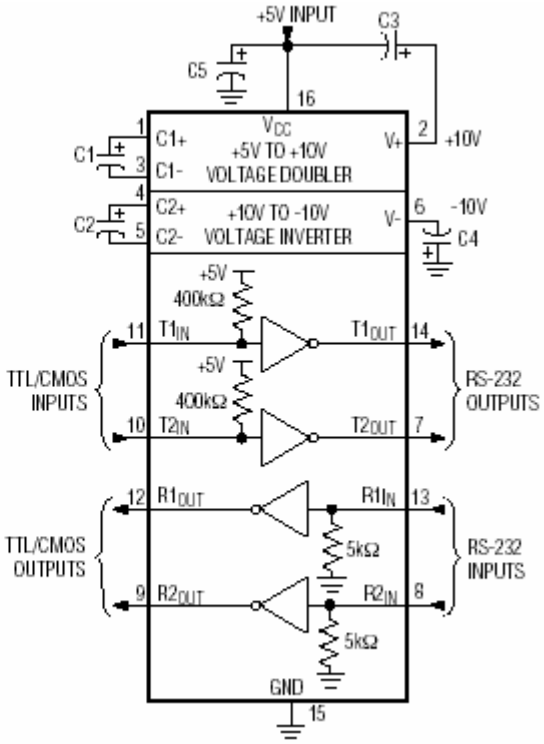
3.4 - Architettura hardware per il controllo RS232

I segnali RS232 hanno delle caratteristiche elettriche non compatibili direttamente con i segnali TTL del microcontrollore. Infatti lo 0 logico corrisponde ad una tensione che può variare da 5 a 12 V, mentre l'1 logico corrisponde ad una tensione che può variare da –5 a –12 V. La fascia compresa tra –3 e 3 V costituisce la fascia di incertezza nella quale non risulta definito alcuno dei due stati logici.
Quindi per adattare tali livelli di tensione è necessario interporre tra PC e microcontrollore un traslatore di livelli. A tal scopo si impiega un integrato specifico progettato per queste applicazioni cioè il 232, dal quale è possibile ottenere delle tensioni di ±10 V anche se si dispone di un'alimentazione singola di +5V.



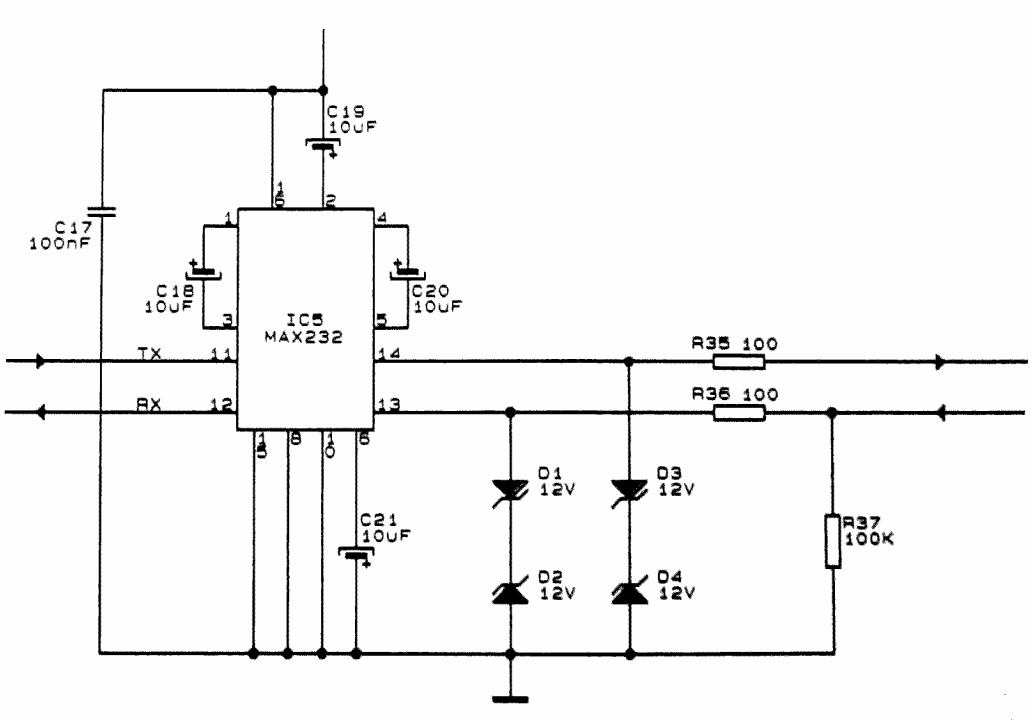
DIP/SO

CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1



Questo componente è costituito fondamentalmente da tre parti: il doppio convertitore di tensione DC\DC, le porte driver di trasmissione e le porte di ricezione.

- Il doppio convertitore DC\DC è formato da due pompe di carica che convertono la tensione positiva di 5V nelle tensioni ±10V necessarie per i livelli logici RS232. Il primo convertitore usa la capacità C1 per duplicare la tensione di +5V a +10V sulla capacità C3 all'uscita V+. Il secondo convertitore usa la capacità C2 per invertire la tensione duplicata di +10V a –10V sulla capacità C4 sull'uscita V-.
- Le porte driver di trasmissione servono per traslare i livelli TTL in livelli RS232 compatibili sfruttando opportunamente il doppio convertitore DC\DC. Le porte di ricezione compiono il lavoro inverso e adattano i segnali RS232 in segnali TTL compatibili.



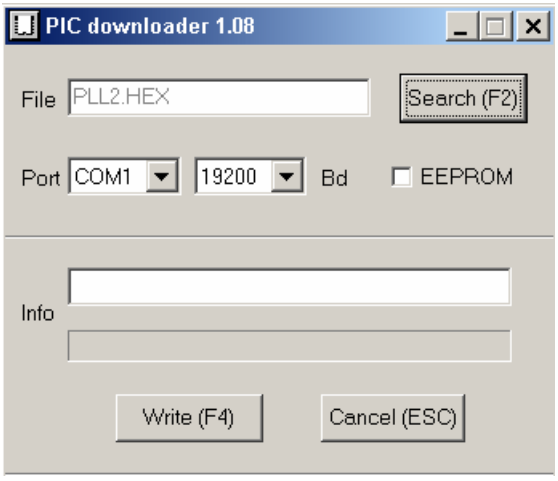
I condensatori della pompa di carica sono stati collegati come da datasheet. La loro capacità è stata aumentata a 10 µF visto che durante le prove di comunicazione è stata riscontrata un'eccessiva frequenza negli errori.
I due diodi zener collegati su entrambe le linee che fanno capo al lato PC servono per proteggere l'hardware da eventuali extratensioni si positive che negative. Le resistenze poste in serie hanno il compito di disperdere un eventuale eccesso di energia presente lungo le linee. La resistenza R37 invece protegge l'ingresso dell'integrato dalla possibile presenza di cariche elettrostatiche.
Per filtrare la radiofrequenza è stato posto in parallelo ai terminali dell'integrato un condensatore al poliestere da 100 nF.

4 – Descrizione generale della parte software

In questo capitolo si tratta in generale la parte di programmazione del microcontrollore e la descrizione riassuntiva del codice del microcontrollore.

4.1 – Programmazione del microcontrollore

La programmazione del microcontrollore può essere effettuata in più modi. Il metodo classico consiste nell'estrazione dell'integrato dalla scheda e l'inserimento in un opportuno programmatore (per qualsiasi evenienza, i piedini di programmazione sono accessibili dal connettore della scheda).
Il metodo non classico consiste nel lasciare il microcontrollore sulla scheda ed effettuare la sua programmazione mediante il circuito RS232. Questo è il metodo del bootloader e risulta molto comodo e veloce.
Con questo metodo è necessario effettuare in ogni caso la prima programmazione in modo classico con un codice di bootloader. Questo codice è un normale programma che si inserisce all'inizio della memoria e gestisce le prime operazioni dal momento che si alimenta il microcontrollore o comunque dopo un reset.
Questo programma non fa altro che abilitare la scrittura della memoria flash e gestire la sua modifica mediante i nuovi dati che arrivano dalla porta USART del microcontrollore.
Solo dopo questa operazione di precaricamento del codice bootloader si possono effettuare le successive programmazioni, per mezzo di uno specifico programma funzionante in ambiente Windows, con il metodo bootloader e con il collegamento RS232.
Quindi il programma bootloader caricato la prima volta mediante il metodo classico, è sempre residente nella prima parte della memoria flash e non fa altro che accodare dopo di lui le istruzioni del programma caricato tramite la porta RS232. Il codice bootloader viene eseguito una volta sola dal momento di applicazione dell'alimentazione o dal momento di pressione del tasto di reset. Infatti questo codice sposta l'indirizzo della prima istruzione del codice caricato via RS232 dopo l'ultimo indirizzo del codice di bootloader.
Il codice di bootloader deve essere specifico per il tipo di microcontrollore usato. E' importante inoltre sceglierlo opportunamente in base alla frequenza del quarzo del microcontrollore e alla velocità di trasmissione RS232.
Di seguito si riporta un screen shot del programma bootloader:



Mediante il tasto "Search" si sceglie il codice da voler caricare già convertito in formato esadecimale, si seleziona la porta seriale del PC sulla quale è stata collegata la scheda con la relativa velocità di trasmissione e infine si preme sul tasto "Write". La programmazione non parte immediatamente ma parte solo all'istante nel quale viene premuto il tasto reset sulla scheda. Infatti bisogna consentire al microcontrollore di eseguire il programma bootloader residente a inizio memoria.

4.2 – Programma completo e main di prova per i vari blocchi

Tra gli allegati di questo documento si riporta il programma completo di gestione delle varie periferiche. In questo programma il main risulta vuoto in quanto come è stato accennato all'inizio, in base alle specifiche iniziali, non è stato possibile scriverne uno completo e definitivo.
Si riportano inoltre i vari main usati per testare i singoli blocchi della scheda in modo da poterne comprendere e verificare il funzionamento. Per eseguire questi programmi è necessario disporre di un programma di comunicazione RS232, anche funzionante in modalità DOS.
Si riporta inoltre una breve spiegazione del funzionamento di ogni programma main.

- *Programma di test della comunicazione seriale RS232*: il programma visualizza sullo schermo la parola "prova" e poi aspetta l'invio di un carattere da parte del PC per poi a sua volta rivisualizzarlo sullo schermo. Quello che si fa è di creare un anello per verificare sia la ricezione e la trasmissione. Quest'ultima operazione può essere fatta al massimo cinque volte prima che il programma termini.
- *Programma di test del sintetizzatore di frequenza*: il programma permette la scrittura via RS232 in sequenza dei registri N_COUNTER_3, N_COUNTER_2 e N_COUNTER_1 e poi di lanciare il protocollo completo di gestione del sintetizzatore. Gli altri registri sono già configurati in maniera corretta come da prova di laboratorio assieme al circuito completo PLL. Quindi la configurazione dei tre registri precedenti permette l'aggancio su una determinata frequenza. L'accensione del led D10 segnala l'avvenuto aggancio mentre l'accensione del led D11 segnala al contrario il non avvenuto aggancio da parte del circuito PLL.
- *Programma di test della memoria EEPROM*: il programma prevede la scrittura delle prime celle della memoria con il codice ASCII dei caratteri della parola "CIAO" e prevede inoltre la lettura delle precedenti celle e la loro visualizzazione sullo schermo del PC. In questo modo si verifica sia la corretta scrittura della memoria sia la corretta lettura.
- *Programma di test del convertitore DAC*: tramite il PC si riempiono rispettivamente i registri dati MSB e LSB del DAC e poi si constata con un voltmetro la corrispondente tensione all'uscita del DAC. Il codice di Device Select è già impostato correttamente via software.
- *Programma di test del convertitore ADC*: tramite il PC si visualizza sullo schermo il valore binario dei due registri verificando l'effettiva conversione. La tensione sull'ingresso del convertitore, per esempio, può essere variata con un trimmer. Il canale selezionato è lo zero.
- *Programma di test dei blocchi a transistor*: il programma provvede ad accendere e spegnere sequenzialmente i blocchi di media – bassa potenza. L'effetto che si vedrà è quindi lo scorrimento della luce dei led prima in un verso e dopo nell'altro.
- *Programma di test dell'interrupt*: usando un tasto, con l'ausilio di un circuito flip – flop antirimbalzo, si può constatare l'accensione e lo spegnimento dei transistor. Quello che si fa è provare ad interrompere il protocollo di scrittura e lettura della memoria eeprom.

4.3 – Specifiche software

Funzione della subroutine	*Nome della subroutine*	*Registro di ingresso*	*Registro di uscita*	*Funzione del registro*	*Tempo di esecuzione della subroutine*
Gestione del sintetizzatore di frequenza	SYNTHESIZER	R_COUNTER_3		Bit dal 23 al 16 del registro R	4,519 ms
		R_COUNTER_2		Bit dal 15 al 8 del registro R	
		R_COUNTER_1		Bit dal 7 al 0 del registro R	
		N_COUNTER_3		Bit dal 23 al 16 del registro N	
		N_COUNTER_2		Bit dal 15 al 8 del registro N	
		N_COUNTER_1		Bit dal 7 al 0 del registro N	
		FUNCTION_LATCH_3		Bit dal 23 al 16 del registro FUNCTION	
		FUNCTION_LATCH_2		Bit dal 15 al 8 del registro FUNCTION	
		FUNCTION_LATCH_1		Bit dal 7 al 0 del registro FUNCTION	
Scrittura di un byte sulla memoria EEPROM (8Kbit)	WR_TO_EEPROM	DEV_SEL_EEPROM		Indirizzo IIC hardware della memoria (10100000)	1,949 ms
		ADDRESS_EEPROM		Indirizzo della locazione di memoria	
		DATA_EEPROM		Dato da scrivere	
Lettura di un byte dalla memoria EEPROM (8Kbit)	RD_TO_EEPROM	DEV_SEL_EEPROM		Indirizzo IIC hardware della memoria	2,67 ms
		ADDRESS_EEPROM		Indirizzo della locazione di memoria	
			DATA_EEPROM	Dato da leggere	
Conversione digitale - analogica	DAC	DEV_SEL_DAC		Indirizzo IIC hardware del convertitore (00011000)	1,94 ms
		DAC_MSB		Byte più significativo di dati	
		DAC_LSB		Byte meno significativo di dati	
Conversione analogica - digitale	ADC	ADCON0		Selezione del canale di converisone	42.4 us
			ADC_MSB	Bit 9 e 8 di conversione	
			ADC_LSB	Bit dal 7 al 0 di conversione	
Trasmissione di un byte tramite la porta RS232	TX_232	BYTE_232_TX		Byte da trasmettere	2,087 ms
Ricezione di un byte tramite la porta RS232	RX_232		BYTE_232_RX	Byte ricevuto	2,086 ms
Ritardo programmabile	DL_TMR0	Modifica della costante K			Funzione di K
Sequenza di accensione dei transistor	TRANSISTOR_ON				Funzione di K
Sequenza di spegnimento dei transistor	TRANSISTOR_OFF				Funzione di K

5 – Caratteristiche del circuito di controllo

In questo capitolo si riassumono le caratteristiche del microcontrollore e della scheda. Per quest'ultima in particolare viene fornita la documentazione hardware.

5.1 – Specifiche del microcontrollore

- CPU*: RISC a alta esecuzione;*
- capacità registri*: 8 bit;*
- memoria programma: *CMOS FLASH;*
- capacità della memoria programma: *8K x 14 words;*
- capacità della memoria RAM: *368 x 8 bytes;*
- profondità dello stack: *8 livelli;*
- interrupt: *più di 14 sorgenti;*
- programmazione: *seriale a 2 pin;*
- periferiche (utilizzate): *Timer0 a 8 bit, Timer2 a 8 bit, ADC a 10 bit, MSSP come SPI e IIC, USART.*
- alimentazione: *5V;*
- assorbimento: *25 mA;*
- assorbimento in standby: *minore di 1 uA.*

5.2 – Specifiche hardware del circuito di controllo

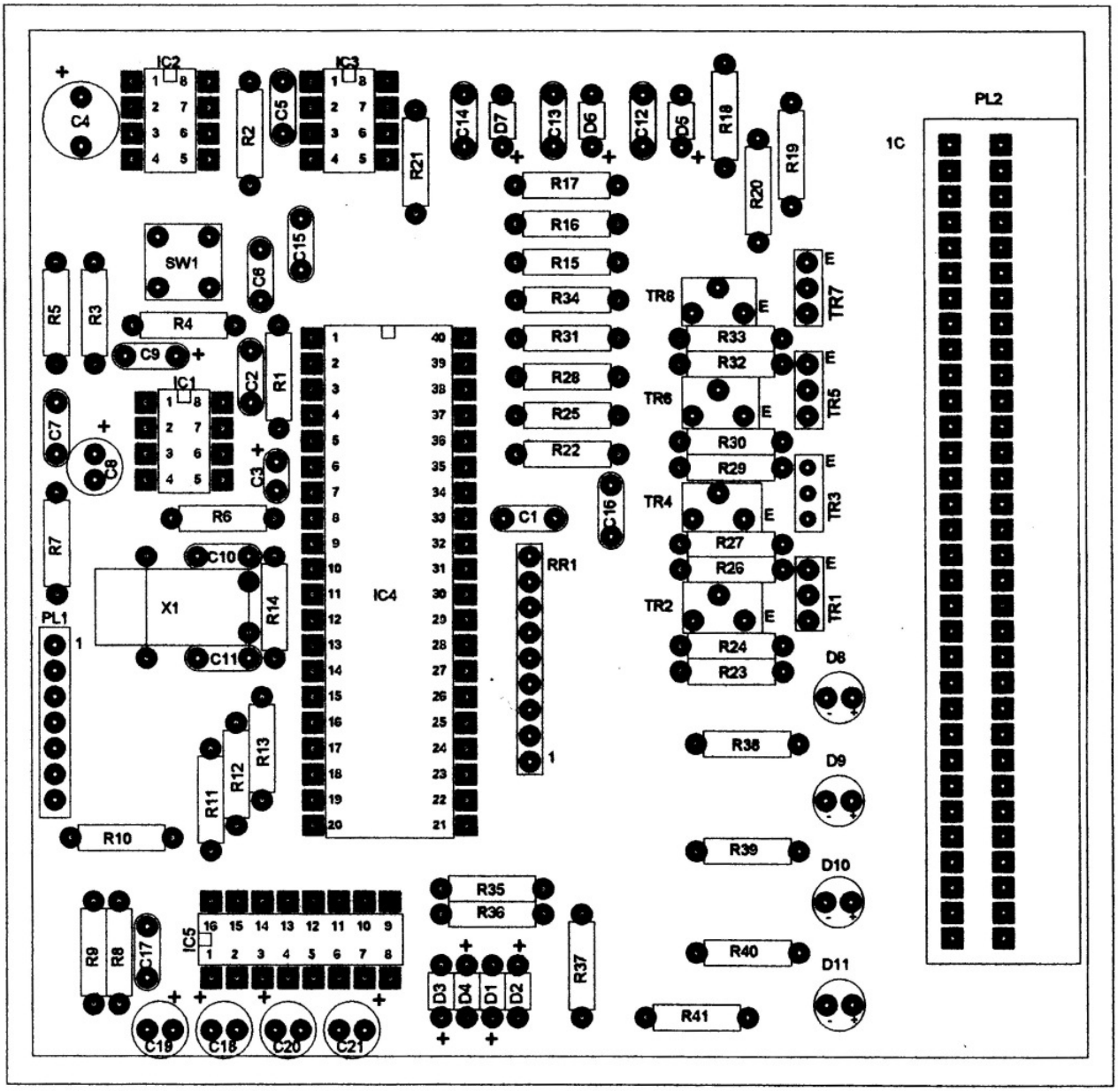
Connessioni:

- pettine a 7 pin per il controllo del sintetizzatore (TAB. 3);
- connettore DIN 41612 a 64 pin per tutte le altre funzioni (TAB.2);
- comunicazione seriale RS232 accessibile da connettore DIN 41612 (TAB.2);

Alimentazione:

- pin 16C connettore DIN 41612 per l'alimentazione dei circuiti integrati (+5V, 50mA);
- pin 10C connettore DIN 41612 per l'alimentazione dei transistor (+5V, 2A);
- pin 1C-17C-18C-19C-20C-21C-22C connettore DIN 41612 per le masse.

5.3 – Planimetria, lista dei componenti e piedinatura dei connettori della scheda



TAB. 1 – Elenco componenti

RESISTENZE:		C9	6,8 uF tant. 16 V
R1	10 OHM 1/4 W	C10	22 pF cer. 50 V
R2	10 OHM 1/4 W	C11	22 pF cer. 50 V
R3	10 KOHM 1/4 W	C12	100 nF pol. 50 V
R4	1 KOHM 1/4 W	C13	100 nF pol. 50 V
R5	100 OHM 1/4 W	C14	100 nF pol. 50 V
R6	10 KOHM 1/4 W	C15	100 nF pol. 50 V
R7	470 OHM 1/4 W	C16	1 nF pol. 50 V
R8	33 KOHM 1/4 W	C17	100 nF pol. 50 V
R9	33 KOHM 1/4 W	C18	10 uF elettr. 25 V
R10	33 KOHM 1/4 W	C19	10 uF elettr. 25 V
R11	100 KOHM 1/4 W	C20	10 uF elettr. 25 V
R12	100 KOHM 1/4 W	C21	10 uF elettr. 25 V
R13	100 KOHM 1/4 W		
R14	10 MOHM 1/4 W	**TRANSISTOR:**	
R15	100 OHM 1/4 W	TR1	BC327
R16	100 OHM 1/4 W	TR2	2N2222
R17	100 OHM 1/4 W	TR3	BC327
R18	100 KOHM 1/4 W	TR4	2N2222
R19	100 KOHM 1/4 W	TR5	BC327
R20	100 KOHM 1/4 W	TR6	2N2222
R21	10 KOHM 1/4 W	TR7	BC327
R22	10 KOHM 1/4 W	TR8	2N2222
R23	100 KOHM 1/4 W		
R24	220 OHM 1/4 W	**DIODI:**	
R25	10 KOHM 1/4 W	D1	ZENER 12 V 1/4 W
R26	100 KOHM 1/4 W	D2	ZENER 12 V 1/4 W
R27	220 OHM 1/4 W	D3	ZENER 12 V 1/4 W
R28	10 KOHM 1/4 W	D4	ZENER 12 V 1/4 W
R29	100 KOHM 1/4 W	D5	ZENER 5,6 V 1/4 W
R30	220 OHM 1/4 W	D6	ZENER 5,6 V 1/4 W
R31	10 KOHM 1/4 W	D7	ZENER 5,6 V 1/4 W
R32	100 KOHM 1/4 W	D8	DIODO LED
R33	220 OHM 1/4 W	D9	DIODO LED
R34	10 KOHM 1/4 W	D10	DIODO LED
R35	100 OHM 1/4 W	D11	DIODO LED
R36	100 OHM 1/4 W		
R37	100 KOHM 1/4 W	**INTEGRATI:**	
R38	470 OHM 1/4 W	IC1	TL7705
R39	470 OHM 1/4 W	IC2	24C08
R40	470 OHM 1/4 W	IC3	AD5321
R41	470 OHM 1/4 W	IC4	PIC16F877
RR1	RETE RESISTIVA	IC5	MAX232
	100 KOHM 8 PIN + 1		
		VARIE:	
CONDENSATORI:		PL1	PETTINE 7 PIN
C1	100 nF pol. 50 V	PL2	DIN 41612 64 PIN
C2	100 nF pol. 50 V	X1	QUARZO 16 MHZ
C3	6,8 uF tant. 16 V	SW1	PULSANTE RESET
C4	470 uF elettr. 16 V		
C5	100 nF pol. 50 V		
C6	100 nF pol. 50 V		
C7	100 nF pol. 50 V		
C8	10 uF elettr. 25 V		

TAB. 2 – Connessioni connettore DIN 41612 a 64 pin

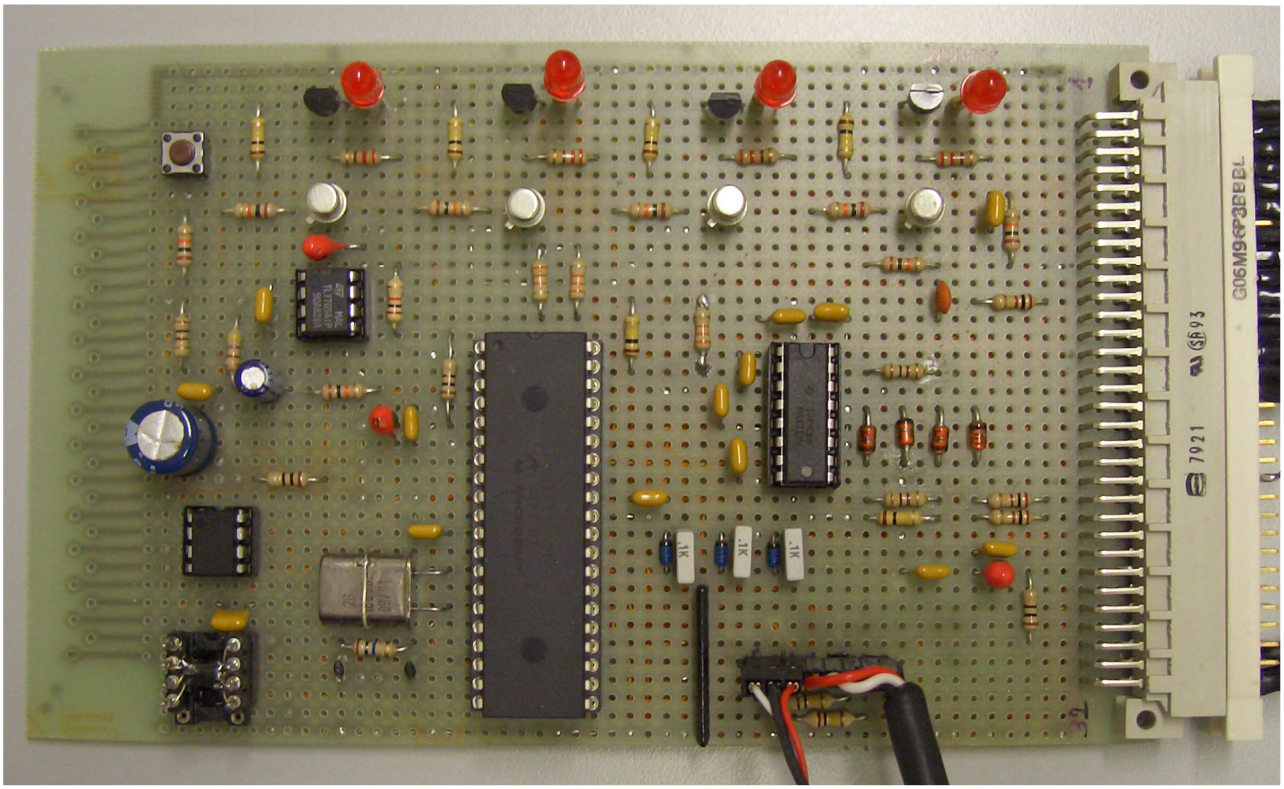
PIN	FUNZIONE	I / O	ZIN / ZOUT	LIVELLI LOGICI (L< 0,8 V, H>2,4)
1C	GND	/	/	/
2C	DAC_VOUT	O	10 KOHM	0 / + 5V
3C	MCLR / VPP	I	1 KOHM	H = ATTIVO
4C	PGD	I/O	100 KOHM	DATA_PROG / H = ATTIVO
5C	PGC	I/O	100 KOHM	CLOCK_PROG / H= ATTIVO
6C	PGM	I	100 KOHM	MODE_PROG / H= ATTIVO
7C	AD2	I	1 MOHM	0 / + 5V
8C	AD1	I	1 MOHM	0 / + 5V
9C	AD0	I	1 MOHM	0 / + 5V
10C	VCC_TX	I	/	/
11C	TX_SEQ_4	O	10 OHM	+ 5V = ATTIVO
12C	TX_SEQ_3	O	10 OHM	+ 5V = ATTIVO
13C	TX_SEQ_2	O	10 OHM	+ 5V = ATTIVO
14C	TX_SEQ_1	O	10 OHM	+ 5V = ATTIVO
15C	INT_TX_IN	I	10 KOHM	L = ATTIVO
16C	VCC + 5V	I	/	/
17C	GND	/	/	/
18C	GND	/	/	/
19C	GND	/	/	/
20C	GND	/	/	/
21C	GND	/	/	/
22C	GND	/	/	/
23C	NON DEFINITO	O	100 KOHM	H = ATTIVO
24C	NON DEFINITO	O	100 KOHM	H = ATTIVO
25C	NON DEFINITO	O	100 KOHM	H = ATTIVO
26C	NON DEFINITO	O	100 KOHM	H = ATTIVO
27C	NON DEFINITO	O	100 KOHM	H = ATTIVO
28C	NON DEFINITO	O	100 KOHM	H = ATTIVO
29C	NON DEFINITO	O	100 KOHM	H = ATTIVO
30C	MUTE	O	100 KOHM	H = ATTIVO
31C	RS232_TX_OUT	O	5 KOHM	+ 12 V / - 12 V
32C	RS232_TX_IN	I	5 KOHM	+ 12 V / - 12 V
1A - 32A	GND	/	/	/

TAB. 3 – Connessioni pettine sintetizzatore a 7 pin

PIN	FUNZIONE	I / O	ZIN / ZOUT	LIVELLI LOGICI (L< 0,8 V, H>2,4)
1	GND	/	/	/
2	GND	/	/	/
3	CE	O	100 KOHM	H = ATTIVO
4	LE	O	100 KOHM	H = ATTIVO
5	MUXOUT	I	100 KOHM	H = ATTIVO
6	SCL	I/O	33 KOHM	H = ATTIVO
7	SDO	O	33 KOHM	H = ATTIVO

5.4 – Realizzazione della scheda

La prima versione del circuito è stata realizzata su una basetta del tipo mille fori e i collegamenti sono stati effettuati mediante la saldatura di sottili fili.
Su tale scheda sono state effettuate le prime prove e le prime verifiche di funzionamento dei vari blocchi.
Essendo presente un hardware non complesso, i problemi iniziali sono stati causati dal codice programma che in alcune parti non funzionava. Sono state fatte molte modifiche al programma prima di ottenere una versione definitiva.
Solo a lavoro concluso si è deciso di progettare uno stampato per ottenere un circuito più compatto e ordinato.
A tal fine è stato sviluppato un master a doppia faccia praticamente indispensabile per ottenere delle dimensioni abbastanza ridotte visto comunque anche l'utilizzo di componenti discreti e non del tipo a montaggio superficiale.
Per la realizzazione del circuito stampato si sono in principio disegnati i vari componenti e i collegamenti su un foglio di carta cercando di ottenere allo stesso tempo una certa compattezza e una certa regolarità senza dover incorrere in ponti per l'incrocio delle piste.
Successivamente è stato disegnato il master mediante un software apposito.
Una volta stampato il master su dei fogli lucidi, si è usato un bromografo per imprimere le piste sulla vernice fotosensibile della basetta.
Il risultato ottenuto è stato soddisfacente in quanto si è ottenuta la fedele riproduzione del master su entrambi i lati della basetta.
Dopo l'incisione per mezzo del percloruro ferrico si è provveduto ad argentare le piste e coprire l'intera basetta mediante una vernice protettiva trasparente per evitare la futura ossidazione.



- Un'immagine del primo prototipo realizzato su basetta millefori -

6 – Conclusioni

La parte più importante sviluppata in questo progetto e cioè il controllo del sintetizzatore di frequenza è stato più volte testato e verificato in laboratorio assieme al circuito PLL. Si è visto un aggancio sulla frequenza desiderata constatando l'effettivo funzionamento sul campo di entrambe le schede.
In futuro si può provare ad aumentare la frequenza di clock del protocollo SPI in modo da ridurre ulteriormente i tempi di controllo.
Si può fare lo stesso anche per il bus IIC della memoria EEPROM e del convertitore digitale – analogico.
Se fosse necessario lavorare con più byte per volta con la memoria EEPROM, si può pensare di modificare la subroutine utilizzata, impiegando altri metodi di scrittura e lettura che la memoria permette.
Per il DAC si può creare una subroutine che faccia una lettura di ritorno dei valori impostati al fine di ottenere un'ulteriore verifica dei dati inviati.
Per il convertitore analogico – digitale invece è possibile creare una subroutine la quale effettui più acquisizioni in un ristretto lasso di tempo con lo scopo di campionare più volte il segnale. Questo per evitare che qualche disturbo falsi la lettura dei canali analogici.
È possibile anche utilizzare un registro apposito supplementare per selezionare prima di ogni chiamata il canale appropriato.
Come ultima modifica al programma del microcontrollore, si può aggiungere una subroutine che faccia entrare nella modalità di standby il microcontrollore in modo da abbassare i consumi nei momenti di non utilizzo.
Se in futuro il ricetrasmettitore fosse ampliato ulteriormente e fossero necessarie funzioni supplementari, si può prendere in considerazione di implementare una versione successiva del microcontrollore PIC e in particolare la serie 18. La piedinatura infatti per i microcontrollori a 40 pin rimane invariata. Queste ultime generazioni possono lavorare anche con quarzi fino a 40 MHz.
I blocchi a transistor sono stati progettati per essere alimentati a 5V ma in qualsiasi momento risultano modificabili per lavorare a tensioni superiori cambiando il valore delle resistenze di base dei transistor.
Se invece è necessario lavorare con assorbimenti di corrente maggiori allora risulta indispensabile sostituire il tipo di transistor impiegato sullo stadio finale. In generale quindi, devono essere ripresi gli aspetti legati all'assorbimento di corrente e all'alimentazione.
Un controllo della potenza del trasmettitore non è stato implementato e quindi in futuro va aggiunto. A tal proposito può risultar utile un ulteriore convertitore DAC.
Per il controllo della scheda durante le varie prove si è utilizzato un PC assieme ad un banale software di comunicazione seriale in modalità DOS. Tale programma altro non faceva che inviare o ricevere dei byte di dati. In futuro è possibile sviluppare un programma, per esempio in Visual Basic, in modo da avere a disposizione un'adeguata interfaccia grafica di controllo più completa e specifica alle varie funzioni da svolgere.
Per il controllo del sintetizzatore di frequenza, per esempio, risulta utile l'impostazione direttamente da tastiera del valore di frequenza desiderato o il canale su cui agganciarsi.
Assieme a questo programma va scritto un opportuno main principale per il microcontrollore che richiami adeguatamente tutte le varie subroutine.

Ringraziamenti

Desidero ringraziare il Prof. Mario Fragiacomo per avermi dato l'opportunità di prendere parte al progetto Atmocube e di aver potuto svolgere questa tesi.
Ringrazio inoltre la Elcon Elettronica, in particolare il Sig. Luciano Generali, per avermi seguito in questo progetto con costanza e grande disponibilità, ma soprattutto per avermi fatto ulteriormente capire e consolidare, direttamente e indirettamente, importanti *cose*.
Ringrazio infine, ma non per meno importanza, tutte le persone che hanno contribuito con i loro semplici ma importantissimi gesti, a farmi passare indispensabili momenti di felicità, e tutte le persone che mi sono state vicine in questi anni, sostenendomi, credendo nelle mie capacità e facendomi trovare la giusta forza e il giusto equilibrio, al fine di superare gli ostacoli più difficili e raggiungere i miei motivati obiettivi.

Bibliografia

- Microchip - PIC16F877 Datasheet, 2001;
- Microchip - AN735 Using the PICmicro MSSP Module for Master IIC Communications, 2002;
- Microchip - Section 17 Master Synchronous Serial Port (MSSP), 1997;
- Microchip - SPI Overview and Use of the PICmicro Serial Peripheral Interface;
- Microchip - IIC Overview and Use of the PICmicro MSSP IIC Interface with a 24xx01x EEPROM;
- Microchip - AN744 Asynchronous Communications with the PICmicro USART, 2003;
- Microchip - Use the USART in Asynchronous Mode;
- Microchip - AN851 A FLASH Bootloader for PIC16 and PIC18 Devices, 2002;
- Philips Semiconductors - The IIC-bus specification, 2000;
- Philips Semiconductors - IIC Logic Family Overview 2004;
- Analog Devices - RF PLL Frequency Synthesizers AD4116 / 17 / 18 Datasheet, 2001;
- Analog Devices - DAC AD5301 / 11 / 21 Datasheet, 1999;
- StMicroelectronics - M24c16 / 08 / 04 / 02 / 01 Datasheet, 2004;
- StMicroelectronics - 2N2222, BC327 Transistor Datasheet;
- Texas Instrument - TL7705 Supply Voltage Supervisor Datasheet, 1995;
- Maxim - MAX232 RS232 Drivers, Receivers, 2003.

Allegati

- Programma completo.
- Main di prova.
- Master lato rame.
- Master lato componenti.
- Schema elettrico.